An Efficient Buffer Insertion Algorithm for Large Networks Based on Lagrangian Relaxation

I-Min Liu[1] Adnan Aziz[1] D. F. Wong[2] Hai Zhou[3]
[1]Electrical and Computer Engineering, The University of Texas at Austin, Texas 78712
[2]Computer Sciences, The University of Texas at Austin, Austin, Texas 78712
[3]Advanced Technology Group, Synopsys Inc., Mountain View, CA 94043

Abstract

We propose a novel buffer insertion algorithm for handling more general networks, whose underlying topology is a directed acyclic graph rather than just a RC tree. The algorithm finds a global buffering which minimizes buffer area while meeting the timing constraints. We use Lagrangian relaxation to translate the timing constraints to a cost in the objective function, and simplify the resulting objective function using the special structure of the problem we are solving. The core of the algorithm is a local refinement procedure, which iteratively computes the optimal buffering for each edge so as to minimize a weighted area and delay objective. The resulting procedure is fast, and takes full advantage of the slack available on noncritical paths.

1 Introduction

Until recently, the cycle time of digital ICs has been dominated by the delays of the combinational modules. Considerable effort has gone into automatic optimization of the cycle time using techniques such as logic restructuring [1] and gate resizing [2, 3]. However, as VLSI fabrication technology reaches deep submicron, interconnect delays are becoming increasingly significant. Today, it is widely reported that in many designs, as much as 50% of the clock cycle is consumed by interconnect delay — this delay comes predominantly from the global interconnect. As technology advances scale device dimensions, the significance of interconnect delay is expected to increase further in the future. Buffer insertion is an extremely effective technique for reducing interconnect delay in both theory and practice. In this paper, we propose an efficient algorithm for buffer insertion that can handle complex network topologies while meeting timing constraints.

There has been considerable previous research in buffer insertion [4, 5, 6, 7, 8]. It has been shown that when a single long wire is properly buffered, its delay can be reduced to a linear function of its length [9]. Van Ginnekan [4] proposed a polynomial time dynamic programming (DP) algorithm for buffer placement in distributed RC trees. It has since been generalized to other applications — low power [5], buffered Steiner tree construction [10], maze routing [11], etc. Advantages of this procedure include the ability to insert buffers of various sizes. However, the DP formulation is limited to inserting buffers at tree vertices; it gives an approximate solution by "chopping", i.e., treating individual tree edges as a connection of wire segments. To obtain a buffering which is close to optimal, the number of wire segments needs to be made large for long wires. Alpert and Devgan [6] propose an algorithm for segmenting individual wires to reduce the size of the input to the Van Ginnekan's algorithm, but still yield a result of reasonable quality (they restrict their attention to inserting fixed size buffers). However, the DP formulation is quadratic in time and space usage — as a result, even with reduced input size, it is of little use when buffering very large trees, such as those corresponding to global nets.

Chu and Wong [7] propose a quadratic programming approach that can be used for inserting buffers on a wire to achieve minimal delay. Their approach is efficient in that the runtime is proportional to the number of buffers inserted; as such it overcomes the complexity introduced by segmenting. However, their algorithm considers only one wire at a time. Their exact formulation is limited to fixed size buffers — the extension to variable size buffers is ad hoc. Kang and Dai [8] propose a delay bounded buffered tree construction procedure (buffers are noninverting and constant size). In their work, a buffered tree is incrementally constructed to meet the delay bound, and area is recovered by deleting unnecessary buffers afterwards using a greedy selection heuristic. The runtime is $O(k \cdot n^2)$, where k is the maximum number of buffers on an edge of the tree, and n is the number of edges in the tree. The quadratic dependence on n leads to unacceptable performance on large designs.

Our primary criticism of all these previous papers is that they operate on individual wires or RC trees. At the chip level, the connections between the component modules constitute a directed acyclic graph, and signals need to meet a large number of timing constraints. An optimum procedure for buffering routing trees can be used to obtain a design which is as fast as possible simply by optimally buffering each individual net. However, such

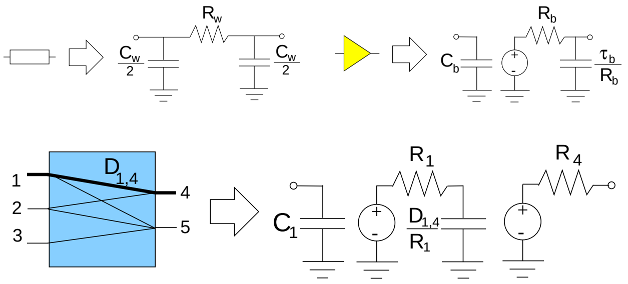

Figure 1. Models of a wire, a buffer and a module.

