Fast and Exact Simultaneous Gate and Wire Sizing by Lagrangian Relaxation

Chung-Ping Chen, Chris C. N. Chu, and D. F. Wong, *Member, IEEE*

***Abstract*— This paper considers simultaneous gate and wire sizing for general very large scale integrated (VLSI) circuits under the Elmore delay model. We present a fast and exact algorithm which can minimize total area subject to maximum delay bound. The algorithm can be easily modified to give exact algorithms for optimizing several other objectives (e.g., minimizing maximum delay or minimizing total area subject to arrival time specifications at all inputs and outputs). No previous algorithm for simultaneous gate and wire sizing can guarantee exact solutions for general circuits. Our algorithm is an iterative one with a guarantee on convergence to global optimal solutions. It is based on Lagrangian relaxation and "one-gate/wire-at-a-time" greedy optimizations, and is extremely economical and fast. For example, we can optimize a circuit with 27 648 gates and wires in 11.53 min using under 23 Mbytes memory on a PC with a 333-MHz Pentium II processor.**

***Index Terms*—Gate sizing, interconnect, Lagrangian relaxation, performance optimization, wire sizing.**

I. Introduction

SINCE the invention of integrated circuits almost 40 years ago, gate sizing has always been an effective technique to achieve desirable circuit performance. As technology continues to scale down, total number of gates and interconnects within a die grows over millions. In such increasingly dense integrated circuits, a significant portion of the total circuit delay comes from the interconnects. Therefore, developing efficient algorithms which can handle large scale gate and interconnect optimization problems are of great importance.

In the past, gate delay was the dominant factor in determining circuit performance. Thus, gate and transistor sizing have been extensively studied in the literature [6], [12], [16], [17], [23]. As interconnect delay plays an increasingly important role in determining circuit performance, wire sizing has been an active research topic in the past few years [2], [4], [7], [9], [19], [22].

Since gate sizes affect wire-sizing solutions and wire sizes affect gate-sizing solutions, it is beneficial to simultaneously size both gates and wires. Several results on simultaneous gate and wire sizing have been reported [2], [7], [8], [18], [20]. Chen *et al*. [2], Cong and Koh [8], Menezes *et al*. [18], and Menezes *et al*. [20] considered a single routing tree together with the driving gate. For simultaneous gate and wire sizing for general circuits, Cong and He [7] minimized a weighted delay using a greedy sizing technique in conjunction with dynamic programming. The algorithm cannot guarantee to give exact solutions for objectives such as minimizing total area subject to maximum delay bound or minimizing maximum delay.

In this paper, we consider simultaneous gate and wire sizing for general very large scale integrated (VLSI) circuits under the Elmore delay model. We present a fast and exact algorithm which can minimize total area subject to maximum delay bound. It can be easily modified to give algorithms for optimizing several other objectives (e.g., minimizing maximum delay or minimizing total area subject to arrival time specifications at all inputs and outputs). Convergence to global optimal solutions is guaranteed for all cases. Our algorithm is based on the Lagrangian relaxation technique, which transforms the problem into a sequence of subproblems called the Lagrangian relaxation subproblems. We show that each subproblem can be greatly simplified by the Kuhn–Tucker conditions and can be solved by an efficient "one-gate/wire-at-a-time" greedy algorithm. So our algorithm is extremely economical and fast. For example, we can optimize a circuit with 27 648 gates and wires in 11.53 min using under 23 Mbytes memory on a PC with a 333-MHz Pentium II processor.

We notice that the gate and wire sizing problem is similar to the transistor sizing problem. In this paper, our problem is formulated as a geometric program [10]. Fishburn and Dunlop [12] have shown a long time ago that the transistor sizing problem can also be formulated as a similar geometric program. However, it would be very slow to solve the geometric program by some general-purpose geometric programming solver. So instead of solving it exactly, Fishburn and Dunlop proposed TILOS [12], which is based on an efficient sensitivity-based heuristic.

Marple [16], [17] solved the geometric program by the Lagrangian augmentation technique. Lagrangian augmentation, like Lagrangian relaxation, is a general technique for constrained nonlinear optimization. The difference between them is that Lagrangian augmentation adds to the Lagrangian a penalty term that helps to steer the solution toward the feasible region. However, we demonstrate in this paper that for the device sizing problems, Lagrangian relaxation is a much better technique. Without the penalty term, tremendous simplification and efficient greedy algorithm to the Lagrangian relaxation subproblems are possible. Hence, our approach is much faster. At the same time, rapid convergence to global optimal solutions is still guaranteed.

Manuscript received July 22, 1998; revised November 24, 1998. This work supported in part by the Texas Advanced Research Program under Grant 003658288 and by a grant from the Intel Corporation. This paper was recommended by Associate Editor K.-T. Cheng.

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

Publisher Item Identifier S 0278-0070(99)05038-1.

Sapatnekar *et al.* [23] later transformed the geometric program into a convex program and solved it by a sophisticated general-purpose convex programming solver based on interior point method. This is the best-known previous algorithm that can guarantee exact transistor sizing solutions. Again, as we explore the special structure of the geometric program, our tailored algorithm is much faster than the algorithm in [23]. For example, the largest test circuit in [23] has 832 transistors and the reported runtime and memory are 9 hours (on a Sun SPARCstation 1) and 11 Mbytes, respectively. Note that for a problem of similar size (864), our approach only needs 1.3 s of runtime (on a PC with a 333-MHz Pentium II processor) and 1.15 Mbytes of memory. According to the SPEC benchmark results,[1] our machine is roughly 40 times faster than the slowest model of Sun SPARCstation 1. Taking the speed difference of the machines into account, our algorithm is about 600 times faster than the general-purpose solver for a small circuit. For larger circuits, we expect the speedup to be even more significant.

The rest of this paper is organized as follows. In Section II, we introduce some notations and terminology that we use in this paper. In Section III, we present our algorithm for the problem of minimizing total area subject to maximum delay bound. In Section IV, we show how to modify our algorithm to minimize maximum delay, to handle arrival time specifications at all inputs and outputs, to consider power dissipation and to use a more accurate gate model. In Section V, experimental results to show the runtime and storage requirements of our algorithm are presented.

II. Preliminaries

In this section, we define some notations and terminology that we use in this paper.

For a general VLSI circuit, we can ignore all latches and optimize its combinational subcircuits. Therefore, we focus on combinational circuits below.

Given a combinational circuit with s input drivers, t output loads, and n gates or wire segments, the gate sizes or the segment widths are allowed to be varied in order to optimize some objective. For $1 \le i \le s$, let R_i^D be the driver resistance of the ith input driver. For $1 \le i \le t$, let C_i^L be the load capacitance of the ith output load. See Fig. 1 for an illustration of a circuit.

A gate, a wire segment, or an input driver is called a *component*. In order to unify the notations that we introduce later, imagine that two factitious components are added to the circuit. The first one is called an output component which consists of all the t output loads. The second one is called an input component which connects to all the s input drivers. Let a *node* be a connection point between two components or the output point of the output component. Note that the output of each component should connect to a distinct node. So it is easy to see that there are $n+s+2$ components and $n+s+2$ nodes.

Let $m = n+s+1$. We label the nodes by indexes $0, \cdots, m$ as follows. The node with index 0 is the output point of the output component. For $1 \le i \le t$, the node with index i is the one connecting to the ith output load. For $t+1 \le i \le n$, the node with index i is a connection point among the gates and wire segments. The indexes are assigned in such a way that if node i and node j are connected to an input and the output of some component, respectively, then $i > j$. For $n+1 \le i \le n+s$, the node with index i is the one connecting to the $(i-n)$th input driver. The node with index m is the output point of the input component. It is not difficult to see that if we view the circuit as a directed acyclic graph, the node index assignment is a reverse topological ordering of the graph. We also label the components by indexes $0, \cdots, m$ such that the output of the component with index i is connected to node i. See Fig. 2 for an illustration of the circuit in Fig. 1 with factitious components, node indexes, and component indexes.

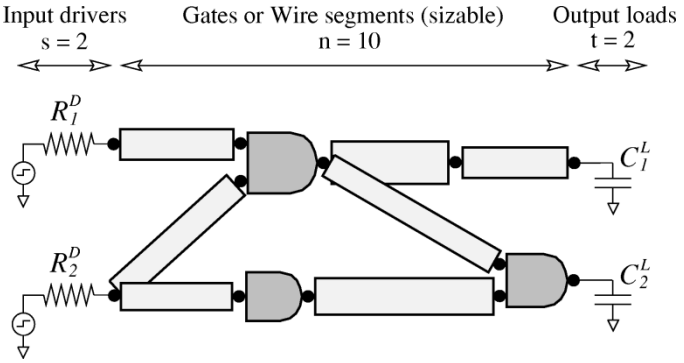

Fig. 1. An illustration of a circuit.

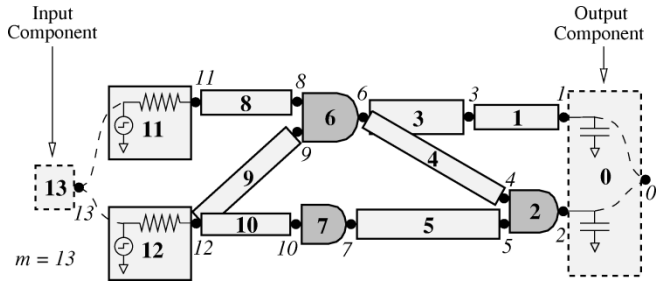

Fig. 2. An illustration of the circuit in Fig. 1 with node indexes and component indexes. The factitious input component and output component are also shown.

