A Quadratic Programming Approach to Simultaneous Buffer Insertion/Sizing and Wire Sizing

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

***Abstract*—In this paper, we present a completely new approach to the problem of delay minimization by simultaneous buffer insertion and wire sizing for a wire. We show that the problem can be formulated as a convex quadratic program, which is known to be solvable in polynomial time. Nevertheless, we explore some special properties of our problem and derive an optimal and very efficient algorithm modified active set method (MASM) to solve the resulting program. Given m buffers and a set of n discrete choices of wire width, the running time of our algorithm is $O(mn^2)$ and is independent of the wire length in practice. For example, an instance of 100 buffers and 100 choices of wire width can be solved in 0.92 s. In addition, we extend MASM to consider simultaneous buffer insertion, buffer sizing, and wire sizing. The resulting algorithm MASM-BS is again optimal and very efficient. For example, with six choices of buffer size and 10 choices of wire width, the optimal solution for a 15 000 μm long wire can be found in 0.05 s. Besides, our formulation is so versatile that it is easy to consider other objectives like wire area or power dissipation, or to add constraints to the solution. Also, wire capacitance lookup tables, or very general wire capacitance models which can capture area capacitance, fringing capacitance, coupling capacitance, etc. can be used.**

***Index Terms*—Buffer insertion, buffer sizing, interconnect, optimization, performance optimization, physical design, quadratic programming.**

I. Introduction

In the past, gate delay was the dominating factor in circuit design. However, as the feature size of very large scale integration (VLSI) devices continues to decrease, interconnect delay becomes increasingly important. Nowadays, feature size has been down to 0.25 μm in advance technology. 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 [11]. It is predicted in the Semiconductor Industry Association (SIA) roadmap [27] that the feature size will be reduced to 0.18 μm by 1999 and 0.13 μm by 2003. So we expect the significance of interconnect delay will further increase in near future.

Buffer insertion and buffer sizing have been known for a long time to be effective techniques to reduce delay and have been extensively studied in the literature [1], [14], [19], [20], [23], [28]. As delay due to interconnect wire becomes more and more important, [13] demonstrated that wire sizing is also very effective in reducing interconnect delay. Almost all the previous approaches to interconnect delay optimization that use wire sizing divide a wire into small fixed-length segments and optimize the width of each segment iteratively. Some examples are [4], [9], [13], [26], and [29] for wire sizing alone, [2], [8], [12], and [25] for simultaneous buffer sizing and wire sizing, and [22] for simultaneous buffer insertion, buffer sizing, and wire sizing. See [11] for a comprehensive survey. In order to obtain accurate results, a wire usually needs to be divided into a large number of segments.

For wire sizing alone, [3], [5], [16], and [17] considered a variant which does not divide a wire into segments. For the problem they considered, the set of choices of wire width is a continuous interval. Therefore, the resulting function describing the wire width is continuous. However, in practice, a discrete set of choices of wire width is usually used. In that case, the continuous wire sizing solution needs to be rounded.

In [7], we considered buffer insertion, buffer sizing, and wire sizing simultaneously and a closed-form optimal solution is obtained. For the problem they considered, the lengths of the wire segments are treated as variables and hence not fixed. However, in that paper, the wire widths are also taken from a continuous interval. Besides, only wire area capacitance is considered. (Terms like wire fringing capacitance are ignored.)

In this paper, we consider the problem of interconnect delay minimization for a wire by buffer insertion, buffer sizing and wire sizing under the Elmore delay model [15]. We first consider the simultaneous buffer insertion and wire sizing problem $\mathcal{BIWS}$, shown at the bottom of the next page. Note that for the problem $\mathcal{BIWS}$, the wire is not divided into fixed-length segments. None of the previous results can be applied to our problem and get exact solutions.

We not only propose a more general problem formulation which does not divide a wire into fixed-length segments, but also present a completely new approach for interconnect delay optimization which has many advantages over previous approaches which optimize the widths of fixed-length segments.

1) Instead of solving $\mathcal{BIWS}$ directly, we solve an equivalent problem $\mathcal{BIWS}'$ which is introduced in Section III. The problem $\mathcal{BIWS}'$ has much less variables than the problem formulated according to the traditional approach of dividing a wire into small fixed-length segments. If m buffers are to be inserted and a set of n choices of wire width is given, $\mathcal{BIWS}'$ will have $(m+1)n$ variables no matter how long the wire is. As

Manuscript received April 24, 1998; revised September 5, 1998. This paper was recommended by Associate Editor M. Pedram.

The authors are with the Department of Computer Sciences, University of Texas at Austin, Austin, TX 78712 USA (e-mail: cnchu@cs.utexas.edu; wong@cs.utexas.edu).

Publisher Item Identifier S 0278-0070(99)03965-2.

in practice, usually only a few buffers and a few choices for the wire width are allowed, $(m+1)n$ is a small number. Moreover, the problem $\mathcal{BIWS}'$ is completely equivalent to $\mathcal{BIWS}$ (not an approximation).

2) $\mathcal{BIWS}'$ can be solved optimally and very efficiently even for large m and n. We prove that our problem is a convex quadratic program. Convex quadratic programming has been well studied and can be solved efficiently by many public domain or commercial software systems. Nevertheless, we derive a tailored iterative algorithm MASM which is even more efficient. MASM runs in about n iterations in practice. Based on the observation that the inverse of the Hessian matrix of the convex quadratic program is tridiagonal, we prove that each iteration needs only linear time. For example, an instance of 100 buffers and 100 choices of wire width (i.e., 10 100 variables) can be solved in 0.92 s by our algorithm.
3) Buffer insertion is generally considered a hard problem and usually some heuristics or dynamic programming are needed to handle it. However, it is interesting to note how naturally and easily buffer insertion is handled in our approach. We observe that it is no more difficult than wire sizing alone.
4) Besides delay, our formulation can be easily extended to consider other objectives like wire area or power dissipation. For example, we can optimally solve the problems of minimizing a weighted sum of delay and wire area, minimizing delay with bounded area, minimizing area with bounded delay, etc. Our formulation also allows adding constraints to the solution. Moreover, our efficient algorithm MASM can still be applied to get optimal results.
5) We can use very general wire capacitance models which can capture area capacitance, fringing capacitance, coupling capacitance (the capacitance due to an adjacent parallel wire), etc. A wire segment is modeled as a π-type RC circuit as shown in Fig. 1. The capacitance of a wire segment of width h and length l is given by $c(h)l$, where $c(h)$ is the unit length wire capacitance for a segment of width h. The only restriction on c is that it has to be an increasing function from $\mathcal{R}^+$ to $\mathcal{R}^+$. For example, to model wire area capacitance, wire fringing capacitance and coupling capacitance at the same time, suppose the distance to an adjacent parallel wire is $d-h$ when the wire width is h. Then we can set $c(h) = c_0 h + c_f + c_c/(d-h)$, where c_0 is the unit wire area capacitance, c_f is the unit wire fringing capacitance and c_c is the unit wire coupling capacitance. The values of $c(h)$ for each h in H can also be obtained from a lookup table.

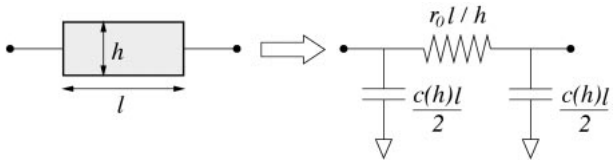


Fig. 1. The model of a wire segment of length l and width h by a π-type RC circuit. r_0 is the unit wire resistance. $c(h)$ is the wire capacitance per unit length for a segment of width h. We assume $c(h)$ is an increasing function in this paper.

In this paper, we also show how the algorithm MASM for $\mathcal{BIWS}$ can be applied to the simultaneous buffer insertion, buffer sizing, and wire sizing problem $\mathcal{BIBSWS}$ introduced in Section IV. In $\mathcal{BIWS}$, the number of buffers and the sizes of the buffers are given as input. If the number of buffers used is not given and there are several choices for the buffer sizes, the optimal solution can be found by trying all possible combinations of number of buffers and buffer sizes. For each combination, the corresponding problem is of the form of $\mathcal{BIWS}$ and, hence, can be solved by MASM. Nevertheless, we derive a simple lower bound on the delay which can be used to prune most of the combinations. The resulting algorithm MASM-BS is optimal and very efficient. For example, with six choices of buffer size and ten choices of wire width, the optimal simultaneous buffer insertion, buffer sizing, and wire sizing solution for a 15 000-μm-long wire can be found in 0.05 s.

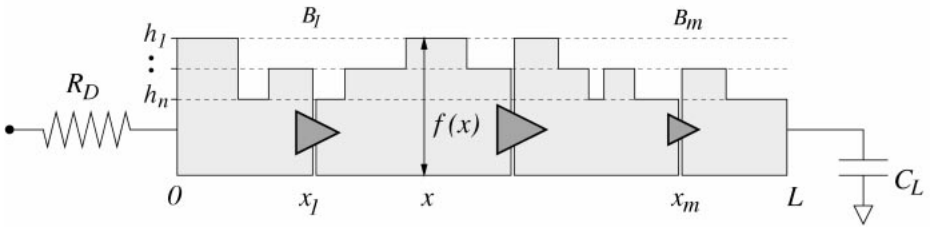
The paper is organized as follows. In Section II, we first consider the problem of wire sizing without buffer insertion. Once the formulation and the algorithm for wire sizing are understood, the extension to simultaneous buffer insertion and wire sizing is easy and is discussed in Section III. In Section IV, we discuss how to apply our result on Section III to determine the optimal number of buffers and to handle

---

PROBLEM $\mathcal{BIWS}$: The Simultaneous Buffer Insertion and Wire Sizing Problem

Given: wire length L, driver resistance R_D, load capacitance C_L, a set $H = \{h_1, \cdots, h_n\}$ of n choices of wire width such that $h_1 > \cdots > h_n$, and m buffers of sizes $B_1, \cdots, B_m$.

Determine: the positions $x_1, \cdots, x_m$ at which the buffers are inserted and the wire width $f(x)$ at each point x along the wire such that the delay from source to sink is minimized.



buffer sizing as well. In Section V, we further extend our results to consider other objectives like wire area or power, and to handle additional constraints to the solution. In Section VI, some experimental results to show the efficiency of our algorithms and some concluding remarks are given.