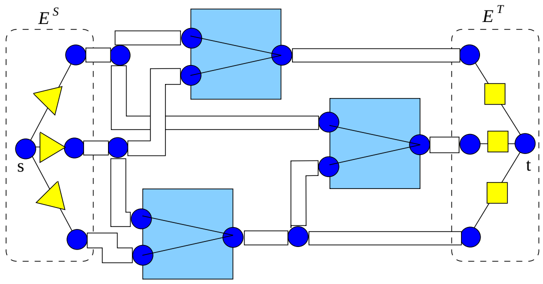

Figure 2. Graph representing a network.

an approach will be extremely wasteful in terms of area of buffers used. Buffers can be recovered by selective removal from noncritical nets. However, such an approach is by nature greedy.

In this paper, we present an extremely efficient buffer insertion algorithm that can handle large networks, whose underlying topology is a DAG, with arbitrary input/output timing specifications. Specifically, it inserts a minimum number of buffers subject to timing constraints. As in [6, 7, 8], buffers are assumed to be noninverting, and of a fixed size. The total storage and runtime per iteration are both linear in the size of the network. Because of this, it can also be used for buffer planning at early design stages and integrated with synthesis tools. The procedure is based on Lagrangian relaxation — it derives its speed from the crucial observation that the structure of the underlying network can be used to considerably simplify the constraints. As such, our approach is inspired by the dramatic advance made to the gate sizing problem by Chen et al. [12] who also exploits network topology to simplify the objective function in the Lagrangian relaxation subproblem.

The remainder of this paper is organized as follows. In Section 2, we present the notation, definitions and formulation of the problem. In Section 3, we discuss buffer insertion for minimizing a weighted area and delay objective. In Section 4, we present our algorithm for minimizing buffer area under timing constraints. We present experimental results in Section 5 and summarize in Section 6.

2 Formulation

All delays referred in this paper are Elmore delay [13]. We use the RC models shown in Figure 1 for wires, buffers and modules. For a wire w, let l_w, C_w, and R_w be the length, capacitance, and resistance, respectively. Using C and R to be the unit length capacitance and resistance, we have $C_w = Cl_w$ and $R_w = Rl_w$. For a buffer b, we use C_b, R_b and τ_b to represent the input capacitance, output resistance and intrinsic delay.

In this paper we consider only combinational modules[1]. We characterize a module by modeling the worst-case path of each input/output pair. That is, an input/output pair in a module is modeled by the capacitances of its input driver, the resistances of its output driver, and the worst-case path delay. In Figure 1, the path from input 1 to output 4 is modeled by the the capacitance of the corresponding input driver c_1, the path delay $D_{1,4}$, and the resistance of the corresponding output driver r_4 and all the other input/output paths are models in the same way. Note that the capacitance c_1 is actually shared by the modeling for path $1 \rightarrow 4$ and path $1 \rightarrow 5$.

The input to our problem is a placed and routed netlists of modules, with a specified set of input drivers and output loads. Our goal is to find a buffering for all nets, such that the specified timing is met and the total buffer area is minimized. A buffering includes the buffer number on each net and their locations. Since we are dealing with fixed-size buffers, minimizing total buffer number is equivalent to minimizing total buffer area.

The fact that both modules and interconnect delays are considered in our formulation makes the convention of using a DAG with modules/junction points corresponding to vertices in the DAG, and wires corresponding to edges notationally inconvenient. Instead, we use an annotated directed acyclic graph. As shown in Figure 2, the vertices of the graph correspond to the primary inputs, primary outputs, tree junctions and module inputs/outputs. In addition, two fictitious nodes s and t are introduced. The source node s is connected to all primary inputs, and all primary outputs are connected to the sink node t. The internal nodes include module inputs/outputs, and tree junctions.