For $0 \le i \le m-1$, let input(i) be the set of indexes of components directly connected to the inputs of component i. For $1 \le i \le m$, let output(i) be the set of indexes of components directly connected to the output of component i. For example, for the circuit in Fig. 2, input(0) $= \{1,2\}$, input(6) $= \{8,9\}$, output(6) $= \{3,4\}$, input(8) $= \{11\}$, and output(8) $= \{6\}$. Note that $j \in$ input(i) if and only if $i \in$ output(j).

Let G be the set of component indexes of gates in the circuit. Let W be the set of component indexes of wire segments in the circuit. For the circuit in Fig. 2, $G = \{2,6,7\}$ and $W = \{1,3,4,5,8,9,10\}$.

For the purpose of delay calculation, we model components as resistance-capacitance (*RC*) circuits. If component i is a gate (i.e., $i \in G$), it is modeled as a switch-level *RC* circuit as shown in Fig. 3. See [24] for a reference of this model. Let x_i be the gate size. Then the output resistance $r_i = \hat{r}_i/x_i$, and

[1] SPEC table; ftp://ftp.cdf.toronto.edu/pub/spectable.

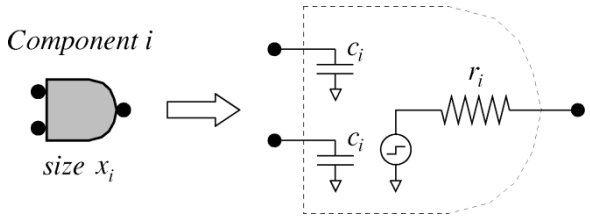

Fig. 3. The model of component i, which is a gate, by a switch-level *RC* circuit. Note that $r_i = \hat{r}_i/x_i$ and $c_i = \hat{c}_i x_i + f_i$, where $\hat{r}_i$, $\hat{c}_i$, and f_i are the unit size output resistance, the unit size gate area capacitance and the gate perimeter capacitance of gate i, respectively. Although the gate shown here is a two-input AND gate, the model can be easily generalized for any gate with any number of input pins.

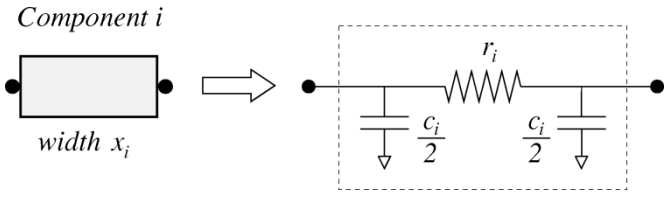

Fig. 4. The model of component i, which is a wire segment, by a π-type *RC* circuit. Note that $r_i = \hat{r}_i/x_i$ and $c_i = \hat{c}_i x_i + f_i$, where $\hat{r}_i$, $\hat{c}_i$, and f_i are the unit width wire resistance, the unit width wire area capacitance, and the wire fringing capacitance of segment i, respectively.

the input capacitance of a pin $c_i = \hat{c}_i x_i + f_i$, where $\hat{r}_i$, $\hat{c}_i$, and f_i are the unit size output resistance, the unit size gate area capacitance and the gate perimeter capacitance of gate i, respectively. (To simplify the notations, we assume the input capacitances of all input pins of a gate are the same. We also ignore the intrinsic gate delay. It is clear that all our results will still hold without these assumptions.)

If component i is a wire segment (i.e., $i \in W$), it is modeled as a π-type *RC* circuit as shown in Fig. 4. Let x_i be the segment width. Then the segment resistance $r_i = \hat{r}_i/x_i$, and the segment capacitance $c_i = \hat{c}_i x_i + f_i$, where $\hat{r}_i$, $\hat{c}_i$, and f_i are the unit width wire resistance, the unit width wire area capacitance and the wire fringing capacitance of segment i, respectively. For $i \in G \cup W$, let L_i and U_i be, respectively, the lower bound and upper bound of the value of x_i, i.e., $L_i \leq x_i \leq U_i$.

Elmore delay model [11] is used for delay calculation. Basically, the Elmore delay along a signal path is the sum of the delays associated with the resistors in the path, where the delay associated with a resistor is equal to its resistance times its downstream capacitance. For our case, each component (except the two factitious components) contains a resistor. We label the resistors by indexes $1, \cdots, n+s$ such that resistor i is the one inside component i. For convenience, for $n+1 \leq i \leq n+s$, let $r_i = R^D_{i-n}$ (i.e., the driver resistance of the $(i-n)$th input driver). So for $1 \leq i \leq n+s$, the resistance of resistor i is r_i. For $1 \leq i \leq n+s$, let C_i be the downstream capacitance of resistor i. Fig. 5 shows the circuit in Fig. 2 after replacing the components by the *RC* models. The resistance of each resistor is marked in the figure. Also, the regions corresponding to the downstream capacitances of resistors 5 and 12 are shaded.

Let $D_i = r_i C_i$ be the delay associated with resistor i. We represent a signal path passing through resistors $i_1, \cdots, i_k$ by the set $p = \{i_1, \cdots, i_k\}$. Let P be the set of all possible paths from node m to node 0 (i.e., from an input driver to an output load). Then for any $p \in P$, the Elmore delay along path p is $\sum_{i \in p} D_i$.

III. Minimizing Total Area Subject to Maximum Delay Bound

In this section, we solve the problem of minimizing the total component area with respect to component sizes $x_1, \cdots, x_n$ subject to the constraint that the maximum delay from any input driver to any output load is at most some constant A_0 (i.e., A_0 is a bound on the arrival time at node 0).

In Section III-A, we first show how to formulate the problem as a constrained optimization problem with a polynomial number of constraints. We call this formulation the primal problem ($\mathcal{PP}$). $\mathcal{PP}$ is a geometric program. There are many standard methods for solving geometric programs [10]. However, because of the special structure of $\mathcal{PP}$, we show that it can be solved very efficiently by Lagrangian relaxation. Lagrangian relaxation is a general technique for solving constrained optimization problems. We outline the basic idea of Lagrangian relaxation below. More details can be found in [1], [13], and [14].

In Lagrangian relaxation, "troublesome" constraints are "relaxed" and incorporated into the objective function after multiplying them by constants called Lagrange multipliers, one multiplier for each constraint. For each fixed vector $\boldsymbol{\lambda}$ of the Lagrange multipliers introduced, we have a new optimization problem (which should be easier to solve because it is free of troublesome constraints) called the Lagrangian relaxation subproblem associated with $\boldsymbol{\lambda}$ ($\mathcal{LRS}/\boldsymbol{\lambda}$). It can be shown that there exists a vector $\boldsymbol{\lambda}$ such that the optimal solution of $\mathcal{LRS}/\boldsymbol{\lambda}$ is also the optimal solution of the original constrained optimization problem $\mathcal{PP}$. The problem of finding such a vector $\boldsymbol{\lambda}$ is called the Lagrangian dual problem ($\mathcal{LDP}$). So if we can solve both $\mathcal{LRS}/\boldsymbol{\lambda}$ and $\mathcal{LDP}$, then the optimal solution of $\mathcal{PP}$ will be given by $\mathcal{LRS}/\boldsymbol{\lambda}$ where $\boldsymbol{\lambda}$ is the optimal solution of $\mathcal{LDP}$.

In Section III-B, we show how $\mathcal{PP}$ is relaxed to obtain the $\mathcal{LRS}/\boldsymbol{\lambda}$ and the corresponding $\mathcal{LDP}$. In Section III-C, we use the Kuhn–Tucker conditions (see [1] for a reference) to derive a set of optimality conditions on $\boldsymbol{\lambda}$. We show that the optimality conditions can be used to greatly simplify $\mathcal{LRS}/\boldsymbol{\lambda}$. We called the simplified version $\mathcal{LRS}/\boldsymbol{\mu}$. In Section III-D, we show how to solve $\mathcal{LRS}/\boldsymbol{\mu}$ (i.e., $\mathcal{LRS}/\boldsymbol{\lambda}$) for any fixed vector $\boldsymbol{\mu}$. In Section III-E, we describe how to solve $\mathcal{LDP}$ by the classical method of subgradient optimization.

A. Problem Formulation

For each i, the area of component i is proportional to its size x_i. Therefore, the total component area can be written as $\sum_{i=1}^{n} \alpha_i x_i$ for some constants $\alpha_1, \cdots, \alpha_n$. Then the problem of minimizing total area subject to maximum delay bound can

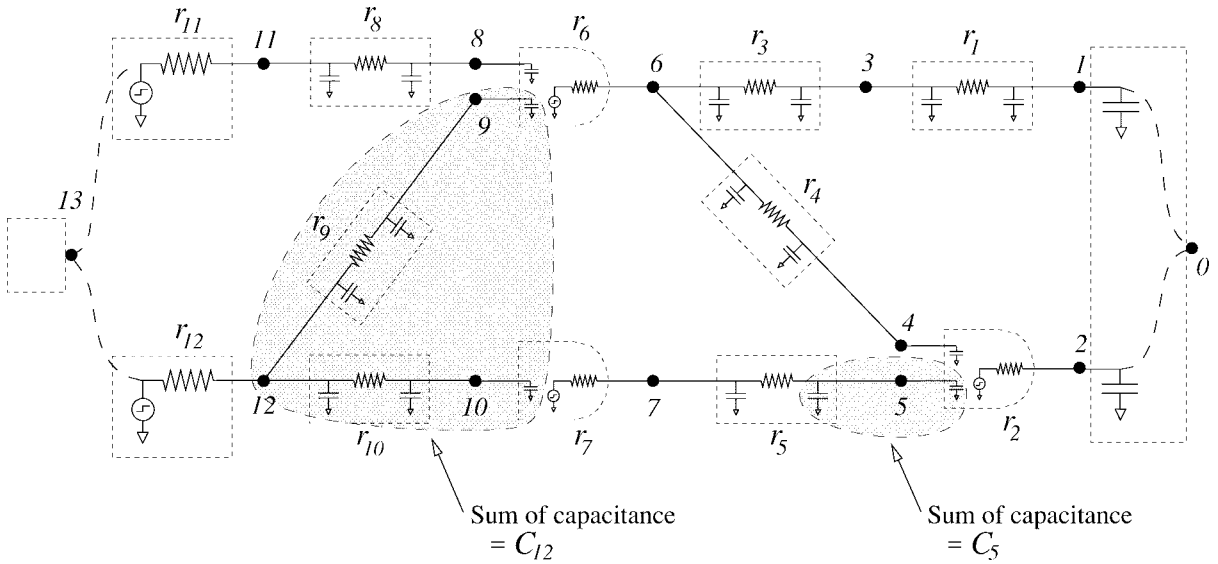

Fig. 5. Illustration of the circuit in Fig. 2 after replacing the gates and wire segments by the *RC* models. The resistance of each resistor is marked in the figure. Also, the regions corresponding to the downstream capacitances of resistors 5 and 12 are shaded.