II. Wire Sizing

In this section, we consider the wire sizing problem $\mathcal{WS}$, shown at the bottom of the page, and derive a very efficient algorithm modified active set method (MASM) for it. In Section II-A, we first show that the problem $\mathcal{WS}$ is equivalent to a much simpler problem called $\mathcal{WS}'$. Then we prove in Section II-B that $\mathcal{WS}'$ can be formulated as a convex quadratic program. In Section II-C, we introduce the active set method. In particular, we present how the active set method can be applied to convex quadratic programs. In Section II-D, we prove the interesting observation that the inverse of the Hessian matrix of the convex quadratic program is tridiagonal. Then with this observation and the idea of the active set method, we derive our algorithm MASM. The extension to include buffer insertion is given in Section III.

For the rest of the paper, we use uppercase boldface letters to denote matrices and lowercase boldface letters to denote vectors. We use the convention that indexes of matrices and vectors start from one. To simplify the presentation, if we refer to an element of a matrix or a vector such that the index is out of range, we assume that the value is zero.

A. Simplification of the Problem

Consider the wire sizing problem $\mathcal{WS}$, where the optimal wire width is represented by a step function $f:[0,L]\rightarrow H$. We first prove that f must be a decreasing function. As we will see later, this property can be used to greatly simplify the problem. A similar monotone property for simpler wire capacitance model and fixed-length segments has been proved in [13].

Lemma 1: The optimal wire sizing function f is a decreasing function.

Proof: Without loss of generality, we assume that the length of every step of f is greater than zero. Suppose f changes from a smaller value h to a larger value h' at a point of a distance l from the source. Therefore, $f(x)=h$ for $l-\delta\le x\le l$ and $f(x)=h'$ for $l<x\le l+\delta$ for some $\delta>0$. Let g be another wire sizing function defined as follows:

$$g(x)=\begin{cases}h', & l-\delta\le x\le l\\ h, & l<x\le l+\delta\\ f(x), & \text{otherwise}\end{cases}$$

Consider the Elmore delay corresponding to these two segments of length δ. Let $R_{\mathcal{U}}$ be the total driver and wire resistance before these two segments and $C_{\mathcal{D}}$ be the total wire and load capacitance after these two segments. The delays with respect to f and g are, respectively

$$\begin{aligned}D(f)&=R_{\mathcal{U}}(c(h)\delta+c(h')\delta+C_{\mathcal{D}})\\&\quad+\frac{r_0\delta}{h}\left(\frac{c(h)\delta}{2}+c(h')\delta+C_{\mathcal{D}}\right)+\frac{r_0\delta}{h'}\left(\frac{c(h')\delta}{2}+C_{\mathcal{D}}\right)\\D(g)&=R_{\mathcal{U}}(c(h')\delta+c(h)\delta+C_{\mathcal{D}})\\&\quad+\frac{r_0\delta}{h'}\left(\frac{c(h')\delta}{2}c(h)\delta+C_{\mathcal{D}}\right)+\frac{r_0\delta}{h}\left(\frac{c(h)\delta}{2}+C_{\mathcal{D}}\right).\end{aligned}$$

So $D(g)-D(f)=(r_0\delta^2c(h))/(h')-(r_0\delta^2c(h'))/(h)<0$ as $h<h'$ and $c(h)<c(h')$. In other words, g is better than f, which contradicts to the fact f is optimal. □

Instead of solving $\mathcal{WS}$ directly, we solve the problem $\mathcal{WS}'$, shown at the bottom of the next page. In $\mathcal{WS}'$, the wire is divided into n segments such that the width of the ith segment is h_i, and the length of each segment is to be determined. By Lemma 1, it is clear that the problem $\mathcal{WS}'$ is equivalent to $\mathcal{WS}$. Note that our new approach divides the wire into only n segments and gives an optimal solution to the original problem. If we approach $\mathcal{WS}$ by dividing the wire into small fixed-length segments, the solution will not be exact. In order to obtain a good approximation, the wire needs to be divided into much more than n segments.

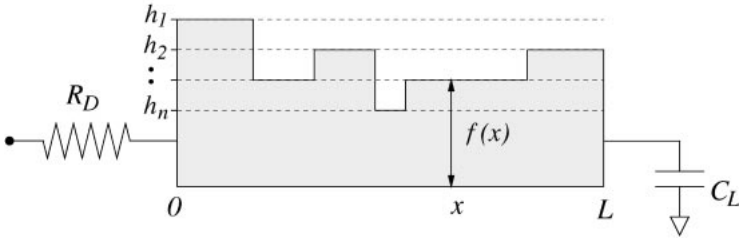
B. Problem Formulation

We show in this subsection that $\mathcal{WS}'$ can be formulated as a quadratic program. In addition, we prove that the quadratic program is convex.

PROBLEM $\mathcal{WS}$: The Wire Sizing Problem

Given: wire length L, driver resistance R_D, load capacitance C_L, a set $H=\{h_1,\cdots,h_n\}$ of n choices of wire width such that $h_1>\cdots>h_n$.

Determine: the wire width $f(x)$ at each point x along the wire such that the delay from source to sink is minimized.



Let $c_i = c(h_i)$ for $1 \le i \le n$. Then for $\mathcal{WS}'$, the delay from source to sink is

$$\begin{aligned} D &= R_D(c_1 l_1 + c_2 l_2 + \cdots + c_n l_n + C_L) \\ &\quad + \frac{r_0 l_1}{h_1}\left(\frac{c_1 l_1}{2} + c_2 l_2 + \cdots + c_n l_n + C_L\right) \\ &\quad + \frac{r_0 l_2}{h_2}\left(\frac{c_2 l_2}{2} + c_3 l_3 + \cdots + c_n l_n + C_L\right) \\ &\quad \vdots \\ &\quad + \frac{r_0 l_n}{h_n}\left(\frac{c_n l_n}{2} + C_L\right) \\ &= \frac{1}{2}\mathbf{l}^T \mathbf{\Phi} \mathbf{l} + \boldsymbol{\rho}^T \mathbf{l} + R_D C_L \end{aligned}$$

where

$$\mathbf{\Phi} = \begin{pmatrix} c_1 r_0/h_1 & c_2 r_0/h_1 & c_3 r_0/h_1 & \cdots & c_n r_0/h_1 \\ c_2 r_0/h_1 & c_2 r_0/h_2 & c_3 r_0/h_2 & \cdots & c_n r_0/h_2 \\ c_3 r_0/h_1 & c_3 r_0/h_2 & c_3 r_0/h_3 & \cdots & c_n r_0/h_3 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_n r_0/h_1 & c_n r_0/h_2 & c_n r_0/h_3 & \cdots & c_n r_0/h_n \end{pmatrix}$$

$$\boldsymbol{\rho} = \begin{pmatrix} R_D c_1 + C_L r_0/h_1 \\ R_D c_2 + C_L r_0/h_2 \\ R_D c_3 + C_L r_0/h_3 \\ \vdots \\ R_D c_n + C_L r_0/h_n \end{pmatrix} \quad \text{and} \quad \mathbf{l} = \begin{pmatrix} l_1 \\ l_2 \\ l_3 \\ \vdots \\ l_n \end{pmatrix}.$$

So $\mathcal{WS}'$ can be formulated as follows:

$$\begin{array}{lll} \mathcal{CQP}: & \text{minimize} & \frac{1}{2}\mathbf{l}^T \mathbf{\Phi} \mathbf{l} + \boldsymbol{\rho}^T \mathbf{l} \\ & \text{subject to} & l_1 + \cdots + l_n = L \\ & & l_i \ge 0 \quad \text{for } 1 \le i \le n. \end{array}$$

$\mathcal{CQP}$ is a quadratic program. In general, quadratic program is a mathematical program with a quadratic objective function subject to linear equality and inequality constraints. If the Hessian matrix $\mathbf{\Phi}$ is positive definite, it is called a convex quadratic program. Note that quadratic programming is NP-hard [18] but convex quadratic programming can be solved in polynomial time [21]. In the following, we prove $\mathbf{\Phi}$ in our case is positive definite, and hence the quadratic program $\mathcal{CQP}$ is convex. First, we make the following two definitions.

Definition 1 (Symmetric Decomposable Matrix): Let $\mathbf{Q} = (q_{ij})$ be an $n \times n$ symmetric matrix. If for some $\boldsymbol{\alpha} = (\alpha_1, \cdots, \alpha_n)^T$ and $\mathbf{v} = (v_1, \cdots, v_n)^T$ such that $0 < \alpha_1 < \cdots < \alpha_n$, $q_{ij} = q_{ji} = \alpha_i v_i v_j$ for $i \le j$, then $\mathbf{Q}$ is called a symmetric decomposable matrix. We denote $\mathbf{Q} = \mathit{SDM}(\boldsymbol{\alpha}, \mathbf{v})$.

Definition 2 (Upper Triangular Decomposable Matrix): Let $\mathbf{U} = (u_{ij})$ be an $n \times n$ upper triangular matrix. If for some $\boldsymbol{\beta} = (\beta_1, \cdots, \beta_n)^T$ and $\mathbf{v} = (v_1, \cdots, v_n)^T$ such that $\beta_i > 0$ for all i, $u_{ij} = \beta_i v_j$ for $i \le j$ and $u_{ij} = 0$ for $i > j$, then $\mathbf{U}$ is called an upper triangular decomposable matrix. We denote $\mathbf{U} = \mathit{UTDM}(\boldsymbol{\alpha}, \mathbf{v})$.

Lemma 2: $\mathbf{\Phi}$ in $\mathcal{CQP}$ is symmetric decomposable.

Proof: Let $\boldsymbol{\alpha} = (\frac{r_0}{c_1 h_1}, \cdots, \frac{r_0}{c_n h_n})^T$ and $\mathbf{v} = (c_1, \cdots, c_n)^T$. Note that $0 < \frac{r_0}{c_1 h_1} < \cdots < \frac{r_0}{c_n h_n}$. Then $\mathbf{\Phi} = \mathit{SDM}(\boldsymbol{\alpha}, \mathbf{v})$. □

Lemma 3: If $\mathbf{Q}$ is symmetric decomposable, then $\mathbf{Q} = \mathbf{U}^T\mathbf{U}$ where $\mathbf{U}$ is upper triangular decomposable. In particular, if $\mathbf{Q} = \mathit{SDM}(\boldsymbol{\alpha}, \mathbf{v})$, then $\mathbf{Q} = \mathbf{U}^T\mathbf{U}$ where $\mathbf{U} = \mathit{UTDM}(\boldsymbol{\beta}, \mathbf{v})$, $\beta_i = \sqrt{\alpha_i - \alpha_{i-1}}$ for $1 \le i \le n$.

