Fast and Exact Simultaneous Gate and Wire Sizing by Lagrangian Relaxation *

Chung-Ping Chen †, Chris C. N. Chu and D. F. Wong
ccp@cs.utexas.edu, cnchu@cs.utexas.edu and wong@cs.utexas.edu
Department of Computer Sciences, University of Texas at Austin, Austin, TX 78712.

Abstract

This paper considers simultaneous gate and wire sizing for general 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" local optimizations, and is extremely economical and fast. For example, we can optimize a circuit with 27,648 gates and wires in about 36 minutes using under 23 MB memory on an IBM RS/6000 workstation.

1 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, 15, 20]. 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, 17, 19].

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, 16, 18, 20]. [8] studied simultaneous driver and wire sizing and [2] considered simultaneous wire and buffer sizing, but both works only apply to circuits that are of tree topology. For simultaneous gate and wire sizing for general circuits, [18] uses a least-square optimization technique, [16] employs a sequential quadratic programming approach, and [7] uses a greedy sizing technique in conjunction with dynamic programming. But none of these algorithms can 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 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). 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" local optimizations, and is extremely economical and fast. For example, we can optimize a circuit with 27,648 gates and wires in about 36 minutes using under 23 MB memory on an IBM RS/6000 workstation.

The problem in this paper is formulated as a geometric program [10]. Note that the transistor sizing problem is similar to our problem and was also formulated as a geometric program long time ago [12]. However, it would be very slow to solve it by some general-purpose geometric programming solver. So instead of solving it exactly, [12] proposed TILOS, which is based on an efficient sensitivity-based heuristic. Years later, [20] transforms the geometric program into a convex program and they solve 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. However, as we explore the special structure of the geometric program, our tailored algorithm is much faster than algorithms using general-purpose solvers as in [20]. For example, the largest test circuit in [20] has 832 transistors and the reported runtime and memory are 9 hours (on a Sun SPARCstation 1) and 11 MB, respectively. For a problem of similar size (864), our approach only needs 7 sec-

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

†Current address of Chung-Ping Chen is: Intel Corporation, 2111 N.E. 25th Ave, Hillsboro, OR 97124-5961.

onds of runtime (on a RS/6000 workstation) and 1.15 MB of memory.

The rest of this paper is organized as follows. In Section 2, we will introduce some notations and terminology that we will use in this paper. In Section 3, we will present our algorithm for the problem of minimizing total area subject to maximum delay bound. In Section 4, we will show how to modify our algorithm to minimize maximum delay, to handle arrival time specifications at all inputs and outputs, to consider power consumption and to use a more accurate gate model. In Section 5, experimental results of our algorithms are presented.

2 Preliminaries

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

For a general VLSI circuit, we can ignore all latches and optimize its combinational subcircuits. Therefore, we will 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 $1 \leq i \leq s$, let R_i^D be the driver resistance of the ith input driver. For $1 \leq i \leq t$, let C_i^L be the load capacitance of the ith output load. See Figure 1 for an illustration of a circuit. Note that it is reasonable to assume that the gates are of bounded fanin. Hence $s = O(n)$ and $t = O(n)$.

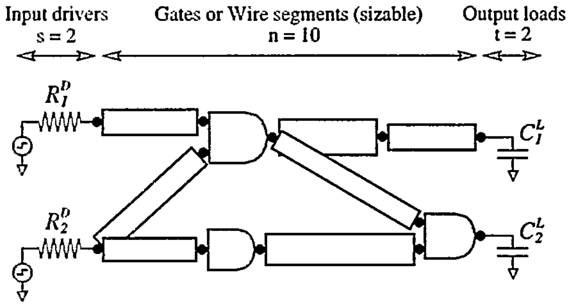

Figure 1: *A combinational circuit.*

A gate, a wire segment, or an input driver is called a *component*. In order to unify the notations that we will introduce later, imagine that two factitious components are added to the circuit as shown in Figure 2. 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 $0, \ldots, m$ as follows. The node with index 0 is the output point of the output component. For $1 \leq i \leq t$, the node with index i is the one connecting to the ith output load. For $t + 1 \leq i \leq n$, 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 $n + 1 \leq i \leq n + s$, 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 $0, \ldots, m$ such that the output of the component with index i is connected to node i. See Figure 2 for an illustration of the circuit in Figure 1 with factitious components, node indexes and component indexes.