be formulated directly as

$$\begin{aligned} \text{Minimize} \quad & \sum_{i=1}^{n} \alpha_i x_i \\ \text{Subject to} \quad & \sum_{i \in p} D_i \le A_0 \qquad \forall p \in P \\ & L_i \le x_i \le U_i \qquad i = 1, \cdots, n. \end{aligned}$$

However, the number of possible signal paths from node m to node 0 (and, hence, the number of constraints in the mathematical program above) can be exponential in n. So this direct formulation is impractical.

This difficulty can be handled by the classical technique of partitioning the constraints on path delay into constraints on delay across components. We associate a variable a_i to each node i. a_i represents the arrival time at node i (i.e., the maximum delay from node m to node i). Then it is not difficult to see that the mathematical program below, which we called the primal problem ($\mathcal{PP}$), is equivalent to the mathematical program above:

$\mathcal{PP}$:

Minimize

$$\sum_{i=1}^{n} \alpha_i x_i$$

Subject to

$$\begin{aligned} & a_j \le A_0 && j \in \textit{input}(0) \text{ /* outputs */} \\ & a_j + D_i \le a_i && i = 1, \cdots, n \text{ and } \forall j \in \text{input}(i) \\ & D_i \le a_i && i = n+1, \cdots, n+s \text{ /* inputs */} \\ & L_i \le x_i \le U_i && i = 1, \cdots, n. \end{aligned}$$

Note that the number of constraints in $\mathcal{PP}$ is polynomial in n and s. Also note that for the problem $\mathcal{PP}$, the objective function is a posynomial [10] and the constraints can be rewritten in the form of polynomials. It is well known that under a variable transformation, the problem is convex. So problem $\mathcal{PP}$ has a unique global minimum and no other local minimum. We consider the formulation $\mathcal{PP}$ in the following.

B. Lagrangian Relaxation

We relax all the constraints on arrival time of $\mathcal{PP}$ since they are difficult to handle. The simple constraints on the component sizes $x_1, \cdots, x_n$ are not relaxed. They are handled in the Lagrangian relaxation subproblem.

Following the Lagrangian relaxation procedure, we introduce a nonnegative value called the Lagrange multiplier for each constraint on arrival time. For all $j \in$ input(0) (i.e., $j = 1, \cdots, t$), we introduce λ_{j0} for the constraint $a_j \le A_0$. For $i = 1, \cdots, n$ and for all $j \in$ input(i), we introduce λ_{ji} for the constraint $a_j + D_i \le a_i$. For $i = n+1, \cdots, n+s$, we introduce λ_{mi} for the constraint $D_i \le a_i$. Let $\boldsymbol{\lambda}$ be a vector of all the Lagrange multipliers introduced. Let $\boldsymbol{x} = (x_1, \cdots, x_n)$ and $\boldsymbol{a} = (a_1, \cdots, a_{n+s})$. Let

$$\begin{aligned} L_\lambda(\boldsymbol{x}, \boldsymbol{a}) = & \sum_{i=1}^{n} \alpha_i x_i + \sum_{j \in \text{input}(0)} \lambda_{j0}(a_j - A_0) \\ & + \sum_{i=1}^{n} \sum_{j \in \text{input}(i)} \lambda_{ji}(a_j + D_i - a_i) \\ & + \sum_{i=n+1}^{n+s} \lambda_{mi}(D_i - a_i). \end{aligned}$$

Then the Lagrangian relaxation subproblem associated with the Lagrange multipliers $\boldsymbol{\lambda}$ is

$$\begin{aligned} \mathcal{LRS}/\boldsymbol{\lambda}\text{:} \quad \text{Minimize} \quad & L_\lambda(\boldsymbol{x}, \boldsymbol{a}) \\ \text{Subject to} \quad & L_i \le x_i \le U_i \qquad i = 1, \cdots, n. \end{aligned}$$

Let the function $Q(\boldsymbol{\lambda})$ be the optimal value of the problem $\mathcal{LRS}/\boldsymbol{\lambda}$. We define the Lagrangian dual problem as follows:

$$\begin{aligned} \mathcal{LDP}\text{:} \quad \text{Maximize} \quad & Q(\boldsymbol{\lambda}) \\ \text{Subject to} \quad & \boldsymbol{\lambda} \ge \boldsymbol{0}. \end{aligned}$$

As said in Section III-A, $\mathcal{PP}$ can be transformed into a convex problem. So [2, Theorem 6.2.4] implies that if $\boldsymbol{\lambda}$ is the optimal solution of $\mathcal{LDP}$, then the optimal solution of $\mathcal{LRS}/\boldsymbol{\lambda}$ will also optimize $\mathcal{PP}$.

C. Simplification of $\mathcal{LRS}/\boldsymbol{\lambda}$

Here, we use the Kuhn–Tucker conditions to derive a set of optimality conditions on $\boldsymbol{\lambda}$. Then we show that the optimality conditions can be used to greatly simplify $\mathcal{LRS}/\boldsymbol{\lambda}$.

For $1 \le i \le n$, let u_i and v_i be the Lagrange multipliers for the constraints $L_i \le x_i$ and $x_i \le U_i$, respectively. Consider the Lagrangian [1] $\mathcal{L}$ of $\mathcal{PP}$

$$\begin{aligned}
\mathcal{L} &= \sum_{i=1}^{n} \alpha_i x_i + \sum_{j\in\text{input}(0)} \lambda_{j0}(a_j - A_0) \\
&\quad + \sum_{i=1}^{n} \sum_{j\in\text{input}(i)} \lambda_{ji}(a_j + D_i - a_i) \\
&\quad + \sum_{i=n+1}^{n+s} \lambda_{mi}(D_i - a_i) + \sum_{i=1}^{n} u_i(L_i - x_i) \\
&\quad + \sum_{i=1}^{n} v_i(x_i - U_i) \\
&= \sum_{k\in\text{input}(0)} \lambda_{k0} a_k - \sum_{j\in\text{input}(0)} \lambda_{j0} A_0 \\
&\quad + \sum_{k=t+1}^{n+s} \sum_{i:\, k\in\text{input}(i)} \lambda_{ki} a_k + \sum_{i=1}^{n} \sum_{j\in\text{input}(i)} \lambda_{ji} D_i \\
&\quad - \sum_{k=1}^{n} \sum_{j\in\text{input}(k)} \lambda_{jk} a_k + \sum_{i=n+1}^{n+s} \lambda_{mi} D_i \\
&\quad - \sum_{k=n+1}^{n+s} \lambda_{mk} a_k + \sum_{i=1}^{n} \alpha_i x_i + \sum_{i=1}^{n} u_i(L_i - x_i) \\
&\quad + \sum_{i=1}^{n} v_i(x_i - U_i) \\
&= \sum_{k=1}^{t} \left(\lambda_{k0} - \sum_{j\in\text{input}(k)} \lambda_{jk} \right) a_k \\
&\quad + \sum_{k=t+1}^{n} \left(\sum_{i:\, k\in\text{input}(i)} \lambda_{ki} - \sum_{j\in\text{input}(k)} \lambda_{jk} \right) a_k \\
&\quad + \sum_{k=n+1}^{n+s} \left(\sum_{i:\, k\in\text{input}(i)} \lambda_{ki} - \lambda_{mk} \right) a_k \\
&\quad + \sum_{i=1}^{n} \left(\sum_{j\in\text{input}(i)} \lambda_{ji} \right) D_i - \sum_{j\in\text{input}(0)} \lambda_{j0} A_0 \\
&\quad + \sum_{i=n+1}^{n+s} \lambda_{mi} D_i + \sum_{i=1}^{n} \alpha_i x_i + \sum_{i=1}^{n} u_i(L_i - x_i) \\
&\quad + \sum_{i=1}^{n} v_i(x_i - U_i) \\
&= \sum_{k=1}^{n+s} \left(\sum_{i\in\text{output}(k)} \lambda_{ki} - \sum_{j\in\text{input}(k)} \lambda_{jk} \right) a_k \\
&\quad + \sum_{i=1}^{n} \left(\sum_{j\in\text{input}(i)} \lambda_{ji} \right) D_i - \sum_{j\in\text{input}(0)} \lambda_{j0} A_0 \\
&\quad + \sum_{i=n+1}^{n+s} \lambda_{mi} D_i + \sum_{i=1}^{n} \alpha_i x_i + \sum_{i=1}^{n} u_i(L_i - x_i) \\
&\quad + \sum_{i=1}^{n} v_i(x_i - U_i).
\end{aligned}$$

The Kuhn–Tucker conditions imply $\partial\mathcal{L}/\partial a_k = 0$ for $1 \le k \le n+s$ at the optimal solution of $\mathcal{PP}$. In other words, the Lagrange multipliers corresponding to the optimal solution of $\mathcal{PP}$ must satisfy the conditions $\partial\mathcal{L}/\partial a_k = 0$ for $1 \le k \le n+s$. So we can consider those Lagrange multipliers only. By setting $\partial\mathcal{L}/\partial a_k = 0$, we obtain the following conditions on $\boldsymbol{\lambda}$.