Proof: Let $\mathbf{U}$ be the matrix as defined in the lemma. Note that $\beta_i > 0$ for all i as $0 < \alpha_1 < \cdots < \alpha_n$. For $i \le j$

$$\begin{aligned} \text{entry } (i,j) \text{ of } \mathbf{U}^T\mathbf{U} &= \sum_{k=1}^{i} (\beta_k v_j)(\beta_k v_i) \\ &= \left(\sum_{k=1}^{i} \beta_k^2\right) v_i v_j \\ &= \alpha_i v_i v_j \\ &= \text{entry } (i,j) \text{ of } \mathbf{Q} \end{aligned}$$

□

Lemma 4: If $\mathbf{Q}$ is symmetric decomposable, then $\mathbf{Q}$ is positive definite.

Proof: Let $\mathbf{Q} = \mathit{SDM}(\boldsymbol{\alpha}, \mathbf{v})$. By Lemma 3, $\mathbf{Q} = \mathbf{U}^T\mathbf{U}$ where $\mathbf{U} = \mathit{UTDM}(\boldsymbol{\beta}, \mathbf{v})$ for some $\boldsymbol{\beta}$. For any $\mathbf{x} \ne \mathbf{0}$, let $\mathbf{y} = \mathbf{U}\mathbf{x}$. Note that $\mathbf{y} \ne \mathbf{0}$ as $\mathbf{U}$ is nonsingular and $\mathbf{x} \ne \mathbf{0}$. So $\mathbf{x}^T\mathbf{Q}\mathbf{x} = \mathbf{x}^T\mathbf{U}^T\mathbf{U}\mathbf{x} = \mathbf{y}^T\mathbf{y} > 0$. In other words, $\mathbf{Q}$ is positive definite.

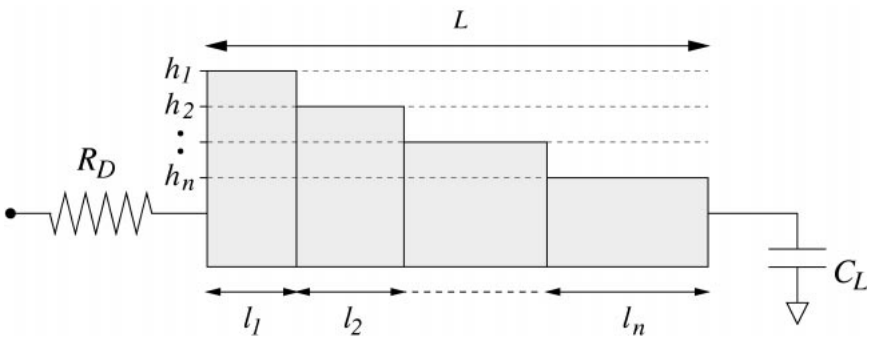
By Lemma 2, Lemma 4, and [21], we have Theorem 1.

Theorem 1: The quadratic program $\mathcal{CQP}$ is convex, and hence can be solved in polynomial time.

PROBLEM $\mathcal{WS}'$: The Simplified Wire Sizing Problem (equivalent to $\mathcal{WS}$)

Given: wire length L, driver resistance R_D, load capacitance C_L, a set $H = \{h_1, \cdots, h_n\}$ of n choices of wire width such that $h_1 > \cdots > h_n$.

Determine: the segment lengths $l_i \ge 0$ for $1 \le i \le n$ such that the delay from source to sink is minimized.



C. Solving Convex Quadratic Programs by Active Set Method

The active set method is a classical technique for constrained optimization problems. It has been shown to be efficient in practice. It is also one of the most popular methods to solve quadratic programs. In this subsection, we give a brief introduction to the active set method. In particular, we present how to solve general convex quadratic programs by the active set method. More details can be found in [24].

Assume without loss of generality that we are considering minimization problems. An inequality constraint $g(\mathbf{x}) \leq 0$ for some mathematical program is said to be *active* at a feasible point $\mathbf{x}$ if $g(\mathbf{x}) = 0$, and *inactive* at $\mathbf{x}$ if $g(\mathbf{x}) < 0$. Consider a general convex quadratic program, which can have both equality and inequality constraints. At the optimal solution, each inequality constraint is either active or inactive. If we solve the program by setting all those active inequality constraints as equalities and ignoring all those inactive inequality constraints, then the resulting program will have equality constraints only. Moreover, its solution will be the same as the optimal solution of the original program. As we will show shortly, equality constrained convex quadratic programs are easy to solve. Therefore, if we know which constraints are active at the optimal solution, then the optimal solution of the original program can be easily obtained by solving a convex quadratic program with equality constraints only. However, the set of active constraints at optimal solution is not known beforehand. Basically, the active set method is a systematic way to find the set of active constraints at the optimal solution.

The active set method works iteratively. In each iteration, the inequality constraints are partitioned into two sets: those that are to be treated as active and those that are to be treated as inactive. We call the set of inequality constraints treated as active the active set $\mathcal{A}$. Those in $\mathcal{A}$ are considered as equality constraints and those not in $\mathcal{A}$ are essentially ignored. The resulting equality constrained program is solved and the active set $\mathcal{A}$ is modified according to the solution obtained. If the current solution is infeasible with respect to the original program, some inequality constraints currently treated as inactive will be added to $\mathcal{A}$. If the current solution is feasible but some Lagrange multipliers corresponding to some constraints in $\mathcal{A}$ are negative (or positive for a maximization problem), then by the sensitivity interpretation of Lagrange multipliers, the objective value can be improved if we relax those constraints (i.e., the solution is not optimal). So in this case, some of those constraints will be removed from $\mathcal{A}$. Exactly which inequality constraints to add or to remove in each iteration depends on the problem and the design of the algorithm. See [24] for some commonly used strategies. The process is repeated until the optimal solution is found.

It is clear that a fundamental component of the active set method is to solve problems with equality constraints only. We present how to solve an equality constrained convex quadratic program below. Consider the following equality constrained convex quadratic program:

$$\mathcal{ECQP}: \quad \begin{array}{ll} \text{minimize} & \frac{1}{2}\mathbf{l}^T\mathbf{\Phi}\mathbf{l} + \boldsymbol{\rho}^T\mathbf{l} \\ \text{subject to} & \mathbf{\Gamma}\mathbf{l} = \mathbf{b} \end{array}$$

where $\mathbf{\Phi}$ is positive definite. It is reasonable to assume that $\mathbf{\Gamma}$ is of full rank, since all the $\mathbf{\Gamma}$ obtained from our formulation of the wire sizing problem and of the extensions in this paper are of full rank. Consider the associated Lagrangian

$$\ell(\mathbf{l}, \boldsymbol{\lambda}) = \frac{1}{2}\mathbf{l}^T\mathbf{\Phi}\mathbf{l} + \boldsymbol{\rho}^T\mathbf{l} + \boldsymbol{\lambda}^T(\mathbf{\Gamma}\mathbf{l} - \mathbf{b}).$$

The Lagrange necessary optimality conditions are $(\partial\ell(\mathbf{l}, \boldsymbol{\lambda}))/(\partial l_i) = 0$ and $(\partial\ell(\mathbf{l}, \boldsymbol{\lambda}))/(\partial\lambda_i) = 0$ for all i. The conditions can be written in matrix form as follows:

$$\mathbf{\Phi}\mathbf{l} + \mathbf{\Gamma}^T\boldsymbol{\lambda} + \boldsymbol{\rho} = \mathbf{0} \tag{1}$$

$$\mathbf{\Gamma}\mathbf{l} - \mathbf{b} = \mathbf{0}. \tag{2}$$

Multiplying (1) by $\mathbf{\Gamma}\mathbf{\Phi}^{-1}$, we have $\mathbf{\Gamma}\mathbf{l} + \mathbf{\Gamma}\mathbf{\Phi}^{-1}\mathbf{\Gamma}^T\boldsymbol{\lambda} + \mathbf{\Gamma}\mathbf{\Phi}^{-1}\boldsymbol{\rho} = \mathbf{0}$, which can be rewritten as $\mathbf{b} + (\mathbf{\Gamma}\mathbf{\Phi}^{-1}\mathbf{\Gamma}^T)\boldsymbol{\lambda} + \mathbf{\Gamma}\mathbf{\Phi}^{-1}\boldsymbol{\rho} = \mathbf{0}$ as $\mathbf{\Gamma}\mathbf{l} = \mathbf{b}$ by (2). Since $\mathbf{\Phi}$ is positive definite and $\mathbf{\Gamma}$ is of full rank, $\mathbf{\Gamma}\mathbf{\Phi}^{-1}\mathbf{\Gamma}^T$ is invertible. Hence

$$\boldsymbol{\lambda} = -\left(\mathbf{\Gamma}\mathbf{\Phi}^{-1}\mathbf{\Gamma}^T\right)^{-1}(\mathbf{\Gamma}\mathbf{\Phi}^{-1}\boldsymbol{\rho} + \mathbf{b}). \tag{3}$$

Also, by rearranging (1), we have

$$\mathbf{l} = -\mathbf{\Phi}^{-1}\mathbf{\Gamma}^T\boldsymbol{\lambda} - \mathbf{\Phi}^{-1}\boldsymbol{\rho}. \tag{4}$$

In other words, the solution to the Lagrange necessary optimality conditions is uniquely given by (3) and (4). Since $\mathcal{ECQP}$ is convex, the Lagrange necessary optimality conditions should also be sufficient. So (3) and (4) also give the optimal solution of $\mathcal{ECQP}$.

Therefore, if we apply the active set method to the convex quadratic program $\mathcal{CQP}$, we need to solve an equality constrained problem in the form of $\mathcal{ECQP}$, which is equivalent to computing (3) and (4), in each iteration. So each iteration can be done in cubic time in general. However, we show below how $\mathcal{ECQP}$ can be solved in linear time in our case.

D. Our Algorithm MASM

In this subsection, we show how to solve $\mathcal{ECQP}$ in linear time. Then we present a very efficient algorithm MASM for the convex quadratic program $\mathcal{CQP}$ based on the active set method.