Let $\mathcal{E}^{\mathcal{S}}$ denote the set of edges connected to the source and $\mathcal{E}^{\mathcal{T}}$ those to the sink. The remaining of edges is referred to as the internal edges, and denoted by $\mathcal{E}^{\mathcal{I}}$, which includes two disjoint sets of edges $\mathcal{E}^{\mathcal{W}}$ and $\mathcal{E}^{\mathcal{M}}$, corresponding to wires and the input/output paths of modules, respectively. More specifically, we annotate an edge in $\mathcal{E}^{\mathcal{S}}$ with the resistance of the associated driver and an edge in $\mathcal{E}^{\mathcal{T}}$ the capacitance of the associated load. We also mark an edge in $\mathcal{E}^{\mathcal{W}}$ with the length of the associated wire and an edge in $\mathcal{E}^{\mathcal{M}}$ with the characteristics

[1] Sequential modules can be handled by making latch inputs primary outputs and latch outputs primary inputs.

of the input/output pair of the associated module.

Given an edge $e = (u, v)$, $input(e)$ denotes the set of edges connected to u and $output(e)$ the set of edges connected from v. The set $Des(e)$ denotes the set of the edges lying on the paths starting at v which lead to isolation elements, i.e., to module inputs, buffers, or output loads; the set $Ans(e)$ the set of the edges lying on the paths ending at v which comes from isolation elements, i.e., from module outputs, buffers, or input drivers.

Let e be an edge representing a wire, the number of buffers on the wire is denoted by k_e. When $k_e \geq 1$, the length of the wire from u to the first buffer is denoted by x_e and the wirelength from the last buffer to v is denoted by z_e. In an optimal buffering using fixed size buffers, the spacing between all buffers on a single wire is equal; with this fact in mind, we define y_e be the wirelength between any two adjacent buffers on e.

Let D_e denote the delay through an edge e and a_e the arrival time at e. we have a_e equal to the sum of D_e and the max of the arrival times of the edges in $input(e)$. If e is an edge from s to a primary input, A_e will denote the user-specified arrival time for the input of the driver. Similarly, if e is an edge from a primary output to t, then Q_e will denote the user-specified required time for that primary output.

We are now ready to mathematically express the problem we are trying to solve. Specifically, the problem of minimum buffer insertion under input/output timing constraints for a network can be formulated as follows:

$$
\begin{array}{lll}
\min & \sum_{e \in \mathcal{E}^{\mathcal{W}}} \alpha k_e & \\
\text{s.t.} & a_e \leq Q_e & \forall e \in \mathcal{E}^{\mathcal{T}} \\
& a_{e'} + D_e \leq a_e & \forall e \in \mathcal{E}^{\mathcal{I}} \text{ and } e' \in input(e) \\
& A_e + D_e \leq a_e & \forall e \in \mathcal{E}^{\mathcal{S}} \\
& k_e \geq 0 & \forall e \in \mathcal{E}^{\mathcal{W}}
\end{array}
$$

The variables to be solved are k_e, x_e, and z_e for all edges representing wires. However, this problem is difficult to solve directly due to the complex network topology and the large number of timing constraints. Indeed, even simple instances of the minimal buffer insertion problem are known to be NP-hard [14]. We will consider a different objective, which is easier to handle in our approach, and then generalize it to handle the primal problem.

3 Weighted Area and Delay Objective

In this section, we consider the problem of minimizing a weighted area and delay objective, i.e., minimizing $\mathcal{O}_\beta = \sum_{e \in \mathcal{E}} (\alpha k_e + \beta_e D_e)$, where β_e is the delay weight for edge e. At first glance this problem has a synthetic quality; however, we will see in the next section that Lagrangian relaxation of the primal problem yields an objective of exactly this form.