Optimality Conditions on $\boldsymbol{\lambda}$:

$$\sum_{i\in\text{output}(k)} \lambda_{ki} = \sum_{j\in\text{input}(k)} \lambda_{jk} \qquad \text{for } 1 \le k \le n+s.$$

We show in Lemma 1 below how the optimality conditions on $\boldsymbol{\lambda}$ can be used to simplify $\mathcal{LRS}/\boldsymbol{\lambda}$. Let $\Omega_\lambda = \{\boldsymbol{\lambda} \ge \mathbf{0}$: $\boldsymbol{\lambda}$ satisfies the optimality conditions on $\boldsymbol{\lambda}\}$.

Lemma 1: For any $\boldsymbol{\lambda} \in \Omega_\lambda$, solving $\mathcal{LRS}/\boldsymbol{\lambda}$ is equivalent to solving

$$\begin{aligned}
\mathcal{LRS}/\boldsymbol{\mu}\text{:} \quad & \text{Minimize} \quad L_\mu(\boldsymbol{x}) \\
& \text{Subject to} \quad L_i \le x_i \le U_i \qquad i = 1, \cdots, n
\end{aligned}$$

where $\boldsymbol{\mu} = (\mu_0, \cdots, \mu_{n+s})$, $\mu_i = \sum_{j\in\text{input}(i)} \lambda_{ji}$ for $0 \le i \le n+s$, and $L_\mu(\boldsymbol{x}) = \sum_{i=1}^{n+s} \mu_i D_i + \sum_{i=1}^{n} \alpha_i x_i$.

Proof: By rearranging the terms, $L_\lambda(\boldsymbol{x}, \boldsymbol{a})$ can be rewritten as follows:

$$\begin{aligned}
L_\lambda(\boldsymbol{x}, \boldsymbol{a}) &= \sum_{k=1}^{n+s} \left(\sum_{i\in\text{output}(k)} \lambda_{ki} - \sum_{j\in\text{input}(k)} \lambda_{jk} \right) a_k \\
&\quad + \sum_{i=1}^{n} \alpha_i x_i + \sum_{i=1}^{n} \left(\sum_{j\in\text{input}(i)} \lambda_{ji} \right) D_i \\
&\quad - \sum_{j\in\text{input}(0)} \lambda_{j0} A_0 + \sum_{i=n+1}^{n+s} \lambda_{mi} D_i.
\end{aligned}$$

So by substituting the optimality conditions on $\boldsymbol{\lambda}$ into $L_\lambda(\boldsymbol{x}, \boldsymbol{a})$, we get

$$\begin{aligned}
L_\lambda(\boldsymbol{x}, \boldsymbol{a}) &= \sum_{i=1}^{n} \alpha_i x_i + \sum_{i=1}^{n} \left(\sum_{j\in\text{input}(i)} \lambda_{ji} \right) D_i \\
&\quad - \sum_{j\in\text{input}(0)} \lambda_{j0} A_0 + \sum_{i=n+1}^{n+s} \lambda_{mi} D_i \\
&= \sum_{i=1}^{n+s} \mu_i D_i + \sum_{i=1}^{n} \alpha_i x_i - \mu_0 A_0. \qquad (1)
\end{aligned}$$

Note that $L_\lambda(\boldsymbol{x}, \boldsymbol{a})$ in (1) no longer depends on $\boldsymbol{a}$. Also note that $\mu_0 A_0$ is a constant. So if let $L_\mu(\boldsymbol{x}) = \sum_{i=1}^{n+s} \mu_i D_i + \sum_{i=1}^{n} \alpha_i x_i$, then minimizing L_μ is the same as minimizing L_λ. After finding the optimal $\boldsymbol{x}$, the optimal $\boldsymbol{a}$ can be found by considering, one by one, the variables a_i's in the order of decreasing i. For each a_i, we set it to the smallest possible value that satisfies the constraints of $\mathcal{PP}$. Hence, the lemma follows. □

D. Solving $\mathcal{LRS}/\boldsymbol{\mu}$

In this subsection, for any fixed $\boldsymbol{\mu} \geq \mathbf{0}$, we show how to solve $\mathcal{LRS}/\boldsymbol{\mu}$ optimally by a greedy algorithm based on iteratively resizing the gates and wire segments. A similar technique has been successfully applied to some other wire or buffer sizing problems before (e.g., [3] and [9]). Chu and Wong [5] proved that for wire sizing of interconnect trees, the greedy algorithm runs in time linear to the number of segments.

If we resize component i (i.e., changing x_i) while keeping the sizes of all the other components fixed, we say that it is a local resizing of component i. An optimal local resizing of component i is a local resizing that minimize $L_\mu(\boldsymbol{x})$.

For $1 \leq i \leq n$, let upstream(i) be the set of resistor indexes (excluding i) on the path(s) from component i to the nearest upstream gate(s) or input driver(s). For example, for the circuit in Fig. 5, upstream(1) = {3, 6} and upstream(6) = {8, 9, 11, 12}. Let $R_i = \sum_{j \in \text{upstream}(i)} \mu_j r_j$ (i.e., R_i is a weighted upstream resistance of component i). For $i \in W$, let $C_i' = C_i - \hat{c}_i x_i/2$, and for $i \in G$ or for $n+1 \leq i \leq n+s$, let $C_i' = C_i$. Note that for $1 \leq i \leq n+s$, C_i' is independent of x_i.

Lemma 2: For $1 \leq i \leq n$, $L_\mu(\boldsymbol{x})$ can be written in the following form:

$$L_\mu(\boldsymbol{x}) = A_i(\boldsymbol{x}) x_i + \frac{B_i(\boldsymbol{x})}{x_i} + E_i(\boldsymbol{x})$$

where $A_i(\boldsymbol{x})$, $B_i(\boldsymbol{x})$, and $E_i(\boldsymbol{x})$ are independent of x_i, $A_i(\boldsymbol{x}) = \hat{c}_i R_i + \alpha_i$, and $B_i(\boldsymbol{x}) = \mu_i \hat{r}_i C_i'$.

Proof:

$$\begin{aligned} L_\mu(\boldsymbol{x}) &= \sum_{i=1}^{n+s} \mu_i r_i C_i + \sum_{i=1}^{n} \alpha_i x_i \\ &= \sum_{i \in G} \mu_i r_i C_i' + \sum_{i \in W} \mu_i r_i \left(C_i' + \frac{\hat{c}_i x_i}{2} \right) \\ &\quad + \sum_{i=n+1}^{n+s} \mu_i r_i C_i' + \sum_{i=1}^{n} \alpha_i x_i \\ &= \sum_{i=1}^{n+s} \mu_i r_i C_i' + \sum_{i \in W} \frac{\mu_i \hat{r}_i \hat{c}_i}{2} + \sum_{i=1}^{n} \alpha_i x_i. \end{aligned}$$

For any i between 1 and n, $\mu_i r_i C_i' = \mu_i \hat{r}_i C_i' / x_i$. For any $j \neq i$, if $j \notin$ upstream(i), then $\mu_j r_j C_j'$ is independent of x_i. If $j \in$ upstream(i), then $\mu_j r_j C_j' = \mu_j r_j \hat{c}_i x_i +$ terms independent of x_i. So

$$\begin{aligned} L_\mu(\boldsymbol{x}) &= \left(\sum_{j \in \text{upstream}(i)} \mu_j r_j \hat{c}_i + \alpha_i \right) x_i \\ &\quad + \frac{\mu_i \hat{r}_i C_i'}{x_i} + \text{terms independent of } x_i \\ &= (\hat{c}_i R_i + \alpha_i) x_i + \frac{\mu_i \hat{r}_i C_i'}{x_i} \\ &\quad + \text{terms independent of } x_i. \end{aligned}$$

Hence the lemma follows. □

Lemma 3: Let $\tilde{\boldsymbol{x}} = (\tilde{x}_1, \cdots, \tilde{x}_n)$ be a component-sizing solution. An optimal local resizing of component i is given by changing the size of component i to

$$x_i^* = \min \left(U_i, \max \left(L_i, \sqrt{\frac{B_i(\tilde{\boldsymbol{x}})}{A_i(\tilde{\boldsymbol{x}})}} \right) \right).$$

Proof: If we fix the size of component j to $\tilde{x}_j$ for all $j \neq i$, and we change x_i, we can view L_μ as a function of x_i. By Lemma 2, it is given by

$$F(x_i) = A_i(\tilde{\boldsymbol{x}}) x_i + \frac{B_i(\tilde{\boldsymbol{x}})}{x_i} + E_i(\tilde{\boldsymbol{x}}).$$

Differentiating F with respect to x_i, we get

$$\frac{dF}{dx_i} = A_i(\tilde{\boldsymbol{x}}) - \frac{B_i(\tilde{\boldsymbol{x}})}{x_i^2}.$$

Let $\theta(\tilde{\boldsymbol{x}}) = \sqrt{B_i(\tilde{\boldsymbol{x}})/A_i(\tilde{\boldsymbol{x}})}$. Note that

$$\begin{aligned} dF/dx_i &= 0 \quad \text{if } x_i = \theta(\tilde{\boldsymbol{x}}) \\ dF/dx_i &< 0 \quad \text{if } x_i < \theta(\tilde{\boldsymbol{x}}) \\ dF/dx_i &> 0 \quad \text{if } x_i > \theta(\tilde{\boldsymbol{x}}). \end{aligned}$$

Hence $F(x_i)$ is decreasing when $x_i < \theta(\tilde{\boldsymbol{x}})$, $F(x_i)$ is increasing when $x_i > \theta(\tilde{\boldsymbol{x}})$, and $F(x_i)$ is minimum at $x_i = \theta(\tilde{\boldsymbol{x}})$. If x_i is constrained to the range $[L_i, U_i]$, we consider three cases:

Case 1—$\theta(\tilde{\boldsymbol{x}}) \in [L_i, U_i]$: In this case, $F(x_i)$ is minimized when $x_i = \theta(\tilde{\boldsymbol{x}})$.