The technique enabling us to solve $\mathcal{ECQP}$ in linear time is based on two observations. The first observation is that all the inequality constraints in $\mathcal{CQP}$ are of the simple form $l_i \geq 0$. If we apply the idea of the active set method presented in Section II-C directly, then in each iteration, we have an equality constrained program $\mathcal{ECQP}$ obtained by treating all inequality constraints in the active set $\mathcal{A}$ as equalities. As $\mathcal{A}$ may contain a lot of constraints, $\mathbf{\Gamma}$ may have many rows. So instead of solving (3) and (4) directly, which may be slow, we set $l_i = 0$ for all i such that $l_i \geq 0$ is in $\mathcal{A}$, and we substitute them into $\mathcal{ECQP}$. The resulting program is of exactly the same form as $\mathcal{ECQP}$ but has only one equality constraint (total length constraint) and has less variables. In particular, if for some iteration, all constraints except $l_{j_1} \geq 0, \cdots, l_{j_r} \geq 0$ are in the active set $\mathcal{A}$, then the program $\mathcal{ECQP}$ corresponding to that iteration will be equivalent to the following reduced equality constrained convex quadratic program

$$\mathcal{RCQP}: \quad \begin{array}{ll} \text{minimize} & \frac{1}{2}\mathbf{l}_{\mathcal{A}}^T\mathbf{\Phi}_{\mathcal{A}}\mathbf{l}_{\mathcal{A}} + \boldsymbol{\rho}_{\mathcal{A}}^T\mathbf{l}_{\mathcal{A}} \\ \text{subject to} & \mathbf{\Gamma}_{\mathcal{A}}\mathbf{l}_{\mathcal{A}} = L \end{array}$$

where $\mathbf{l}_{\mathcal{A}} = (l_{j_1}, \cdots, l_{j_r})^T$, $\mathbf{\Gamma}_{\mathcal{A}} = (1 \;\; 1 \;\; \cdots \;\; 1)$, $\boldsymbol{\rho}_{\mathcal{A}} = (R_D c_{j_1} + C_L r_0/h_{j_1}, \cdots, R_D c_{j_r} + C_L r_0/h_{j_r})^T$,

and $\mathbf{\Phi}_{\mathcal{A}}$ is the symmetric decomposable matrix corresponding to $\mathcal{A}$ (i.e., $\mathbf{\Phi}_{\mathcal{A}} = SDM(\boldsymbol{\alpha}_{\mathcal{A}}, \mathbf{v}_{\mathcal{A}})$ with $\boldsymbol{\alpha}_{\mathcal{A}} = ((r_0)/(c_{j_1}h_{j_1}), \cdots, (r_0)/(c_{j_r}h_{j_r}))^T$ and $\mathbf{v}_{\mathcal{A}} = (c_{j_1}, \cdots, c_{j_r})^T$). As before, the solution to the Lagrange necessary optimality conditions for $\mathcal{RCQP}$ is

$$\lambda_{\mathcal{A}} = -\left(\mathbf{\Gamma}_{\mathcal{A}}\mathbf{\Phi}_{\mathcal{A}}^{-1}\mathbf{\Gamma}_{\mathcal{A}}^T\right)^{-1}\left(\mathbf{\Gamma}_{\mathcal{A}}\mathbf{\Phi}_{\mathcal{A}}^{-1}\boldsymbol{\rho}_{\mathcal{A}} + L\right)$$
$$\mathrm{l}_{\mathcal{A}} = -\mathbf{\Phi}_{\mathcal{A}}^{-1}\mathbf{\Gamma}_{\mathcal{A}}^T\lambda_{\mathcal{A}} - \mathbf{\Phi}_{\mathcal{A}}^{-1}\boldsymbol{\rho}_{\mathcal{A}}$$

The second observation is that all symmetric decomposable matrices are tridiagonal. Hence, $\mathbf{\Phi}_{\mathcal{A}}^{-1}$ is tridiagonal. The following 2 lemmas prove this interesting observation.

Lemma 5: If $\mathbf{U}$ is upper triangular decomposable, then $\mathbf{U}^{-1}$ is bidiagonal. In particular, if $\mathbf{U} = UTDM(\boldsymbol{\beta}, \mathbf{v})$, then $\mathbf{U}^{-1} = (\omega_{ij})$ where $\omega_{ii} = \frac{1}{\beta_i v_i}$ for $1 \le i \le n$, $\omega_{i,i+1} = (-1)/(\beta_{i+1}v_i)$ for $1 \le i \le n-1$ and $\omega_{ij} = 0$, otherwise.

Proof: Let $\mathbf{W} = (\omega_{ij})$ where ω_{ij} is defined as in the lemma.

Case 1) $1 \le i = j \le n-1$**:**

$$\text{entry } (i,j) \text{ of } \mathbf{WU} = \frac{1}{\beta_i v_i}\cdot\beta_i v_i + \frac{1}{\beta_{i+1}v_i}\cdot 0 = 1.$$

Case 2) $i = j = n$**:**

$$\text{entry } (i,j) \text{ of } \mathbf{WU} = \frac{1}{\beta_n v_n}\cdot\beta_n v_n = 1.$$

Case 3) $1 \le i < j \le n$**:**

$$\text{entry } (i,j) \text{ of } \mathbf{WU} = \frac{1}{\beta_i v_i}\cdot\beta_i v_j - \frac{1}{\beta_{i+1}v_i}\cdot\beta_{i+1}v_j = 0.$$

Case 4) $1 \le j < i \le n$**:**

$$\text{entry } (i,j) \text{ of } \mathbf{WU} = 0, \text{ obviously.}$$

So $\mathbf{W} = \mathbf{U}^{-1}$. □

Lemma 6: If $\mathbf{Q}$ is symmetric decomposable, then $\mathbf{Q}^{-1}$ is tridiagonal. In particular, if $\mathbf{Q} = SDM(\boldsymbol{\alpha}, \mathbf{v})$, then $\mathbf{Q}^{-1} = (\theta_{ij})$, where $\theta_{ii} = \frac{1}{(\alpha_i - \alpha_{i-1})v_i^2} + \frac{1}{(\alpha_{i+1} - \alpha_i)v_i^2}$, $\theta_{i,i+1} = \theta_{i+1,i} = \frac{-1}{(\alpha_{i+1} - \alpha_i)v_i v_{i+1}}$ for $1 \le i \le n-1$, $\theta_{nn} = \frac{1}{(\alpha_n - \alpha_{n-1})v_n^2}$, and $\theta_{ij} = 0$ otherwise.

Proof: By Lemma 3 and Lemma 5, $\mathbf{Q}^{-1} = \mathbf{U}^{-1}(\mathbf{U}^{-1})^T$ such that $\mathbf{U}^{-1} = (\omega_{ij})$, where $\omega_{ii} = \frac{1}{\beta_i v_i}$ for $1 \le i \le n$, $\omega_{i,i+1} = \frac{-1}{\beta_{i+1}v_i}$ for $1 \le i \le n-1$, $\omega_{ij} = 0$, otherwise, and $\beta_i = \sqrt{\alpha_i - \alpha_{i-1}}$ for $1 \le i \le n$. Then

Case 1) $1 \le i = j \le n-1$**:**

$$\theta_{ij} = \left(\frac{1}{\beta_i v_i}\right)^2 + \left(\frac{-1}{\beta_{i+1}v_i}\right)^2 = \frac{1}{(\alpha_i - \alpha_{i-1})v_i^2} + \frac{1}{(\alpha_{i+1} - \alpha_i)v_i^2}.$$

Case 2) $1 \le i \le n-1, j = i+1$**:**

$$\theta_{ij} = \theta_{ji} = \frac{-1}{\beta_{i+1}v_i}\cdot\frac{1}{\beta_{i+1}v_{i+1}} = \frac{-1}{(\alpha_{i+1} - \alpha_i)v_i v_{i+1}}.$$

Case 3) $i = j = n$**:**

$$\theta_{ij} = \left(\frac{1}{\beta_n v_n}\right)^2 = \frac{1}{(\alpha_n - \alpha_{n-1})v_n^2}.$$

Case 4) Otherwise**:**

$$\theta_{ij} = \theta_{ji} = 0, \text{ obviously.}$$

By Lemma 6, we have Theorem 2. □

Theorem 2: $\mathbf{\Phi}_{\mathcal{A}}^{-1}$ in $\mathcal{RCQP}$ is tridiagonal.

Let $\mathbf{\Phi}_{\mathcal{A}}^{-1} = (\theta_{ij})$ where θ_{ij} is given by Lemma 6 and let $\boldsymbol{\rho}_{\mathcal{A}} = (\rho_1, \cdots, \rho_r)^T$. Because of the simple structure of the matrix $\mathbf{\Phi}_{\mathcal{A}}^{-1}$ and $\mathbf{\Gamma}_{\mathcal{A}}$ in $\mathcal{RCQP}$, the solution to the Lagrange optimality conditions for $\mathcal{RCQP}$ can be written in closed form as follows:

$$\lambda_{\mathcal{A}} = -\frac{L + \sum_{i=1}^{r}(\theta_{i-1,i} + \theta_{ii} + \theta_{i+1,i})\rho_i}{\sum_{i=1}^{r}(\theta_{i-1,i} + \theta_{ii} + \theta_{i+1,i})}$$
$$l_{j_i} = -(\theta_{i-1,i} + \theta_{ii} + \theta_{i+1,i})\lambda_{\mathcal{A}} - (\theta_{i-1,i}\rho_{i-1} + \theta_{ii}\rho_i + \theta_{i+1,i}\rho_{i+1}) \quad \text{for } 1 \le i \le r.$$