To simplify the representation, we use the following notation to abbreviate expressions. We use $\bar{R}_e$ to represent the resistance of an edge e that is not isolated from downstream edges, and $\bar{C}_e$ the capacitance of an edge e that is not isolated from upstream edges. For example, for a wire e, when $k_e = 0$, $\bar{R}_e = Rl_e$ and $\bar{C}_e = Cl_e$; when $k_e = 3$, $\bar{R}_e = R_b + Rz_e$ and $\bar{C}_e = C_b + Cx_e$. Let $\hat{R}_e = \sum_{e' \in\, Ans(e)} \bar{R}_{e'}$ and $\hat{C}_e = \sum_{e' \in\, Des(e)} \bar{C}_{e'}$; intuitively, $\hat{R}_e$ is the sum of upstream resistances and $\hat{C}_e$ is the sum of downstream capacitances. Define $\hat{R'}_e$ to be $\sum_{e' \in\, Ans(e)} \beta_{e'} \bar{R}_{e'}$, i.e., the sum of weighted upstream resistances. Also, we denote the sum $\tau_b + \alpha/\beta_e$ by $\hat{\tau}_e$.

Our procedure for optimizing $\mathcal{O}_\beta$ will proceed by local refinement. That is for each edge we will select a buffer that locally minimize $\mathcal{O}_\beta$. Focusing on an edge e, the objective can be considered as the sum of two parts: a function of the variable associated with e and terms independent of those parameters. That is, $\mathcal{O}_\beta = F_e +$ other terms. When $\mathcal{O}_\beta$ is being minimized with respect to e, we only need to concentrate on F_e. When $k_e = 0$,

$$
\begin{aligned}
F_{e,0} &= \Big[\sum_{e' \in Ans(e)} \beta_{e'} \bar{R}_{e'}\Big] \Big[Cl_e + \sum_{e' \in Des(e)} \bar{C}_{e'}\Big] \\
&+ Rl_e \Big[\frac{1}{2} Cl_e + \sum_{e' \in Des(e)} \bar{C}_{e'}\Big] \\
&= (\hat{R'}_e + \frac{1}{2} Rl_e)(Cl_e + \hat{C}_e)
\end{aligned}
$$

When $k_e = 1$,

$$
\begin{aligned}
F_{e,1} &= \alpha k_e + \left[\sum_{e' \in Ans(e)} \beta_{e'} \bar{R}_{e'}\right] (Cx_e + C_b) \\
&+ \beta_e \left[\frac{1}{2} RCx_e^2 + Rx_e C_b + \tau_b \right. \\
&+ R_b \left[C(l_e - x_e) + \sum_{e' \in Des(e)} \bar{C}_{e'}\right] \\
&+ \left. \frac{1}{2} RC(l_e - x_e)^2 + R(l_e - x_e) \sum_{e' \in Des(e)} \bar{C}_{e'}\right) \\
&= \hat{R'}_e (Cx_e + C_b) + \beta_e \left[\frac{1}{2} RCx_e^2 + Rx_e C_b + \hat{\tau}_e \right. \\
&+ \left. R_b \left[C(l_e - x_e) + \hat{C}_e\right] + \frac{1}{2} RC(l_e - x_e)^2 + R(l_e - x_e)\hat{C}_e\right]
\end{aligned}
$$

Since $F_{e,1}$ is quadratic in x_e and the first derivative of $F_{e,1}$ is positive, it is minimized when $\frac{dF_{e,1}}{dx_e} = 0$ or at the boundary. We have the optimal buffer location

$$
x_e^* = \min(\max(\frac{l_e + \frac{R_b - \hat{R'}_e/\beta_e}{R} - \frac{C_b - \hat{C}_e}{C}}{2}, 0), l_e)
$$

As a special case, $x_e^* = l/2$ when $\hat{R'}_e/\beta_e = R_b$ and $\hat{C}_e = C_b$. Physically, this result states that the optimum location for a buffer on a wire with downstream capacitance C_b and upstream resistance R_b is exactly in the middle. This justifies that the wire lengths between consecutive buffers on an edge are equal.