Case 2—$\theta(\tilde{\boldsymbol{x}}) > U_i$: Then $F(x_i)$ is decreasing in $[L_i, U_i]$. So $F(x_i)$ is minimized when $x_i = U_i$.

Case 3—$\theta(\tilde{\boldsymbol{x}}) < L_i$: Then $F(x_i)$ is increasing in $[L_i, U_i]$. So $F(x_i)$ is minimized when $x_i = L_i$.

Hence the lemma follows. □

$\mathcal{LRS}/\boldsymbol{\mu}$ can be solved by a greedy algorithm based on iteratively resizing the components. In each iteration, the components are examined one at a time; each time a component is resized optimally using Lemma 3 while keeping the sizes of the other components fixed. We call the algorithm SOLVE_LRS/$\boldsymbol{\mu}$ and it is described below. Note that in order to use Lemma 3 to resize component i, we need to compute C_i' and R_i first. Our algorithm SOLVE_LRS/$\boldsymbol{\mu}$ computes C_i''s and R_i's incrementally by traversing the circuit in a reverse topological order (Step 2) and in a topological order (Step 3), respectively. So it is not difficult to see that each iteration of

the algorithm takes only $O(n)$ time.

ALGORITHM SOLVE_LRS/$\boldsymbol{\mu}$:

Output: $\boldsymbol{x} = (x_1, \cdots, x_n)$ which minimizes $L_\mu(\boldsymbol{x})$

1. **for** $i := 1$ **to** n **do** $x_i := L_i$
2. /* Finding C_i' for $1 \le i \le n$ by traversing the circuit in a reverse topological order */

   for $i := 1$ **to** t **do**

$$C_i' := \begin{cases} C_i^L & \text{if } i \in G \\ C_i^L + f_i/2 & \text{if } i \in W \end{cases}$$

   for $i := t+1$ **to** n **do**

$$C_i' := \begin{cases} 0 & \text{if } i \in G \\ f_i/2 & \text{if } i \in W \end{cases}$$

   for all k s.t. $i \in \text{input}(k)$ **do**

$$C_i' := \begin{cases} C_i' + \hat{c}_k x_k + f_k & \text{if } k \in G \\ C_i' + \hat{c}_k x_k + f_k/2 + C_k' & \text{if } k \in W \end{cases}$$

3. /* Finding R_i and x_i for $1 \le i \le n$ by traversing the circuit in a topological order */

   for $i := n$ **downto** 1 **do**

   $R_i := 0$

   for all $j \in \text{input}(i)$ **do**

$$R_i := \begin{cases} R_i + \mu_j R_{j-n}^D & \text{if } n+1 \le j \le n+s \\ R_i + \mu_j \hat{r}_j / x_j & \text{if } j \in G \\ R_i + \mu_j \hat{r}_j / x_j + R_j & \text{if } j \in W \end{cases}$$

$$x_i := \min\left(U_i, \max\left(L_i, \sqrt{\frac{(\mu_i \hat{r}_i C_i')}{(\hat{c}_i R_i + \alpha_i)}}\right)\right)$$

4. Repeat Step 2 and 3 until no improvement.

Note that $L_\mu(\boldsymbol{x})$ is a posynomial [10] in $\boldsymbol{x}$. It is well known that under a variable transformation, a posynomial is equivalent to a convex function. So $L_\mu(\boldsymbol{x})$ has a unique global minimum and no other local minimum. We show in the following that algorithm SOLVE_LRS/$\boldsymbol{\mu}$ always converges to the global minimum.

Lemma 4: If algorithm SOLVE_LRS/$\boldsymbol{\mu}$ converges, then the solution is optimal to $\mathcal{LRS}/\boldsymbol{\mu}$.

Proof: Suppose the algorithm converges to $\boldsymbol{x}^* = (x_1^*, \cdots, x_n^*)$. Then for $1 \le i \le n$, by Lemma 3, $x_i^* = \min(U_i, \max(L_i, \sqrt{B_i(\boldsymbol{x}^*)/A_i(\boldsymbol{x}^*)}))$. Note that $L_\mu(\boldsymbol{x})$ is a posynomial in $\boldsymbol{x}$, and that under the transformation $x_i = e^{z_i}$ for $1 \le i \le n$, the function $H(\boldsymbol{z}) = L_\mu(e^{z_1}, \cdots, e^{z_n})$ is convex over $\Omega_z = \{\boldsymbol{z}: L_i \le e^{z_i} \le U_i, 1 \le i \le n\}$. Let $\boldsymbol{z}^* = (z_1^*, \cdots, z_n^*)$ where $x_i^* = e^{z_i^*}$ for $1 \le i \le n$. We now consider three cases.

Case 1—$x_i^* = \sqrt{(B_i(\boldsymbol{x}^*))/(A_i(\boldsymbol{x}^*))}$: In this case, we have $\partial L_\mu / \partial x_i(\boldsymbol{x}^*) = 0$. Thus

$$\frac{\partial H}{\partial z_i}(\boldsymbol{z}^*) = \frac{\partial L_\mu}{\partial x_i}(\boldsymbol{x}^*)\frac{\partial x_i}{\partial z_i}(\boldsymbol{z}^*) = \frac{\partial L_\mu}{\partial x_i}(\boldsymbol{x}^*)e^{z_i^*} = 0.$$

Case 2—$x_i^* = L_i$: In this case, $L_i \ge \sqrt{(B_i(\boldsymbol{x}^*))/(A_i(\boldsymbol{x}^*))}$. We have $\partial L_\mu / \partial x_i(\boldsymbol{x}^*) \ge 0$ and $z_i - z_i^* \ge 0$, $\forall \boldsymbol{z} \in \Omega_z$. Hence

$$\frac{\partial H}{\partial z_i}(\boldsymbol{z}^*)(z_i - z_i^*) = \frac{\partial L_\mu}{\partial x_i}(\boldsymbol{x}^*)e^{z_i^*}(z_i - z_i^*) \ge 0, \qquad \forall \boldsymbol{z} \in \Omega_z.$$

Case 3—$x_i^* = U_i$: In this case, $U_i \le \sqrt{(B_i(\boldsymbol{x}^*))/(A_i(\boldsymbol{x}^*))}$. We have $\partial L_\mu / \partial x_i(\boldsymbol{x}^*) \le 0$ and $z_i - z_i^* \le 0$, $\forall \boldsymbol{z} \in \Omega_z$. Hence

$$\frac{\partial H}{\partial z_i}(\boldsymbol{z}^*)(z_i - z_i^*) = \frac{\partial L_\mu}{\partial x_i}(\boldsymbol{x}^*)e^{z_i^*}(z_i - z_i^*) \ge 0, \qquad \forall \boldsymbol{z} \in \Omega_z.$$

So $\partial H / \partial z_i(\boldsymbol{z}^*)(z_i - z_i^*) \ge 0$ for all i and for all $\boldsymbol{z} \in \Omega_z$. Thus for any feasible solution $\boldsymbol{x}$,

$$\begin{aligned} L_\mu(\boldsymbol{x}) - L_\mu(\boldsymbol{x}^*) &= H(\boldsymbol{z}) - H(\boldsymbol{z}^*) \\ &\ge \nabla H(\boldsymbol{z}^*)(\boldsymbol{z} - \boldsymbol{z}^*) \qquad \text{as } H \text{ is convex} \\ &= \sum_{i=1}^{n} \frac{\partial H}{\partial z_i}(\boldsymbol{z}^*)(z_i - z_i^*) \\ &\ge 0. \end{aligned}$$

Therefore, $\boldsymbol{x}^*$ is the global minimum point. □

Lemma 5: The algorithm SOLVE_LRS/$\boldsymbol{\mu}$ always converges.

Proof: For any two vectors $\boldsymbol{x}$ and $\boldsymbol{x}'$, we use $\boldsymbol{x} \le \boldsymbol{x}'$ to denote that $x_i \le x_i'$ for all i. Let $\boldsymbol{x}^*$ be the optimal solution, $\boldsymbol{x}$ be a feasible solution, and $\boldsymbol{x}'$ be the solution after locally resizing a component of $\boldsymbol{x}$. If $\boldsymbol{x} \le \boldsymbol{x}^*$, then we can prove that $\boldsymbol{x} \le \boldsymbol{x}' \le \boldsymbol{x}^*$ (this is similar to the dominance property in [7]).

In Step 1 of algorithm SOLVE_LRS/$\boldsymbol{\mu}$, we set $x_i = L_i$ for all i initially. So we know that for all i, x_i is nondecreasing for each local resizing, and is upper bounded by x_i^*. Hence, the algorithm SOLVE_LRS/$\boldsymbol{\mu}$ converges. □

By Lemmas 4 and 5, we have the following theorem.

Theorem 1: For any fixed vector $\boldsymbol{\mu} \ge \mathbf{0}$, algorithm SOLVE_LRS/$\boldsymbol{\mu}$ always converges to the optimal component-sizing solution of the problem $\mathcal{LRS}/\boldsymbol{\mu}$.