With the solution of $\mathcal{RCQP}$, we show how to find the solution l and $\boldsymbol{\lambda}$ of $\mathcal{ECQP}$ below. Obviously, $l_j = 0$ for all $j \notin \{j_1, \cdots, j_r\}$. So l can be found in linear time. Once l is found, $\boldsymbol{\lambda}$ can be found as follows. Let $\mathbf{x} = \mathbf{\Phi}^{-1}\mathbf{\Gamma}^T\boldsymbol{\lambda}$. By (4), $\mathbf{x} = -\mathbf{l} - \mathbf{\Phi}^{-1}\boldsymbol{\rho}$ and so $\mathbf{x}$ can be computed in linear time. Let $\mathbf{\Gamma}'$ be the submatrix of $\mathbf{\Gamma}$, such that $\mathbf{\Gamma}^T = (\mathbf{\Gamma}_{\mathcal{A}}{}^T\mathbf{\Gamma}'^T)$. Therefore, $\mathbf{\Gamma}'$ consists of all the rows corresponding to the nonnegative constraints in the active set. Let $\boldsymbol{\lambda}'$ be the subvector of $\boldsymbol{\lambda}$, such that $\boldsymbol{\lambda}^T = (\lambda_{\mathcal{A}}\boldsymbol{\lambda}'^T)$. Therefore, $\boldsymbol{\lambda}'$ consists of all Lagrange multipliers corresponding to the nonnegative constraints in the active set. Then $\mathbf{\Gamma}^T\boldsymbol{\lambda} = \mathbf{\Gamma}_{\mathcal{A}}^T\lambda_{\mathcal{A}} + \mathbf{\Gamma}'^T\boldsymbol{\lambda}'$. So $\mathbf{x} = \mathbf{\Phi}^{-1}\mathbf{\Gamma}^T\boldsymbol{\lambda} = \mathbf{\Phi}^{-1}(\mathbf{\Gamma}_{\mathcal{A}}^T\lambda_{\mathcal{A}} + \mathbf{\Gamma}'^T\boldsymbol{\lambda}')$. Hence, $\mathbf{\Phi}^{-1}\mathbf{\Gamma}'^T\boldsymbol{\lambda}' = \mathbf{x} - \mathbf{\Phi}^{-1}\mathbf{\Gamma}_{\mathcal{A}}^T\lambda_{\mathcal{A}}$. Note that the right-hand side can be computed in linear time. Then it is not difficult to see that $\boldsymbol{\lambda}'$ (i.e., $\boldsymbol{\lambda}$) can be solved in linear time. As a result, the program $\mathcal{ECQP}$ can also be solved in linear time.

We now state our strategy to add constraints to the active set and to remove constraints from the active set at each iteration. In our implementation, we add all the inactive constraints corresponding to negative segment lengths, and we remove all the active constraints corresponding to negative Lagrange multipliers. We observe that this simple strategy works well in practice. The algorithm can be summarized as

Algorithm MASM (Modified active set method):

1. Set the active set $\mathcal{A} = \emptyset$.
2. **repeat**
3. Solve for $\boldsymbol{\lambda}$ and l with respect to $\mathcal{A}$ by our special technique.
4. **if** $(\mathrm{l} \not\ge \mathbf{0})$ **then** /* check for feasibility */
5. Add all the constraints $l_j \ge 0$ to $\mathcal{A}$, for all j such that $l_j < 0$.
6. **else if** $(\boldsymbol{\lambda} \not\ge \mathbf{0})$ **then** /* check for optimality */
7. Remove all the constraints $l_j \ge 0$ from $\mathcal{A}$, for all j such that $\lambda_j < 0$.
8. **until** ($\mathrm{l} \ge \mathbf{0}$ and $\boldsymbol{\lambda} \ge \mathbf{0}$)

Theorem 3: The wire sizing problem $\mathcal{WS}$ (or equivalently, $\mathcal{WS}'$) can be solved by the algorithm MASM such that each iteration takes $O(n)$ time.

We show in Section VI that when our algorithm MASM is applied to wire sizing, the number of iterations of MASM is less than n and hence the total runtime is $O(n^2)$ in practice.

III. Simultaneous Buffer Insertion and Wire Sizing

In this section, we show that the simultaneous buffer insertion and wire sizing problem $\mathcal{BIWS}$ introduced in Section I can also be solved by the algorithm MASM. For the problem

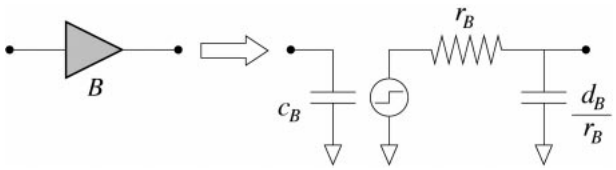


Fig. 2. The model of a buffer of size B by a switch-level RC circuit. c_B, r_B, and d_B are the input capacitance, output resistance, and the intrinsic delay of the buffer, respectively.

$\mathcal{BIWS}$, m buffers of sizes $B_1, \cdots, B_m$ are given and they are inserted into a wire in this order (with the buffer of size B_1 nearest to the source). A buffer is modeled as a switch-level RC circuit as shown in Fig. 2. For convenience, we treat the driver and the load as buffers. Assume without loss of generality that the driver is a buffer of size B_0 and the load is a buffer of size B_{m+1}.

Note that for $\mathcal{BIWS}$, the wire width is not necessary to be decreasing across a buffer. However, the sizing problem of the piece of wire between any two consecutive buffers is basically the same as $\mathcal{WS}$ discussed in Section II (except that the length of that piece of wire is not fixed). So by Lemma 1, the optimal wire sizing function between two adjacent buffers is still a decreasing step function. Hence, we can approach the problem as before by dividing the piece of wire between every pair of consecutive buffers into n segments of decreasing width, and determining the length of each segment. Instead of having a total length constraint for each piece of wire between buffers, we have a single constraint specifying that the sum of all the segment lengths equals L. The simplified problem $\mathcal{BIWS}'$ is shown at the bottom of the page.

For the delay from the kth buffer to the $(k+1)$th buffer, let $\mathbf{\Phi}$ be the matrix corresponding to the coefficients of the quadratic terms (i.e., the Hessian matrix as defined in Section II-B) and $\boldsymbol{\rho_k}$ be the vector corresponding to the coefficients of the linear terms. Let

$$\mathbf{\Psi} = \begin{pmatrix} \mathbf{\Phi} & & & 0 \\ & \mathbf{\Phi} & & \\ & & \ddots & \\ 0 & & & \mathbf{\Phi} \end{pmatrix} \quad \text{and} \quad \boldsymbol{\varrho} = \begin{pmatrix} \boldsymbol{\rho_0} \\ \boldsymbol{\rho_1} \\ \vdots \\ \boldsymbol{\rho_m} \end{pmatrix}.$$

Then the delay from source to sink is

$$D = \frac{1}{2}\mathbf{l}^T\mathbf{\Psi}\mathbf{l} + \boldsymbol{\varrho}^T\mathbf{l} + \sum_{k=0}^{m} r_{B_k} c_{B_{k+1}} + \sum_{k=1}^{m} d_{B_k}. \qquad (5)$$

So $\mathcal{BIWS}'$ can be formulated as follows:

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}\mathbf{l}^T\mathbf{\Psi}\mathbf{l} + \boldsymbol{\varrho}^T\mathbf{l} \\ \text{subject to} & l_1 + \cdots + l_{(m+1)n} = L \\ & l_i \geq 0 \quad \text{for } 1 \leq i \leq (m+1)n \end{array}$$

which is of the same form as $\mathcal{CQP}$. $\mathbf{\Psi}$ is clearly positive definite as $\mathbf{\Phi}$ is positive definite. Hence, the quadratic program is again convex. In addition, for each iteration, we can find $\mathbf{l}$ and $\boldsymbol{\lambda}$ as before by reducing the equality constrained program as in $\mathcal{ECQP}$ to one as in $\mathcal{RCQP}$. The reduced matrix $\mathbf{\Psi}_{\mathcal{A}}$ corresponding to $\mathcal{A}$ is

$$\mathbf{\Psi}_{\mathcal{A}} = \begin{pmatrix} \mathbf{\Phi}_{\mathcal{A}_0} & & & 0 \\ & \mathbf{\Phi}_{\mathcal{A}_1} & & \\ & & \ddots & \\ 0 & & & \mathbf{\Phi}_{\mathcal{A}_m} \end{pmatrix}$$

where $\mathbf{\Phi}_{\mathcal{A}_k}$ is the reduced symmetric decomposable matrix corresponding to the set $\mathcal{A}_\mathbf{k}$ of active constraints for segments between the kth buffer and the $(k+1)$th buffer. So

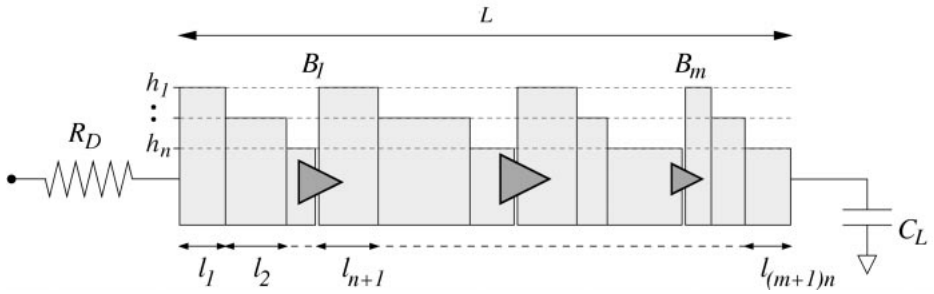
$$\mathbf{\Psi}_{\mathcal{A}}^{-1} = \begin{pmatrix} \mathbf{\Phi}_{\mathcal{A}_0}^{-1} & & & 0 \\ & \mathbf{\Phi}_{\mathcal{A}_1}^{-1} & & \\ & & \ddots & \\ 0 & & & \mathbf{\Phi}_{\mathcal{A}_m}^{-1} \end{pmatrix}.$$

Therefore $\mathbf{\Psi}_{\mathcal{A}}^{-1}$ is also tridiagonal as $\mathbf{\Phi}_{\mathcal{A}_0}^{-1}, \mathbf{\Phi}_{\mathcal{A}_1}^{-1}, \cdots, \mathbf{\Phi}_{\mathcal{A}_m}^{-1}$ are all tridiagonal. Hence, $\mathcal{BIWS}'$ can be solved as before.

Theorem 4: The simultaneous buffer insertion and wire sizing problem $\mathcal{BIWS}$ (or equivalently, $\mathcal{BIWS}'$) can be solved by the algorithm MASM such that each iteration takes $O(mn)$ time.

We show in Section VI that when our algorithm MASM is applied to simultaneous buffer insertion and wire sizing, the number of iterations is about n and hence the total runtime is $O(mn^2)$ in practice.