When $k_e > 1$, the objective can be written as

$$\begin{aligned}
F_{e,k_e} &= \alpha k_e + \left[\sum_{e' \in Ans(e)} \beta_{e'} \bar{R}_{e'}\right](Cx_e + C_b) \\
&+ \beta_e \left[\frac{1}{2}RCx_e^2 + Rx_eC_b + \right. \\
&+ (k_e - 1)\left[R_b(Cy_e + C_b) + \tau_b + \frac{1}{2}RCy_e^2 + Ry_eC_b\right] \\
&+ R_b\left[Cz_e + \sum_{e' \in Des(e)} C_{e'}\right] + \tau_b \\
&+ \left.\frac{1}{2}RCz_e^2 + Rz_e \sum_{e' \in Des(e)} C_{e'}\right] \\
&= \hat{R'}_e(Cx_e + C_b) + \beta_e \left[\frac{1}{2}RCx_e^2 + Rx_eC_b\right. \\
&+ (k_e - 1)\left[R_b(Cy_e + C_b) + \hat{\tau}_e + \frac{1}{2}RCy_e^2 + Ry_eC_b\right] \\
&+ \left. R_b(Cz_e + \hat{C}_e) + \hat{\tau}_e + \frac{1}{2}RCz_e^2 + Rz_e\hat{C}_e\right]
\end{aligned}$$

Note that interbuffer spacing y_e on a wire of length l_e, containing k_e buffers where the first buffer is placed x_e from the beginning of the wire, and the last buffer is placed z_e from the end of the wire is equal to $(l_e - x_e - z_e)/(k_e - 1)$.

Considering F_{e,k_e} as a function of x_e and z_e, since the Hessian [15] is positive definite for any $k_e > 1$, F_{e,k_e} is a convex function of x_e and z_e. Thus, F_{e,k_e} is minimized when $\frac{\partial F_{e,k_e}}{\partial x_e} = 0$ and $\frac{\partial F_{e,k_e}}{\partial z_e} = 0$ or at the boundary. Through a case analysis we have derived the optimal x_e and z_e under different conditions. These results are summarized in Table 1.

Substituting x_e^* and z_e^* into F_{e,k_e}, we have F_{e,k_e} in terms of k_e. For the unconstrained case, the function is of the form $F_{e,k_e} = \frac{Ak_e^2 + Bk_e + C}{k_e + 1}$ where $A > 0$. Setting $\frac{dF_{e,k_e}}{dk_e} = 0$ yields $Ak_e^2 + 2Ak_e + (B - C) = 0$, which has two roots of the form $-1 \pm \frac{\sqrt{A-B+C}}{\sqrt{A}}$, in which there is at most one positive root. Since $\lim_{k_e \to \infty} F_{e,k_e} = \infty$, by finding the k_e satisfying $\frac{dF_{e,k_e}}{dk_e} = 0$, we have the optimal number of buffers when x_e and z_e is unconstrained. We have conducted a similar analysis and obtain a k_e^* under different conditions. These results are summarized in Table 2.

In summary, by the case analysis above we can guarantee that the Table 1 and 2 give the optimal k_e^*, x_e^* and z_e^* for locally minimizing the objective $\mathcal{O}_\beta$.

4 Minimizing Area under Timing Constraints

In this section, we transform the timing constrained buffer area minimization problem into weighted area and delay minimization sub-problems by Lagrangian relaxation. Lagrangian relaxation is a general strategy for solving a constrained optimization problem. It permits us to decompose a problem into sub-problems. Even when the relaxed problem is convex, direct solution of the Lagrangian subproblem using general-purpose nonlinear solvers is not usually very efficient. We will use the special structure of our problem to considerably simplify the relaxed objective function. Readers are encouraged to read in [16, Chapter 16] for a comprehensive introduction to Lagrangian relaxation and its application to combinatorial optimization.

We relax the constraints in the primal problem and add them to the objective function by multiplying each by a nonnegative penalty, called the corresponding "Lagrangian multiplier". Following this, the primal problem is transformed to maximizing the following dual problem:

$$\begin{aligned}
L_\lambda &= \min\{\sum_{e \in \mathcal{E}} \alpha k_e + \sum_{e \in \mathcal{E}^{\mathcal{T}}} \lambda_{et}(a_e - Q_e) \\
&+ \sum_{e \in \mathcal{E}^{\mathcal{I}}} \sum_{e' \in\, input(e)} \lambda_{e'e}(a_{e'} + D_e - a_e) \\
&+ \sum_{e \in \mathcal{E}^{\mathcal{S}}} \lambda_{se}(A_e + D_e - a_e)\}
\end{aligned}$$

subject to the constraint that the Lagrangian multipliers are nonnegative real numbers, and the k_e's are nonnegative integers.