Algorithm SOLVE_LRS/$\boldsymbol{\mu}$ runs in $O(rn)$ time using $O(n)$ storage, where n is the number of components and r is the number of iterations. We observe that the number of iterations r is constant (i.e., the run time of SOLVE_LRS/$\boldsymbol{\mu}$ is linear) in practice.

E. Solving $\mathcal{LDP}$

As we point out in Section III-C, instead of considering all $\boldsymbol{\lambda} \ge \mathbf{0}$, we can focus on those $\boldsymbol{\lambda} \in \Omega_\lambda$. So $\mathcal{LDP}$ can be redefined as below:

$$\mathcal{LDP}: \quad \begin{aligned} &\text{Maximize} && Q(\boldsymbol{\lambda}) \\ &\text{Subject to} && \boldsymbol{\lambda} \in \Omega_\lambda \end{aligned}$$

where $Q(\boldsymbol{\lambda})$ is the optimal value of $\mathcal{LRS}/\boldsymbol{\lambda}$.

By [2, Theorem 6.3.1], $Q(\boldsymbol{\lambda})$ is a concave function over $\boldsymbol{\lambda} \ge \mathbf{0}$. However, $Q(\boldsymbol{\lambda})$ is not differentiable in general. So methods like steepest descent, which depends on the gradient directions, are not applicable. The subgradient optimization method is usually used instead. The subgradient optimization method can be viewed as a generalization of the steepest

descent method in which the gradient direction is substituted by a subgradient-based direction (see [1] for a reference).

Basically, starting from an arbitrary point $\boldsymbol{\lambda}$, the method iteratively moves from the current point to a new point following the subgradient direction. At Step k, we first solve $\mathcal{LRS}/\boldsymbol{\lambda}$ (by solving the simpler $\mathcal{LRS}/\boldsymbol{\mu}$). Then for each relaxed constraint, we define the subgradient to be the right hand side minus the left hand side of the constraint, evaluated at the current solution. The subgradient direction is the vector of all the subgradients. We move to a new point by multiplying a step size ρ_k to the subgradient direction and adding it to $\boldsymbol{\lambda}$. After each time we moved, we project $\boldsymbol{\lambda}$ back to the nearest point in Ω_λ so that we can solve $\mathcal{LRS}/\boldsymbol{\mu}$ instead of $\mathcal{LRS}/\boldsymbol{\lambda}$ for the next iteration. The procedure is repeated until it converges.

It is well known (see [2, Theorem 8.9.2] for example) that if the step size sequence $\{\rho_k\}$ satisfies the conditions $\lim_{k\to\infty}\rho_k = 0$ and $\sum_{k=1}^{\infty}\rho_k = \infty$, then the subgradient optimization method will always converge to the optimal solution.

The description is summarized in the algorithm SOLVE_LDP below.

ALGORITHM SOLVE_LDP :

Output: $\boldsymbol{\lambda}$ which maximizes $\mathcal{LRS}/\boldsymbol{\lambda}$

1. $k := 1$ /* step counter */
   $\boldsymbol{\lambda}$:= arbitrary initial vector in Ω_λ
2. Let $\boldsymbol{\mu} = (\mu_0, \cdots, \mu_{n+s})$ where $\mu_i = \sum_{j\in \text{input}(i)} \lambda_{ji}$.
   Solve $\mathcal{LRS}/\boldsymbol{\lambda}$ by calling SOLVE_LRS/$\boldsymbol{\mu}$ to solve $\mathcal{LRS}/\boldsymbol{\mu}$ and calculating $a_1, \cdots, a_{n+s}$ as described in the proof of Lemma 1.
3. /* Move to a new $\boldsymbol{\lambda}$ by adjusting the Lagrange multipliers λ_{ji} */
   /**for** $i := 0$ **to** $n+s$ **do**
   for all $j \in$ input(i) **do**

$$\lambda_{ji} := \begin{cases} \lambda_{ji} + \rho_k(a_j - A_0) & \text{if } i = 0 \\ \lambda_{ji} + \rho_k(a_j + D_i - a_i) & \text{if } 1 \le i \le n \\ \lambda_{ji} + \rho_k(D_i - a_i) & \text{if } n+1 \le i \le n+s \end{cases}$$

4. Project $\boldsymbol{\lambda}$ onto the nearest point in Ω_λ.
5. $k := k + 1$
6. Repeat Step 2–5 until

$$\left(\sum_{i=1}^{n}\alpha_i x_i - Q(\boldsymbol{\lambda})\right) \le \text{ error bound.}$$

Theorem 2: The algorithm SOLVE_LDP always converges to the optimal solution of $\mathcal{LDP}$.

We conclude Section III by giving the algorithm simultaneous gate and wire sizing by Lagrangian relaxation (SGWS-LR) below.

ALGORITHM SGWS-LR:

Output: the optimal gate and wire sizing solution $\boldsymbol{x}$

1. Call SOLVE_LDP to find the optimal $\boldsymbol{\lambda}$.
2. Let $\boldsymbol{\mu} = (\mu_0, \cdots, \mu_{n+s})$ where $\mu_i = \sum_{j\in \text{input}(i)} \lambda_{ji}$.
3. Call SOLVE_LRS/$\boldsymbol{\mu}$ to find the optimal $\boldsymbol{x}$.

Theorem 3: For simultaneous gate and wire sizing, the problem of minimizing total area subject to maximum delay bound can be solved optimally by SGWS-LR.

IV. Extensions

In Section III, the objective of our problem is the total component area and the constraint is on the maximum delay from any input to any output (i.e., the arrival time at node 0). In this section, we extend our Lagrangian relaxation approach to handle problems with other objectives and with other constraints. In Section IV-A, we treat the maximum delay as the objective and show how to minimize it. We also point out that the problem of minimizing maximum delay subject to total area bound is easy to handle. In Section IV-B, instead of assuming that all the input signals arrive at time 0 and all the output signals have a single bound on the arrival time, we allow different arrival time specifications on the input and output signals. In Section IV-C, we show how power dissipation can be handled. In Section IV-D, we show that a more accurate gate model can be used.

For all the extensions, only slight modifications to our algorithm presented in Section III are needed. Moreover, convergence to global optimum solutions is still guaranteed. Actually, it is not difficult to see that any combination of the problem in Section III or the extensions can be handled similarly. For example, we can optimally solve the problem of minimizing power subject to bounds on area and on maximum delay from any input to any output.

A. Minimizing Maximum Delay

Instead of having a constant bound A_0 for the arrival time at node 0, we introduce one more variable a_0 to represent the arrival time at node 0, and we want to minimize a_0. As in Section III-A, by partitioning the constraints on path delay into constraints on delay across components, the problem of minimizing maximum delay by simultaneous gate and wire sizing can be formulated as

$\mathcal{PP}$:

Minimize

a_0

Subject to

$$\begin{aligned} & a_j \le a_0 && j \in \textit{input}(0) \text{ /* outputs */} \\ & a_j + D_i \le a_i && i = 1, \cdots, n \text{ and } \forall\, j \in \text{input}(i) \\ & D_i \le a_i && i = n+1, \cdots, n+s \text{ /* inputs */} \\ & L_i \le x_i \le U_i && i = 1, \cdots, n. \end{aligned}$$

If all the constraints on arrival time are relaxed, then the Lagrangian relaxation subproblem associated with the Lagrange multipliers $\boldsymbol{\lambda}$ will be:

$$\mathcal{LRS}/\boldsymbol{\lambda}: \quad \text{Minimize} \quad L_\lambda(\boldsymbol{x}, \boldsymbol{a}) \qquad \text{Subject to} \quad L_i \le x_i \le U_i \qquad i = 1, \cdots, n$$

where

$$L_\lambda(\boldsymbol{x}, \boldsymbol{a}) = a_0 + \sum_{j \in \text{input}(0)} \lambda_{j0}(a_j - a_0) + \sum_{i=1}^{n} \sum_{j \in \text{input}(i)} \lambda_{ji}(a_j + D_i - a_i) + \sum_{i=n+1}^{n+s} \lambda_{mi}(D_i - a_i).$$

As before, by the Kuhn–Tucker conditions, we have the following optimality conditions on $\boldsymbol{\lambda}$:

$$1 = \sum_{j \in \text{input}(0)} \lambda_{j0}$$

$$\sum_{i \in \text{output}(k)} \lambda_{ki} = \sum_{j \in \text{input}(k)} \lambda_{jk} \qquad \text{for } 1 \le k \le n+s.$$

Then for $\boldsymbol{\lambda}$ satisfying the conditions, $\mathcal{LRS}/\boldsymbol{\lambda}$ can be simplified to

$$\mathcal{LRS}/\boldsymbol{\mu}: \quad \text{Minimize} \quad L_\mu(\boldsymbol{x}) \qquad \text{Subject to} \quad L_i \le x_i \le U_i \qquad i = 1, \cdots, n$$

where $\boldsymbol{\mu} = (\mu_0, \cdots, \mu_{n+s})$, $\mu_i = \sum_{j \in \text{input}(i)} \lambda_{ji}$ for $0 \le i \le n+s$, and $L_\mu(\boldsymbol{x}) = \sum_{i=1}^{n+s} \mu_i D_i$.

It is easy to see that $\mathcal{LRS}/\boldsymbol{\mu}$ can be solved optimally by the iterative local resizing algorithm in Section III-D and the corresponding $\mathcal{LDP}$ can be solved optimally by the subgradient optimization method as described in Section III-V. Therefore the problem of minimizing maximum delay can also be solved optimally by our approach.

In fact, the problem of minimizing maximum delay subject to area bound can also be optimally solved by the Lagrangian relaxation approach. The constraint on area can be relaxed and incorporated into the objective function as well. The function $L_\lambda(\boldsymbol{x}, \boldsymbol{a})$ is of the same form as the one in Section III-B.