PROBLEM $\mathcal{BIWS}'$: The Simplified Simultaneous Buffer Insertion and Wire Sizing Problem (equivalent to $\mathcal{BIWS}$)
Given: wire length L, driver resistance R_D, load capacitance C_L, a set $H = \{h_1, \cdots, h_n\}$ of n choices of wire width such that $h_1 > \cdots > h_n$, and m buffers of sizes $B_1, \cdots, B_m$.
Determine: the segment lengths $l_i \geq 0$ for $1 \leq i \leq (m+1)n$ such that the delay from source to sink is minimized.



IV. Simultaneous Buffer Insertion, Buffer Sizing, and Wire Sizing

In this section, we present an algorithm MASM-BS (Modified Active Set Method with Buffer Sizing consideration) for the simultaneous buffer insertion, buffer sizing, and wire sizing problem $\mathcal{BIBSWS}$, shown at the bottom of the page.

For the problem $\mathcal{BIWS}$ considered in Section III, it is assumed that the number of buffers to be inserted and the sizes of those buffers are given. But in practice, usually a library of buffers of several different sizes is given instead. The problem is to determine the optimal number of buffers and the size of each buffer used, as well as the position of each buffer and the width of the wire at each point. This more general problem can be solved by trying all possible combinations of number of buffers and buffer sizes. The problem instance corresponding to a particular combination (i.e., the number of buffers and the buffer sizes are fixed) is of the form of $\mathcal{BIWS}$ and so can be solved by our algorithm MASM.

Suppose the size of the buffer library is q, the maximum number of buffers allowed for a single wire is M (excluding the driver and the load), and the number of choices of wire width is n. When m buffers are inserted, there are q^m choices of buffer sizes. For each choice, as shown in Section VI, the runtime of MASM to solve the corresponding $\mathcal{BIWS}$ problem instance is $O(mn^2)$ in practice. Since the range of m is from 0 to M, the total runtime to solve $\mathcal{BIBSWS}$ is $\sum_{m=0}^{M} q^m O(mn^2) = O(q^M M n^2)$. As q, M, and n are usually small numbers in practice, this simple approach to handle buffer sizing should work well. Nevertheless, we present an effective technique below which can prune a lot of suboptimal combinations.

The basic idea of the pruning technique is that for each combination, we derive a lower bound on the delay of the corresponding $\mathcal{BIWS}$ problem instance. If the lower bound is already larger than the minimum delay obtained up to the previous combination (which is an upper bound on the optimal delay), then we know this combination will not give a delay better than the delay of the current best solution and can be pruned. The lower bound is given in the following lemma.

Lemma 7: Consider a combination $B_0, B_1, \cdots, B_m, B_{m+1}$ of buffer sizes. (B_0 and B_{m+1} are the sizes of the driver and the load, respectively. Therefore, m buffers are inserted.) The delay of the corresponding $\mathcal{BIWS}$ problem instance

$$D \geq (m+1)D^* + \varrho_{\min}(B_0, \cdots, B_{m+1})L + \sum_{k=0}^{m} r_{B_k} c_{B_{k+1}} + \sum_{k=1}^{m} d_{B_k}$$

where $D^* = \min\{\frac{1}{2}\mathbf{l}^T\mathbf{\Phi}\mathbf{l} : \sum_{i=1}^{n} l_i = L/(m+1)\}$, $\mathbf{\Phi}$ is the Hessian matrix corresponding to the wire sizing problem between any two consecutive buffers, and $\varrho_{\min}(B_0, \cdots, B_{m+1})$ is the minimum over all the entries in the vector $\boldsymbol{\varrho}$ of the corresponding $\mathcal{BIWS}$ problem instance.

Proof: Given a combination $B_0, \cdots, B_{m+1}$, the delay D of the corresponding $\mathcal{BIWS}$ problem is given by (5). So

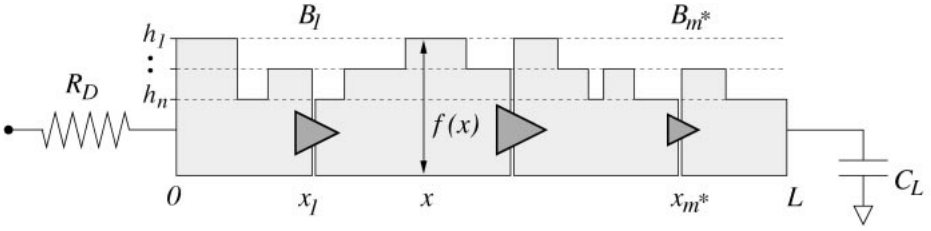
$$\begin{aligned}
D &\geq \min\left\{ \frac{1}{2}\mathbf{l}^T\mathbf{\Psi}\mathbf{l} + \boldsymbol{\varrho}^T\mathbf{l} + \sum_{k=0}^{m} r_{B_k} c_{B_{k+1}} + \sum_{k=1}^{m} d_{B_k} : \sum_{i=1}^{(m+1)n} l_i = L \text{ and } l_i \geq 0 \ \forall i \right\} \\
&\geq \min\left\{ \frac{1}{2}\mathbf{l}^T\mathbf{\Psi}\mathbf{l} : \sum_{i=1}^{(m+1)n} l_i = L \right\} + \min\left\{ \boldsymbol{\varrho}^T\mathbf{l} : \sum_{i=1}^{(m+1)n} l_i = L \text{ and } l_i \geq 0 \ \forall i \right\} \\
&\quad + \sum_{k=0}^{m} r_{B_k} c_{B_{k+1}} + \sum_{k=1}^{m} d_{B_k} \\
&= (m+1)\min\left\{ \frac{1}{2}\mathbf{l}^T\mathbf{\Phi}\mathbf{l} : \sum_{i=1}^{n} l_i = \frac{L}{m+1} \right\} + \varrho_{\min}(B_0, \cdots, B_{m+1})L + \sum_{k=0}^{m} r_{B_k} c_{B_{k+1}} + \sum_{k=1}^{m} d_{B_k} \\
&= (m+1)D^* + \varrho_{\min}(B_0, \cdots, B_{m+1})L + \sum_{k=0}^{m} r_{B_k} c_{B_{k+1}} + \sum_{k=1}^{m} d_{B_k}.
\end{aligned}$$

□

PROBLEM $\mathcal{BIBSWS}$: The Simultaneous Buffer Insertion, Buffer Sizing and Wire Sizing Problem

Given: wire length L, driver resistance R_D, load capacitance C_L, a set $H = \{h_1, \cdots, h_n\}$ of n choices of wire width such that $h_1 > \cdots > h_n$, a set S of q choices of buffer size, and an upper bound M on the number of buffers inserted.

Determine: the optimal number of buffers m^*, the sizes $B_1, \cdots, B_{m^*}$ of the m^* buffers, the positions $x_1, \cdots, x_{m^*}$ at which the buffers are inserted and the wire width $f(x)$ at each point x along the wire such that the delay from source to sink is minimized.



This lower bound is simple and easy to calculate. It can also be shown experimentally to be very effective in pruning. However, for future technologies, the value of M will probably be set to larger values (around 5–10). The time to calculate the lower bounds for all the $\sum_{m=0}^{M} q^m$ combinations will then become dominating. In the following, we improve the idea above so that instead of pruning a single combination each time, an entire subset of the possible combinations can be pruned.

Suppose m buffers are inserted and the sizes of the first i buffers inserted (i.e., $B_1, \cdots, B_i$) are fixed. Then there are q^{m-i} combinations corresponding to the different choices for the sizes of the remaining $m-i$ buffers. We call the minimum over all the lower bounds for those q^{m-i} combinations an aggregate lower bound. If the aggregate lower bound is larger than the minimum delay obtained so far, then all the q^{m-i} combinations can be pruned. Otherwise, if i equals m, the sizes of all m buffers are fixed. We call MASM to solve the combination $B_0, B_1, \cdots, B_m, B_{m+1}$. If i is less than m, we fix the size of one more buffer (i.e., B_{i+1}) and try pruning again.

For $0 \le i \le M$ and $b_0, \cdots, b_{i+1} \in S \cup \{B_0, B_{m+1}\}$, let

$$t(b_0, \cdots, b_{i+1}) = (i+1)D^* + \varrho_{\min}(b_0, \cdots, b_{i+1})L + \sum_{k=0}^{i} r_{b_k} c_{b_{k+1}} + \sum_{k=1}^{i} d_{b_k}$$

$$\hat{t}(b_0, \cdots, b_{i+1}) = (i+1)D^* + \sum_{k=0}^{i} r_{b_k} c_{b_{k+1}} + \sum_{k=1}^{i} d_{b_k}$$

For $0 \le k \le M$ and $b, b' \in S \cup \{B_0, B_{m+1}\}$, let

$$T_k(b, b') = \min\{\, t(b, b_1, \cdots, b_k, b') : b_1, \cdots, b_k \in S\}$$
$$\hat{T}_k(b, b') = \min\{\, \hat{t}(b, b_1, \cdots, b_k, b') : b_1, \cdots, b_k \in S\}.$$

It is not difficult to see that for the combinations with fixed $B_1, \cdots, B_i$, the aggregate lower bound is given by either $t(B_0, \cdots, B_i) + d_{B_i} + \hat{T}_{m-i}(B_i, B_{m+1})$ or $\hat{t}(B_0, \cdots, B_i) + d_{B_i} + T_{m-i}(B_i, B_{m+1})$, depending on whether the minimum entry in $\boldsymbol{\varrho}$ is before or after the ith buffer.

$t(\,)$, $\hat{t}(\,)$, $T_k(\,)$ and $\hat{T}_k(\,)$ (and hence the aggregate lower bound) can be calculated efficiently as follows. For $T_k(\,)$ and $\hat{T}_k(\,)$, instead of enumerating all the q^k combinations, they can be precomputed by the dynamic programming technique.