For a given set of λ, at an optimal solution for k^* and a^*, the slope must be zero (this is referred to as the Kuhn-Tucker condition [17]). Thus for every $e \in \mathcal{E}$, we have $\frac{\partial L_\lambda(k^*, a^*)}{\partial a_e} = 0$. These equations simplify to $\sum_{e' \in\, input(e)} \lambda_{e'e} = \sum_{e' \in\, output(e)} \lambda_{ee'}$. When this equation is satisfied, the a_e terms are dropped. Let $\otimes_\lambda$ be the set of λ satisfying this relation. This yields to the problem of minimizing a function which has the form of a weighted area and delay objective:

$$\sum_{e \in \mathcal{E}^{\mathcal{I}}} \alpha k_e + \sum_{e \in \mathcal{E}^{\mathcal{S}} \cup \mathcal{E}^{\mathcal{I}}} \sum_{e' \in\, input(e)} \lambda_{e'e} D_e$$

Denoting $\sum_{e' \in\, input(e)} \lambda_{e'e}$ by β_e, we have the following objective function to minimize:

$$\sum_{e \in \mathcal{E}^{\mathcal{I}}} \alpha k_e + \sum_{e \in \mathcal{E}^{\mathcal{S}} \cup \mathcal{E}^{\mathcal{I}}} \beta_e D_e$$

	$x_e^* = 0$	$0 < x_e^* \le l_e$
$z_e^* = 0$	$x_e^* = 0$ $z_e^* = 0$	$x_e^* = \frac{l_e+(k_e-1)(R_b-\hat{R'}_e/\beta_e)/R}{k_e}$ $z_e^* = 0$
$0 < z_e^* \le l_e$	$x_e^* = 0$ $z_e^* = \frac{l_e+(k_e-1)(C_b-\hat{C}_e)/C}{k_e}$	$x_e^* = \frac{l_e+k_e(R_b-\hat{R'}_e/\beta_e)/R-(C_b-\hat{C}_e)/C}{k_e+1}$ $z_e^* = \frac{l_e-(R_b-\hat{R}'_e/\beta_e)/R+k_e(C_b-\hat{C}_e)/C}{k_e+1}$

Table 1. Optimal x_e and z_e when $k_e > 1$

	$x_e^* = 0$	$0 < x_e^* \le l_e$
$z_e^* = 0$	$\frac{l_eRC}{\sqrt{2RC(R_bC_b+\hat{\tau}_e)}} + 1$	$\frac{l_eRC-C(R_b-\hat{R'}_e/\beta_e)}{\sqrt{2RC(R_bC_b+\hat{\tau}_e)}}$
$0 < z_e^* \le l_e$	$\frac{l_eRC-R(C_b-\hat{C}_e)}{\sqrt{2RC(R_bC_b+\hat{\tau}_e)}}$	$\frac{l_eRC-C(R_b-\hat{R'}_e/\beta_e)-R(C_b-\hat{C}_e)}{\sqrt{2RC(R_bC_b+\hat{\tau}_e)}} - 1$

Table 2. Optimal k_e when $k_e > 1$

Algorithm $\mathcal{WOBA}$
input: a network and the weights of the edges
output: number of buffers and their locations
1. for each e in a topological order from t to s **do**
2. compute $\hat{C}_e$;
3. for each e in a topological order from s to t **do**
4. compute $\hat{R'}_e$;
5. compute k_e^*, x_e^*, z_e^*;
6. repeat 1-5 if there is improvement;

Figure 3. Algorithm $\mathcal{WOBA}$.

Algorithm $\mathcal{WOBA}/\mathcal{LR}$
input: a network, timing constraints
output: number of buffers and their locations
1. find a initial $\lambda \in \otimes_\lambda$;
2. compute β_e for every $e \in E$;
3. call $\mathcal{WOBA}$ to solve $\mathcal{O}_\beta$;
4. compute a_e's for every $e \in E$;
5. update λ;
6. project λ to the nearest λ in $\otimes_\lambda$;
7. repeat 1-6 if there is improvement;

Figure 4. Algorithm $\mathcal{WOBA}/\mathcal{LR}$.