B. Arrival Time Specifications on Inputs and Outputs

In Section III, we assume that all the input signals arrive at time 0 and we want to bound the arrival time at the outputs uniformly by a single constant A_0. We show in this subsection that different arrival time specifications on the input and output signals can be easily handled. We demonstrate the idea by considering the problem of minimizing total area subject to different arrival time constraints at inputs and outputs.

For $n+1 \le i \le n+s$, let A_i be the arrival time specification of the input signal at the $(i-n)$th input driver. For $1 \le j \le t$, let A_j be the arrival time requirement on the output signal at the jth output load. Then the problem can be formulated as follows:

$\mathcal{PP}$:

Minimize

$$\sum_{i=1}^{n} \alpha_i x_i$$

Subject to

$$\begin{aligned} &a_j \le A_j && j \in \textit{input}(0) \ /^* \text{ outputs } ^*/ \\ &a_j + D_i \le a_i && i = 1, \cdots, n \text{ and } \forall\, j \in \textit{input}(i) \\ &A_i + D_i \le a_i && i = n+1, \cdots, n+s \ /^* \text{ inputs } ^*/ \\ &L_i \le x_i \le U_i && i = 1, \cdots, n. \end{aligned}$$

If all the constraints on arrival time are relaxed, then the Lagrangian relaxation subproblem associated with the Lagrange multipliers $\boldsymbol{\lambda}$ will be:

$$\mathcal{LRS}/\boldsymbol{\lambda}: \quad \text{Minimize} \quad L_\lambda(\boldsymbol{x}, \boldsymbol{a}) \qquad \text{Subject to} \quad L_i \le x_i \le U_i \qquad i = 1, \cdots, n$$

where

$$L_\lambda(\boldsymbol{x}, \boldsymbol{a}) = \sum_{i=1}^{n} \alpha_i x_i + \sum_{j \in \text{input}(0)} \lambda_{j0}(a_j - A_j) + \sum_{i=1}^{n} \sum_{j \in \text{input}(i)} \lambda_{ji}(a_j + D_i - a_i) + \sum_{i=n+1}^{n+s} \lambda_{mi}(A_i + D_i - a_i).$$

Again, by the Kuhn–Tucker conditions, we have the following optimality conditions on $\boldsymbol{\lambda}$.

$$\sum_{i \in \text{output}(k)} \lambda_{ki} = \sum_{j \in \text{input}(k)} \lambda_{jk} \qquad \text{for } 1 \le k \le n+s.$$

So for $\boldsymbol{\lambda}$ satisfying the conditions, we can simplify $L_\lambda(\boldsymbol{x}, \boldsymbol{a})$

$$L_\lambda(\boldsymbol{x}, \boldsymbol{a}) = \sum_{i=1}^{n+s} \mu_i D_i + \sum_{i=1}^{n} \alpha_i x_i + \sum_{i=n+1}^{n+s} \lambda_{mi} A_i - \sum_{j \in \text{input}(0)} \lambda_{j0} A_j.$$

So the Lagrangian relaxation subproblem can be formulated in exactly the same form as the problem $\mathcal{LRS}/\boldsymbol{\mu}$ in Section III-C. $\mathcal{LRS}/\boldsymbol{\mu}$ and $\mathcal{LDP}$ can be solved as before. Therefore even with different arrival time specifications on inputs and outputs, the problem can still be solved optimally by our approach.

C. Power Consideration

The power dissipation of a circuit is mainly due to the dynamic power. The dynamic power is the power dissipated in charging and discharging capacitances in the circuit. For each i, the capacitance of component i is a linear function of its size x_i. Hence, the total dynamic power is also a linear function of $x_1, \cdots, x_n$. In other words, the dynamic power can be handled in exactly the same way as the total component area.

Sapatnekar and Chuang [21] showed that the short-circuit power of gates [25] can sometimes be a nonnegligible part of

the total power dissipation. We notice that our approach can also be extended to handle short-circuit power. As pointed out in [21], the short-circuit power of a gate is proportional to the MOS transistor gain factor and the Elmore delays of the driving gates. Since the MOS transistor gain factors are proportional to the gate sizes and the Elmore delays are posynomial functions in the gate sizes, the short-circuit power can be written as a posynomial function. So the sum of the dynamic power and the short-circuit power is also a posynomial in the gate sizes.

Consider as an example the objective of minimizing power subject to maximum delay bound. We can use Lagrangian relaxation to handle the constraints on arrival time and use the optimality conditions on $\boldsymbol{\lambda}$ to simplify $\mathcal{LRS}/\boldsymbol{\lambda}$ to $\mathcal{LRS}/\boldsymbol{\mu}$ as before. The only difference is that the objective function of $\mathcal{LRS}/\boldsymbol{\mu}$ here consists of a weighted sum of the component delays and the posynomial function corresponding to the power. This problem can still be solved by the greedy technique as in Section III-D. Each optimal local resizing step will be a bit different from before. However, it is not too difficult to see that the optimality of the greedy algorithm can still be proved similarly.

D. More Accurate Gate Model

For higher precision timing requirements, more accurate gate models are desirable. Although in Section II, we model a gate as a switch-level *RC* circuit with a resistance proportional to the gate size, better gate models can be easily integrated into our algorithm. We now show an example of using precharacterized function as the delay model for gates.

The following precharacterized delay function $D_i()$ and output slope function $T_i()$ can capture the input slope effect as well as the diffusion capacitance effect to the delay of gate i

$$D_i(x_i, t_i, C_i) = \hat{s}_i + \hat{p}_i t_i + \hat{q}_i x_i + \frac{\hat{r}_i}{x_i} C_i,$$
$$T_i(x_i, t_i, C_i) = \tilde{s}_i + \tilde{p}_i t_i + \tilde{q}_i x_i + \frac{\tilde{r}_i}{x_i} C_i,$$

where x_i is the gate size, t_i is the input rise or fall time of gate i, C_i is the capacitance load, $\hat{s}_i$, $\hat{q}_i$, $\hat{r}_i$, $\tilde{s}_i$, $\tilde{q}_i$, and $\tilde{r}_i$ are precharacterized coefficients. It is not difficult to see that while keeping the size of other components fixed, the input slope t_i is a linear function of x_i since gate i contributes only the linear term $\hat{c}_i x_i$ to its parents' capacitance load. Hence, the delay of gate i can be rewritten as follows:

$$D_i(x_i, t_i, C_i) = \hat{s}'_i + \hat{q}'_i x_i + \frac{\hat{r}_i}{x_i} C_i$$

where $\hat{s}'_i = \hat{s}_i + \hat{p}_i(\tilde{s}_j + \tilde{p}_j t_j + \tilde{q}_j x_j)$, $\hat{q}'_i = \hat{q}_i + \hat{p}_i \hat{c}_i(\tilde{r}_j / x_j)$, and component j is the parent of component i. It is not hard to see that after the substitution, $A_i(\boldsymbol{x}) = \hat{c}_i R_i + \alpha_i + \hat{q}'_i$. Hence, our algorithm in Section III still converges to the optimal solution under this modification.

V. Experimental Results

We implemented our algorithms on a PC with a 333-MHz Pentium II processor. Table I shows the experimental results on adders [15] of different sizes ranging from eight bits to 1024 bits. Number of gates range from 120 to 15 360. Number of wires range from 96 to 12 288 (note that the number of wires here means the number of sizable wire segments). The total number of sizable components range from 216 to 21 648. The stopping criteria of our algorithm is the solution is within 1% of the optimal solution. The lower bound and upper bound of the size of each gate are 1 and 100, respectively. The lower bound and upper bound of the width of each wire are 1 and 3 μm, respectively.

TABLE I
The Runtime and Storage Requirements of our Algorithm on Test Circuits of Different Sizes

Circuit Name	Circuit Size			Runtime	Memory
	# Gates	# Wires	Total	(minutes)	(Mbytes)
adder (8 bits)	120	96	216	0.00	0.48
adder (16 bits)	240	192	432	0.01	0.76
adder (32 bits)	480	384	864	0.02	1.15
adder (64 bits)	960	768	1728	0.09	1.75
adder (128 bits)	1920	1536	3456	0.28	2.82
adder (256 bits)	3840	3072	6912	0.85	5.37
adder (512 bits)	7680	6144	13824	2.75	11.83
adder (1024 bits)	15360	12288	27648	11.53	22.92

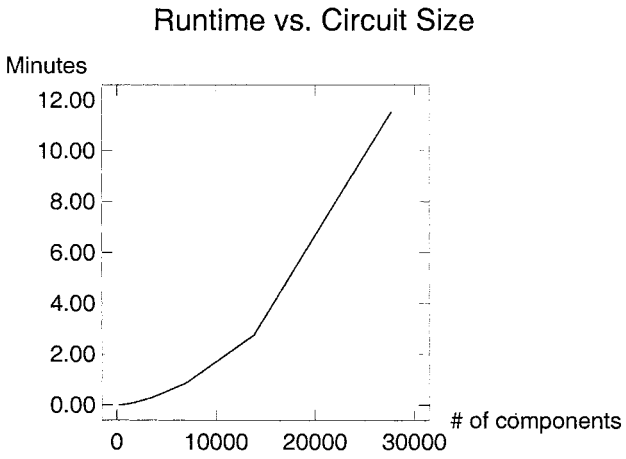

Fig. 6. The runtime requirement of our algorithm versus circuit size.

Table I shows the runtime and storage requirements of our algorithm. For a circuit with 864 sizable components, the runtime and storage requirements of our algorithm are just 1.3 s and 1.15 Mbytes. Even for a circuit with 27 648 sizable components, the runtime and storage requirements of our algorithm are 11.53 min and about 23 Mbytes only.