For $0 \le k \le M$ and $b, b' \in S \cup \{B_0, B_{m+1}\}$

$$T_k(b, b') = \begin{cases} t(b, b'), & \text{if } k = 0 \\ \min_{b'' \in S}\{T_{k-1}(b, b'') + d_{b''} + \hat{T}_0(b'', b'), \\ \quad \hat{T}_{k-1}(b, b'') + d_{b''} + T_0(b'', b')\}, & \text{if } k > 0 \end{cases}$$

$$\hat{T}_k(b, b') = \begin{cases} \hat{t}(b, b'), & \text{if } k = 0 \\ \hat{T}_{k-1}(b, b'') + d_{b''} + \hat{T}_0(b'', b'), & \text{if } k > 0 \end{cases}$$

$t(\,)$ and $\hat{t}(\,)$ can be calculated incrementally. For $0 \le i \le M$ and $b_0, \cdots, b_{i+1} \in S \cup \{B_0, B_{m+1}\}$

$$t(b_0, \cdots, b_{i+1}) = \min\{t(b_0, \cdots, b_i) + d_{b_i} + \hat{t}(b_i, b_{i+1}), \hat{t}(b_0, \cdots, b_i) + d_{b_i} + t(b_i, b_{i+1})\}$$
$$\hat{t}(b_0, \cdots, b_{i+1}) = \hat{t}(b_0, \cdots, b_i) + d_{b_i} + \hat{t}(b_i, b_{i+1})$$

Note that only those $t(\,)$ and $\hat{t}(\,)$ associated with combinations which are not pruned need to be calculated.

The algorithm MASM-BS can be summarized as

Algorithm MASM-BS (Modified Active Set Method with Buffer Sizing consideration)

1. precompute $t(b, b')$, $\hat{t}(b, b')$, $T_k(b, b')$ and $\hat{T}_k(b, b')$ for all k, b and b'.
2. **for** $m := 0$ **to** M **do**
3. Call PRUNE$(m, 0, (B_0, B_{m+1}), \infty, -d_{B_0})$.

Procedure PRUNE$(\mathbf{m}, \mathbf{i}, (\mathbf{B_0}, \cdots, \mathbf{B_i}, \mathbf{B_{m+1}}), \mathbf{t}, \hat{\mathbf{t}})$

/* $t = t(B_0, \cdots, B_i), \hat{t} = \hat{t}(B_0, \cdots, B_i)$ */

4. $LB := \min\{t + d_{B_i} + \hat{T}_{m-i}(B_i, B_{m+1}), \hat{t} + d_{B_i} + T_{m-i}(B_i, B_{m+1})\}$
5. **if** ($LB <$ minimum delay so far) **then**
6. **if** $(i = m)$ **then**
7. Call MASM to solve the combination $B_0, \cdots, B_{m+1}$.
8. **else**
9. **for** $B_{i+1} \in S$ **do**
10. $\tau := \min\{t + d_{B_i} + \hat{t}(B_i, B_{i+1}), \hat{t} + d_{B_i} + t(B_i, B_{i+1})\}$
11. $\hat{\tau} := \hat{t} + d_{B_i} + \hat{t}(B_i, B_{i+1})$
12. Call PRUNE$(m, i+1, (B_0, \cdots, B_i, B_{i+1}, B_{m+1}), \tau, \hat{\tau})$.

D^* can be computed in $O(n)$ time by the technique for solving $\mathcal{ECQP}$ in Section II-C. Then it is clear that $t(b, b')$ and $\hat{t}(b, b')$ for all b and b' can be computed in $O(q^2 n)$ time, and $T_k(b, b')$ and $\hat{T}_k(b, b')$ for all k, b and b' can be computed in $O(Mq^3)$ time. So the total time for the precomputation step is $O(q^2(n + Mq))$. Once the precomputation is done, each aggregate lower bound can be calculated in constant time. In Section VI, we show experimentally that our pruning technique is very effective. Far less than $O(q^M)$ aggregate lower bounds need to be calculated, and even less combinations need to be solved by MASM. As a result, the algorithm MASM-BS is very efficient in practice.

V. Extensions

In the following two subsections, we extend our result for simultaneous buffer insertion and wire sizing to consider wire area (and, hence, power dissipation), and to handle additional constraints to the solution respectively. We show that for both extensions, the resulting problem can still be solved by MASM such that each iteration can be done in linear time. Besides, it is easy to see that we can also handle a combination of the two extensions and the resulting program can again be solved by MASM such that each iteration takes linear time. Moreover, all extensions can be easily incorporated into MASM-BS to consider buffer sizing as well.

A. Wire Area Consideration

Besides delay, we sometimes want to consider some other objectives as well. In this subsection, we use wire area as an example and we consider three cases.

1) *Minimization of a Weighted Sum of Delay and Area:* First, note that

$$\text{wire area} = \sum_{k=0}^{m} h_1 l_{kn+1} + \cdots + h_n l_{(k+1)n} = \mathbf{h}^T \mathbf{l}$$

where $\mathbf{h} = (h_1, \cdots, h_n, \cdots\cdots, h_1, \cdots, h_n)^T$. Then the objective is $\mu(\frac{1}{2}\mathbf{l}^T \mathbf{\Psi}\ \mathbf{l} + \boldsymbol{\varrho}^T \mathbf{l}) + \gamma \mathbf{h}^T \mathbf{l} = \frac{1}{2}\mu \mathbf{l}^T \mathbf{\Psi} \mathbf{l} + (\mu \boldsymbol{\varrho}^T + \gamma \mathbf{h}^T)\mathbf{l}$ for some given constants μ and γ. So the problem can be formulated as follows:

$$\begin{array}{ll} \text{minimize} & (\mu/2)\mathbf{l}^T \mathbf{\Psi} \mathbf{l} + (\mu \boldsymbol{\varrho}^T + \gamma \mathbf{h}^T)\mathbf{l} \\ \text{subject to} & l_1 + \cdots + l_{(m+1)n} = L \\ & l_i \geq 0 \quad \text{for } 1 \leq i \leq (m+1)n. \end{array}$$

Note that this program is of the same form as $\mathcal{CQP}$ and hence can be solved by MASM as in Section II-D.

2) *Delay Minimization with Bounded Area:* The problem can be formulated as follows:

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}\mathbf{l}^T \mathbf{\Psi} \mathbf{l} + \boldsymbol{\varrho}^T \mathbf{l} \\ \text{subject to} & l_1 + \cdots + l_{(m+1)n} = L \\ & \mathbf{h}^T \mathbf{l} \leq b_{\text{area}} \\ & l_i \geq 0 \quad \text{for } 1 \leq i \leq (m+1)n \end{array}$$

where b_{area} is the area bound. We can solve the program by the active set method. If the area constraint is inactive, than that iteration can be solved in closed form as before. If the area constraint is active, the matrix $\mathbf{\Gamma}_{\mathcal{A}}$ in $\mathcal{RCQP}$ for that iteration will contain two rows. However, it is clear that it can still be solved in linear time. In fact, it is not difficult to see that if $\mathbf{\Gamma}_{\mathcal{A}}$ has $O(1)$ rows (i.e., we have $O(1)$ nontrivial equalities), the iteration still takes linear time.

3) *Area Minimization with Bounded Delay:* This case is not as simple since the resulting mathematical program is no longer a quadratic program

$$\begin{array}{ll} \text{minimize} & \mathbf{h}^T \mathbf{l} \\ \text{subject to} & \frac{1}{2}\mathbf{l}^T \mathbf{\Psi} \mathbf{l} + \boldsymbol{\varrho}^T \mathbf{l} \leq b_{\text{delay}} \\ & l_1 + \cdots + l_{(m+1)n} = L \\ & l_i \geq 0 \quad \text{for } 1 \leq i \leq (m+1)n \end{array}$$

where b_{delay} is the delay bound. We can solve this problem by the Lagrangian relaxation technique as in [2]. Basically, the program above is reduced to a sequence of programs of the following form:

$$\begin{array}{ll} \text{minimize} & \mathbf{h}^T \mathbf{l} + \lambda(\frac{1}{2}\mathbf{l}^T \mathbf{\Psi} \mathbf{l} + \boldsymbol{\varrho}^T \mathbf{l} - b_{\text{delay}}) \\ \text{subject to} & l_1 + \cdots + l_{(m+1)n} = L \\ & l_i \geq 0 \quad \text{for } 1 \leq i \leq (m+1)n \end{array}$$

where λ is the Lagrange multiplier. It is again of the form of $\mathcal{CQP}$ and hence can be solved by MASM.

B. Additional Constraints

We can add restrictions to the solution by adding constraints to the convex quadratic program. For example, we may require that the section of the wire within a distance d from the sink cannot be wider than h. If t is the index such that $h_t > h \geq h_{t+1}$, then the corresponding program will be

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}\mathbf{l}^T \mathbf{\Psi} \mathbf{l} + \boldsymbol{\varrho}^T \mathbf{l} \\ \text{subject to} & l_1 + \cdots + l_{mn+t} \leq L - d \\ & l_1 + \cdots + l_{(m+1)n} = L \\ & l_i \geq 0 \quad \text{for } 1 \leq i \leq (m+1)n \end{array}$$

We can also easily restrict the position of the buffers inserted, the distance between two consecutive buffers, etc. For example, if we want the first buffer to be at a distance between d_1 and d_2 from the source, then the problem can be formulated as follows:

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}\mathbf{l}^T \mathbf{\Psi} \mathbf{l} + \boldsymbol{\varrho}^T \mathbf{l} \\ \text{subject to} & l_1 + \cdots + l_n \geq d_1 \\ & l_1 + \cdots + l_n \leq d_2 \\ & l_1 + \cdots + l_{(m+1)n} = L \\ & l_i \geq 0 \quad \text{for } 1 \leq i \leq (m+1)n. \end{array}$$

As we mentioned above, as long as $O(1)$ constraints are added, the resulting program can be handled by MASM such that each iteration can be done in linear time.

VI. Experimental Results and Concluding Remarks

In this section, we demonstrate the performance of the algorithms MASM and MASM-BS in practice. For MASM, we show that the runtime is $O(mn^2)$ in practice. For MASM-BS, we show that because of the pruning technique, the actual runtime is much better than the complexity bound given in Section IV. We have implemented the algorithms in the C Language. We run them on a PC with a 300-MHz Pentium II processor and 64 MB of memory. We use the parameters for the 0.18-μm technology listed in [10].

For MASM, we have shown that the runtime of each iteration is only $O(mn)$. In the experiment below, we would like to find out the dependency of the number of iterations of MASM on the number of choices of wire width n and on the number of buffers m. We run MASM for the simultaneous buffer insertion and wire sizing problem on a wide range of values for n and m. For each pair of values for n and m, we run our algorithm on 100 randomly generated interconnect instances of length between 5000 μm and 20 000 μm. The average number of iterations and CPU time over the 100 instances are reported in Table I.

An observation is that the number of iterations is about n and is basically independent of m. So for the problem $\mathcal{BIWS}$, algorithm MASM runs in $O(mn^2)$ time in practice.