We have discussed the problem of minimizing the weighted area and delay objective earlier. Now we present an algorithm for solving it by local refinement. The pseudocode is shown in Figure 3. Let n be the number of the edges in a graph. At each iteration, in steps 1–2 the routine computes for each edge the sum of downstream capacitances. After calculating the sum of the downstream capacitances, we store the resulting value. In this manner, in steps 1–2 can be performed in linear time, We saw in the previous section how to compute the values k_e^*, x_e^*, z_e^* in constant time. The sum of the weighted upstream resistances can also be computed in linear time, analogous to the computation for the capacitances. Thus steps 3–5 can also be performed in linear time. Thus steps 1–5 can be performed in linear time.

The algorithm for solving a timing constrained network buffer insertion problem is shown in Figure 4. The Lagrangian multipliers used to decide the weight of the edges are nonnegative numbers. We use the subgradient method to update the Lagrangian multipliers. That is,

$$\begin{array}{lcll} \lambda_{et} & = & [\lambda_{et} + \rho_k(a_e - Q_e)]^+ & \forall e \in \mathcal{E}^{\mathcal{T}} \\ \lambda_{e'e} & = & [\lambda_{e'e} + \rho_k(a'_e + D_e - a_e)]^+ & \forall e \in \mathcal{E}^{\mathcal{I}},\ e' \in input(e) \\ \lambda_{se} & = & [\lambda_{se} + \rho_k(A_e + D_e - a_e)]^+ & \forall e \in \mathcal{E}^{\mathcal{S}} \end{array}$$

where t^+ denotes t if $t > 0$ and 0 otherwise, and ρ_k is the step size [17].

We have observed that the weighted area and delay objective is a convex function when looking at only the variables associated with an edge. However, there is no proof that the objective is convex with respect to all variables. Nevertheless, experimental results justify the performance of the algorithm regardless of this constraint for optimality.

5 Experimental Results

We implemented our algorithm in C and experimented with it on a 64MB Pentium-II-266 running RedHat Linux 5.2. The experiments are performed on a $0.1\mu m$ technology predicted in NTRS'97 [18] and we use the same values of the parameters as those derived in [9]. We have generated a series of benchmarks and run our experiments on them.

Table 3 shows the results for minimizing delay without considering area (this can be achieved by setting area weight α to zero). We observed the initial and final delays, and derived the lower bound from the dual problem [16]. The results show that the solutions of $\mathcal{WOBA}/\mathcal{LR}$ are nearly-optimal, i.e., the resulting delays are close to the lower bounds. In addition, the runtime is acceptable even for the large benchmarks.

In Table 4 we report the results obtained by minimizing buffer area under delay constraints. We also compare them to those obtained by minimizing the maximum de-

Network Size		Delay (ns)			Runtime
Total	# Wires	Init	Final	LB	(seconds)
35	17	19.86	19.81	19.63	0.0
66	37	29.84	28.79	28.79	0.6
172	98	50.79	31.99	31.65	1.4
343	197	73.40	43.76	43.42	10.4
681	398	102.6	50.96	50.87	36.9
1360	799	99.31	56.85	56.01	48.9
2040	1200	112.6	57.02	55.79	179.1
3413	1997	107.2	64.53	59.81	324.0
6807	3997	161.4	77.59	75.44	2091.4

Table 3. The results obtained by minimizing maximum delays.

Network Size		Min Delay		Min Area		
Total	# Wires	Delay	Area	Delay	Area	% Area
35	17	19.81	132	19.81	84	63
66	37	28.79	356	28.79	120	34
172	98	31.99	933	32.03	306	33
343	197	43.76	1708	43.76	420	25
681	398	50.96	3365	50.96	753	22
1360	799	56.85	6740	56.90	1575	23
2040	1200	57.02	10549	57.06	2264	21
3413	1997	64.53	17448	65.43	3456	20
6807	3997	77.59	33990	77.80	6858	20

Table 4. Comparison between the results obtained by (1) minimizing delay without area consideration (Min Delay) and by (2) minimizing buffer area subject to delay constraints (Min Area).

lays. More specifically, we take the delays obtained from the min delay experiments and use them as the delay constraints and minimize the buffer area. The results show that by taking buffer area into consideration, the total buffer areas are significantly smaller, compared to their min-delay counter parts. Furthermore, the delays achieved in both experiments are comparable.