Figs. 6 and 7 show the runtime and storage requirements of our algorithm. By performing a linear regression on the logarithm of the data in Fig. 6, we find that the empirical runtime of our program is about $O(n^{1.7})$. Fig. 7 shows that the ratio of the storage versus the circuit size of our algorithm is close to linear. The storage requirement for each sizable component is about 0.8 kbytes.

Fig. 8 shows the convergence sequence of our algorithm SOLVE_LDP on a 128-bit adder. It shows that our algorithm converges smoothly to the optimal solution. The solid line represents the upper bound of the optimal solution and the dotted line represents the lower bound of it. The lower bound

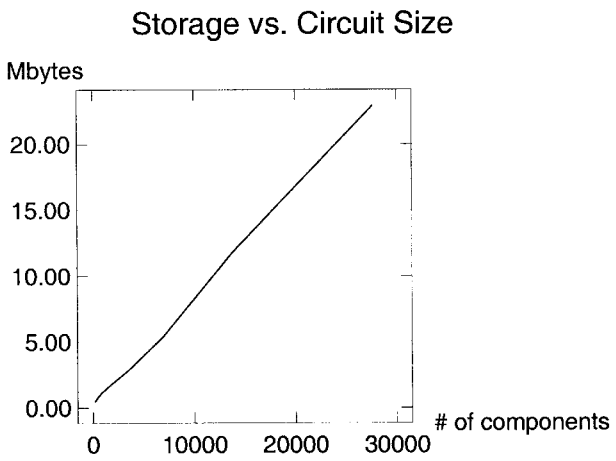

Fig. 7. The storage requirement of our algorithm versus circuit size.

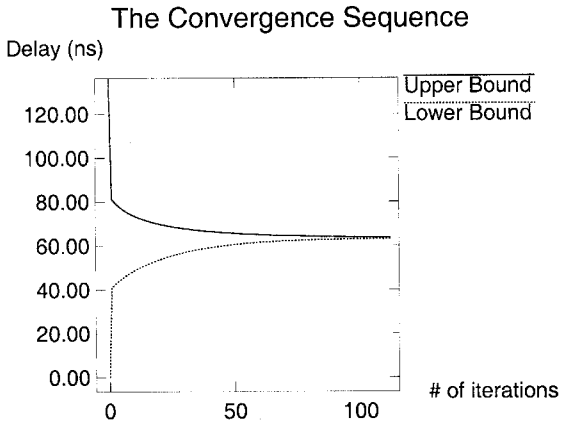

Fig. 8. The convergence sequence for a 128-bit adder.

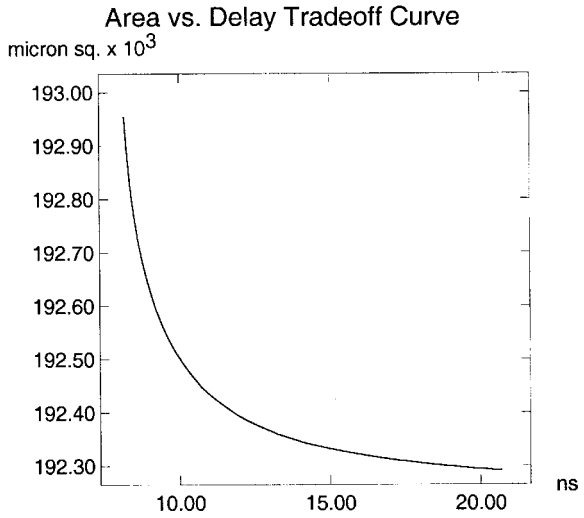

Fig. 9. The area versus delay tradeoff curve for a 16-bit adder.

values comes from the optimal value $Q(\lambda)$ of $\mathcal{LRS}/\lambda$ at current iteration. Note that the optimal solution is always in between the upper bound and the lower bound. So these curves provide useful information about the distance between the optimal solution and the current solution, and help users to decide when to stop the programs.

Fig. 9 shows the area versus delay tradeoff curve of a 16-bit adder. In our experiment, we observe that to generate a new point in the area and delay tradeoff curve, SOLVE_LDP converges in only about five iterations. It is because the λ of the previous point is a good approximation for that of the new point and, hence, the convergence of SOLVE_LDP is fast. As a result, generating these tradeoff curves requires only a little bit more runtime but provides precious information.

VI. Conclusion

We have presented a Lagrangian relaxation approach to simultaneous gate and wire sizing for combinational circuits. We have shown that this approach can handle optimally and efficiently several different objective functions and constraints, like minimize total area subject to maximum delay bound and minimizing maximum delay. We have demonstrated the idea by deriving the algorithm SGWS-LR in Section III in detail.

The Lagrangian relaxation technique reduces the problem into two subproblems, namely the Lagrangian relaxation subproblem and the Lagrangian dual problem. For the Lagrangian relaxation subproblem, we have shown that it can be greatly simplified by the Kuhn–Tucker conditions. The simplified Lagrangian relaxation subproblem is solved exactly by a very efficient greedy algorithm. For the Lagrangian dual problem, it is solved exactly by the classical subgradient optimization method.

In this paper, Elmore delay and relatively simple gate delay models are used. In the future, we would like to incorporate more accurate timing models into the Lagrangian relaxation approach. Lagrangian relaxation is such a flexible technique that the same framework should still work for more accurate timing models. However, maintaining the exactness and efficiency of the algorithm would be a challenge.

Acknowledgment

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

References

[1] M. S. Bazaraa, H. D. Sherali, and C. M. Shetty, *Nonlinear Programming: Theory and Algorithms*, 2nd ed. New York: Wiley, 1993.
[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. ACM/IEEE Design Automation Conf.*, 1996, pp. 405–408.
[3] 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.
[4] 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.
[5] C. C. N. Chu and D. F. Wong, "Greedy wire-sizing is linear time," in *Proc. Int. Symp. Physical Design*, 1998, pp. 39–44.
[6] M. A. Cirit, "Transistor sizing in CMOS circuits," in *Proc. ACM/IEEE Design Automation Conf.*, 1987, pp. 121–124.
[7] J. Cong and L. He, "An efficient approach to simultaneous transistor and interconnect sizing," in *Proc. IEEE Int. Conf. Computer-Aided Design*, 1996, pp. 181–186.
[8] J. Cong and C.-K. Koh, "Simultaneous driver and wire sizing for performance and power optimization," in *Proc. IEEE Int. Conf. Computer-Aided Design*, 1994, pp. 206–212.

[9] J. Cong and K.-S. Leung, "Optimal wiresizing under the distributed Elmore delay model," *IEEE Trans. Computer-Aided Design*, vol. 14, pp. 321–336, Mar. 1995.
[10] R. J. Duffin, E. L. Peterson, and C. Zener. *Geometric Programming—Theory and Application*. New York: Wiley, 1967.
[11] 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.
[12] J. P. Fishburn and A. E. Dunlop, "TILOS: A posynominal programming approach to transistor sizing," in *Proc. IEEE Int. Conf. Computer-Aided Design*, 1985, pp. 326–328.
[13] M. L. Fisher, "An application oriented guide to Lagrangian relaxation," *Interfaces*, vol. 15, no. 2, pp. 10–21, Mar./Apr. 1985.
[14] D. G. Luenberger, *Linear and Nonlinear Programming*, 2nd ed. Reading, MA: Addison Wesley, 1984.
[15] M. Morris Mano, *Digital Logic and Computer Design*. Englewood Cliffs, NJ: Prentice-Hall, 1979.
[16] D. P. Marple, "Performance optimization of digital VLSI circuits," Stanford Univ, Stanford, CA, Tech. Rep. CSL-TR-86-308, Oct. 1986.
[17] ——, "Transistor size optimization in the Tailor layout system," in *Proc. ACM/IEEE Design Automation Conf.*, 1989, pp. 43–48.
[18] 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.
[19] N. Menezes, S. Pullela, F. Dartu, and L. T. Pileggi, "RC interconnect syntheses—A moment fitting approach," in *Proc. IEEE Int. Conf. Computer-Aided Design*, 1994, pp. 418–425.
[20] N. Menezes, S. Pullela, and L. T. Pileggi, "Simultaneous gate and interconnect sizing for circuit level delay optimization," in *Proc. ACM/IEEE Design Automation Conf.*, 1995, pp. 690–695.
[21] S. S. Sapatnekar and W. Chuang, "Power vs. delay in gate sizing: Conflicting objectives?" in *Proc. IEEE Int. Conf. Computer-Aided Design*, 1995, pp. 463–466.
[22] S. S. Sapatnekar, "RC interconnect optimization under the Elmore delay model," in *Proc. ACM/IEEE Design Automation Conf.*, 1994, pp. 387–391.
[23] 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," *IEEE Trans. Computer-Aided Design*, vol. 12, pp. 1621–1634, Nov. 1993.
[24] J. Shyu, J. P. Fishburn, A. E. Dunlop, and A. L. Sangiovanni-Vincentelli, "Optimization-based transistor sizing," *IEEE J. Solid-State Circuits*, vol. 23, pp. 400–409, Apr. 1988.
[25] H. Veendrick, "Short-circuit dissipation of static CMOS circuitry and its impact on the design of buffer circuits," *IEEE J. Solid-State Circuits*, vol. SC-19, pp. 468–473, Aug. 1984.

Chung-Ping Chen received the B.S. degree in computer science from National Chiao-Tung University, Hsinchu, Taiwan, in 1990. He received the M.S. and Ph.D. degrees in computer science from the University of Texas, Austin, in 1996 and 1998, respectively.

In 1997, he joined Intel Corporation, where he is currently a Senior Computer-Aided Design (CAD) Engineer with Strategic CAD Laboratories. His research interests are in the area of computer-aided design and microprocessor circuit design with an emphasis on interconnect and circuit optimization as well as signal integrity analysis and optimization.

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

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