Nowadays, the values of n and m that actually used are usually less than ten. So the running time is negligible. Even for an instance of 100 choices of wire width and 100 buffers, the algorithm still takes only 0.92 s.

We also compare the runtime of MASM with some general purpose convex quadratic program solvers. There are many public domain or commercial software systems that can solve convex quadratic programs (e.g., LOQO, CPlex, OSL, MINO). LOQO is one of the fastest systems available. So we use LOQO for the comparison. We notice that MASM, being based on the observations in Section II-D, is much faster than LOQO. For small problems ($n = 10$ and $m = 10$),

TABLE I
THE AVERAGE NUMBER OF ITERATIONS AND CPU TIME OF THE ALGORITHM MASM FOR SIMULTANEOUS BUFFER INSERTION AND WIRE SIZING

# width chices	# buffers	# variables	Algorithm MASM	
n	m	$(m+1)n$	# iterations	CPU time (s)
10	0	10	9.25	<0.001
10	10	110	11.86	0.001
10	40	410	13.15	0.004
10	70	710	13.83	0.008
10	100	1010	14.09	0.011
40	0	40	37.60	0.001
40	10	440	41.60	0.015
40	40	1640	43.11	0.059
40	70	2840	43.89	0.103
40	100	4040	44.08	0.146
70	0	70	65.87	0.004
70	10	770	71.86	0.047
70	40	2870	73.17	0.177
70	70	4970	73.67	0.309
70	100	7070	74.04	0.447
100	0	100	94.13	0.008
100	10	1100	101.72	0.095
100	40	4100	103.14	0.356
100	70	7100	103.59	0.632
100	100	10100	104.03	0.920

TABLE II
THE OPTIMAL NUMBER OF BUFFERS INSERTED (m^*), THE NUMBER OF COMBINATIONS SOLVED BY MASM (I.E., NOT PRUNED), THE NUMBER OF AGGREGATE LOWER BOUNDS CALCULATED, AND THE CPU TIME OF THE ALGORITHM MASM-BS FOR SIMULTANEOUS BUFFER INSERTION, BUFFER SIZING, AND WIRE SIZING

$L(\mu$m)	m^*	# Combinations solved	# Aggregate lower bounds calculated	CPU time (s)
3000	1	3	17	<0.001
6000	1	4	17	<0.001
9000	1	24	47	0.002
1 2000	2	152	269	0.018
1 5000	2	373	797	0.052

our algorithm is about 15 times faster. For larger problems ($n = 100$ and $m = 100$), our algorithm is more than 30 times faster.

For MASM-BS, we are interested to know the number of combinations that are solved by MASM (i.e., the number of combinations not pruned). We run our algorithm MASM-BS on interconnect wires of different length. We use ten choices for wire width and six choices for buffer size. We set the maximum number of buffers allowed for a single wire M to ten. So for each simultaneous buffer inserting, buffer sizing and wire sizing problem instance, we have $1+6+6^2+\cdots+6^{10} = 73$ million possible combinations. The optimal number of buffers inserted, the number of combinations that are solved by MASM (i.e., not pruned), the number of aggregate lower bounds calculated, and the CPU time are reported in Table II. It shows that the algorithm MASM-BS is extremely efficient in practice. Even for the slowest case, only 0.052 s is required. It also shows that the pruning technique introduced in Section IV is very effective. Almost all the combinations are pruned.

We notice that the pruning technique is more effective for a larger m. In fact, we observe that if the optimal number of buffers inserted is m^*, then the bound in Lemma 7 will prune almost all combinations with more than $m^* + 1$ buffers. An explanation for this observation is given below. A larger m means more combinations have been considered. This implies a smaller minimum delay obtained so far, or equivalently, a better upper bound used in pruning. In particular, when m equals m^*, the optimal combination is found. On the other hand, a larger m also means more delay due to the input capacitance of the buffers and more intrinsic buffer delay. So for a larger value of m, a larger proportion of the combinations is pruned.

The memory requirement of MASM and MASM-BS is proportional to the size of the convex quadratic programs formulated. Each variable needs about 50 bytes. So for the simultaneous buffer insertion, buffer sizing and wire sizing problem considered above ($M = 10, n = 10$), the memory requirement is only $(M+1)n \times 50$ bytes $= 0.005$ MB.

Note that for our approach, the size of the quadratic program formulated is independent of the wire length. So for MASM, the runtime is basically independent of the wire length. For MASM-BS, a longer wire requires more buffers. So in general, more combinations needed to be solved. For the traditional approach of dividing the wire into small fixed-length segments, a longer wire needs to be divided into more segments in order to obtain comparable accuracy.

In the future, we would like to extend our approach to handle nets with tree topology. For weighted sink delay objective, our algorithm MASM-BS can be applied to nets with tree topology by a similar technique as in [6]. That is we use an iterative algorithm to optimize the tree edges one at a time.

At each time we optimize an edge, we keep all the other edges fixed and apply MASM-BS to that edge. For other objectives like minimizing maximum delay or minimizing area with delay bounds, the problems can be solved by the Lagrangian relaxation technique as in [6]. Basically, the Lagrangian relaxation technique reduces the problems into a sequence of problems of minimizing weighted sink delay, where the sink weights are just the Lagrange multipliers. The weighted sink delay problems can be solved by the iterative technique described above. The idea above solves the problems using the optimal number of buffers. However, if there is a bound on the total number of buffers allowed for the whole tree, we speculate that combining our approach with dynamic programming to distribute the buffers among the edges will be needed.

ACKNOWLEDGMENT

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

REFERENCES

[1] C. Alpert and A. Devgan, "Wire segmenting for improved buffer insertion," in *Proc. ACM/IEEE Design Automation Conf.*, 1997, pp. 588–593.
[2] C.-P. Chen, Y.-W. Chang, and D. F. Wong, "Fast performance-driven optimization for buffered clock trees based on Lagrangian relaxation," in *Proc. IEEE Int. Conf. Computer-Aided Design*, 1996, pp. 405–408.
[3] C.-P. Chen, Y.-P. Chen, and D. F. Wong, "Optimal wire-sizing formula under the Elmore delay model," in *Proc. ACM/IEEE Design Automation Conf.*, 1996, pp. 487–490.
[4] 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.
[5] ——, "Optimal wire-sizing function with fringing capacitance consideration," in *Proc. ACM/IEEE Design Automation Conf.*, 1997, pp. 604–607.
[6] C.-P. Chen, H. Zhou, and D. F. Wong, "Optimal nonuniform wire-sizing under the Elmore delay model," in *Proc. IEEE Int. Conf. Computer-Aided Design*, 1996, pp. 38–43.
[7] C. C. N. Chu and D. F. Wong, "Closed form solution to simultaneous buffer insertion/sizing and wire sizing," in *Proc. Int. Symp. Physical Design*, 1997, pp. 192–197.
[8] C. C. N. Chu and D. F. Wong, "A polynomial time optimal algorithm for simultaneous buffer and wire sizing," in *Proc. Conf. Design Automation and Test in Europe*, 1998, pp. 479–485.
[9] J. Cong and L. He, "Optimal wiresizing for interconnects with multiple sources," *ACM Trans. Design Automation of Electron. Syst.*, vol. 1, no. 4, Oct. 1996.
[10] 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. Computer-Aided Design*, 1997, pp. 478–485.
[11] 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.
[12] J. Cong, C.-K. Koh, and K.-S. Leung, "Simultaneous buffer and wire sizing for performance and power optimization," in *Proc. Int. Symp. Low-Power Electronics and Design*, Aug. 1996, pp. 271–276.
[13] J. Cong and K.-S. Leung, "Optimal wiresizing under the distributed Elmore delay model," in *Proc. IEEE Int. Conf. Computer-Aided Design*, 1993, pp. 634–639.
[14] S. Dhar and M. A. Franklin, "Optimum buffer circuits for driving long uniform lines," *IEEE J. Solid-State Circuits*, vol. 26, no. 1, pp. 32–40, Jan. 1991.
[15] 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.
[16] J. P. Fishburn, "Shaping a VLSI wire to minimize Elmore delay," in *Proc. Eur. Design and Test Conf.*, 1997.
[17] J. P. Fishburn and C. A. Schevon, "Shaping a distributed-RC line to minimize Elmore delay," *IEEE Trans. Circuits and Systems I*, vol. 42, pp. 1020–1022, Dec. 1995.
[18] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman, NY, 1979.
[19] N. Hedenstierna and K. O. Jeppson, "CMOS circuit speed and buffer optimization," *IEEE Trans. Computer-Aided Design*, vol. CAD-6, pp. 270–281, Mar. 1987.
[20] R. C. Jaeger, "Comments on 'An optimized output stage for MOS integrated circuits'," *IEEE J. Solid-State Circuits*, vol. SC-10, pp. 185–186, June 1975.
[21] M. K. Kozlov, S. P. Tarasov, and L. G. Khachiyan, "Polynomial solvability of convex quadratic programming," *Soviet Mathematics Doklady*, vol. 20, pp. 1108–1111, 1979.
[22] 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.
[23] H. C. Lin and L. W. Linholm, "An optimized output stage for MOS integrated circuits," *IEEE J. Solid-State Circuits*, vol. SC-10, pp. 106–109, Apr. 1975.
[24] D. G. Luenberger, *Linear and Nonlinear Programming*, 2nd ed. Reading, MA: Addison Wesley, 1984.
[25] N. Menezes, R. Baldick, and L. T. Pileggi, "A sequential quadratic programming approach to concurrent gate and wire sizing," in *Proc. IEEE Int. Conf. Computer-Aided Design*, 1995, pp. 144–151.
[26] S. S. Sapatnekar, "RC interconnect optimization under the Elmore delay model," in *Proc. ACM/IEEE Design Automation Conf.*, 1994, pp. 387–391.
[27] Semiconductor Industry Association, *The National Technology Roadmap for Semiconductors*, 1997.
[28] L. P. P. P. van Ginneken, "Buffer placement in distributed RC-tree networks for minimal Elmore delay," in *Proc. Int. Symp. Circuits and Systems*, 1990, pp. 865–868.
[29] Q. Zhu, W. W.-M. Dai, and Joe G. Xi, "Optimal sizing of high-speed clock networks based on distributed RC and lossy transmission line models," in *Proc. IEEE Int. Conf. Computer-Aided Design*, 1993, pp. 628–633.

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

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