6 Conclusion

In summary, we have developed a novel buffer insertion algorithm for handling more general networks, whose underlying topology is a directed acyclic graph rather than just a RC tree. The algorithm finds a global buffering which minimizes buffer area while meeting the timing constraints. We use Lagrangian relaxation to translate the timing constraints to a cost in the objective function, and simplify the resulting objective function using the special structure of the problem we are solving. The core of the algorithm is a local refinement procedure, which iteratively computes the optimal buffering for each edge so as to minimize a weighted area and delay objective. The resulting procedure is fast, and takes full advantage of the slack available on noncritical paths. Currently we are working on incorporating gate/wire sizing into the existing algorithm, and on inserting variable size buffers.

References

[1] G. D. Micheli, *Synthesis and Optimization of Digital Circuits*. McGraw Hill, 1994.

[2] J. P. Fishburn and A. E. Dunlop, "TILOS: A posynomial programming approach to transistor sizing," in *Proc. Intl. Conf. on Computer-Aided Design*, pp. 326–328, 1985.

[3] S. S. Sapatnekar, V. B. Rao, P. M. Vaidya, and S. M. Kang, "An Exact Solution to The Transistor Sizing Problem for CMOS Circuits Using Convex Optimization," in *IEEE Trans. Computer-Aided Design*, pp. 1621–1634, 1993.

[4] L. P. P. P. van Ginneken, "Buffer Placement in Distributed RC-tree Networks for Minimal Elmore Delay," in *Proc. Intl. Symposium on Circuits and Systems*, pp. 865–868, 1990.

[5] J. Lillis, C.-K. Cheng, and T.-T. Y. Lin, "Optimal Wire Sizing and Buffer Insertion for Low Power and a Generalized Delay Model," in *Proc. Intl. Conf. on Computer-Aided Design*, pp. 138–143, 1995.

[6] C. Alpert and A. Devgan, "Wire Segmenting for Improved Buffer Insertion," in *Proc. of the Design Automation Conf.*, pp. 588–589, 1997.

[7] C. C. N. Chu and D. F. Wong, "A New Approach to Simultaneous Buffer Insertion and Wire Sizing," in *Proc. Intl. Conf. on Computer-Aided Design*, pp. 614–621, 1997.

[8] M. Kang and W. W.-M. Dai, "Delay Bounded Buffered Tree Construction for Timing Driven Floorplanning," in *Proc. Intl. Conf. on Computer-Aided Design*, pp. 707–712, 1997.

[9] J. Cong and Z. Pan, "Interconnect Performance Estimation Models For Synthesis and Design Planning," in *Proc. Intl. Workshop on Logic Synthesis*, pp. 427–433, 1998.

[10] T. Okamoto and J. Cong, "Buffered Steiner Tree Construction with Wire Sizing for Interconnect Layout Optimization," in *Proc. Intl. Conf. on Computer-Aided Design*, pp. 44–49, 1996.

[11] H. Zhou, D. F. Wong, I.-M. Liu, and A. Aziz, "Simultaneously Routing and Buffer Insertion with Restrictions on Buffer Locations," in *Proc. of the Design Automation Conf.*, pp. 96–99, 1999.

[12] C. P. Chen, C. C. N. Chu, and D. F. Wong, "Fast and Exact Simultaneous Gate and Wire Sizing by Lagrangian Relaxation," in *Proc. Intl. Conf. on Computer-Aided Design*, pp. 614–621, 1998.

[13] W. C. Elmore, "The Transient Response of Damped Linear Network with Particular Regard to Wideband Amplifier," in *Journal of Applied Physics*, pp. 55–63, 1948.

[14] H. Zhou and A. Aziz, "Buffer Minimization in Pass Transistor Logic," in *Proc. Intl. Workshop on Logic Synthesis*, May 1998.

[15] D. Russell, *Optimization Theory*. W. A. Benjamin, 1970.

[16] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network Flows : Theory, Algorithms, And Applications*. Prentice Hall, 1993.

[17] J. F. Shapiro, *Mathematical Programming: Structures and Algorithms*. John Wiley, 1979.

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