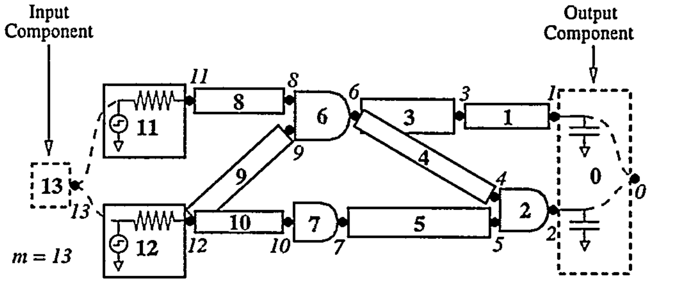

Figure 2: *The circuit in Figure 1 with factitious components, node indexes and component indexes.*

For $0 \leq i \leq m - 1$, let *input*(i) be the set of indexes of components directly connected to the input(s) of component i. For $1 \leq i \leq m$, let *output*(i) be the set of indexes of components directly connected to the output of component i. For example, for the circuit in Figure 2, *input*$(0) = \{1, 2\}$, *input*$(2) = \{4, 5\}$, and *output*$(2) = \{0\}$. Note that $j \in$ *input*(i) if and only if $i \in$ *output*(j).

Let $\mathcal{G}$, $\mathcal{W}$ and $\mathcal{D}$ be respectively the set of component indexes of gates, wire segments and input drivers in the circuit. For the circuit in Figure 2, $\mathcal{G} = \{2, 6, 7\}$, $\mathcal{W} = \{1, 3, 4, 5, 8, 9, 10\}$ and $\mathcal{D} = \{11, 12\}$.

If $i \in \mathcal{G}$, then let x_i be the gate size, r_i be the output resistance of the gate and c_i be the input capacitance of a pin of the gate. (To simplify the notations, we assume without loss of generality that the input capacitances of all input pins of a gate are the same.) Let $\widehat{r}_i$ and $\widehat{c}_i$ be respectively the unit size output resistance and the input capacitance per unit size of gate i. Then $r_i = \widehat{r}_i / x_i$ and $c_i = \widehat{c}_i x_i$. If $i \in \mathcal{W}$, then let x_i be the segment width, r_i be the segment resistance and c_i be the segment capacitance. Let $\widehat{r}_i$, $\widehat{c}_i$ and f_i be respectively the unit width wire resistance, the wire area capacitance per unit width and the wire fringing capacitance of segment i. Then $r_i = \widehat{r}_i / x_i$ and $c_i = \widehat{c}_i x_i + f_i$. For $i \in \mathcal{G} \cup \mathcal{W}$, let L_i and U_i be respectively the lower bound and upper bound of the value of x_i, i.e. $L_i \leq x_i \leq U_i$.

For the purpose of delay calculation, we model components as RC circuits. A gate is modeled as a switch-level RC circuit as shown in Figure 3. (For simplicity, we ignore the intrinsic gate delay in the model. It is easy to see that all our results will still hold even if intrinsic delay is considered.) A wire segment is modeled as a π-type RC circuit as shown in Figure 4.

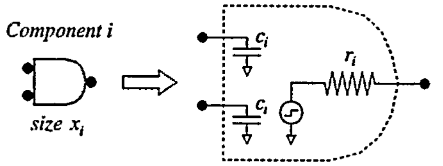

Figure 3: *The model of component i, which is a gate, by a switch-level RC circuit. Although the gate shown here is a 2-input AND gate, the model can be easily generalized for any gate with any number of input pins.*

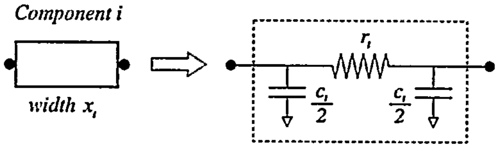

Figure 4: *The model of component i, which is a wire segment, by a π-type RC circuit.*

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 2 factitious components) contains a resistor. We label the resistors by indexes $1, \ldots, n+s$ such that resistor i is the one inside component i. For convenience, for $i \in \mathcal{D}$, let $r_i = R^D_{i-n}$ (i.e. the driver resistance of the $(i-n)$th input driver). So for $i \in \mathcal{G} \cup \mathcal{W} \cup \mathcal{D}$, the resistance of resistor i is r_i. For $i \in \mathcal{G} \cup \mathcal{W} \cup \mathcal{D}$, let C_i be the downstream capacitance of resistor i. Figure 5 shows the circuit in Figure 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 resistor 5 and resistor 12 are shaded.

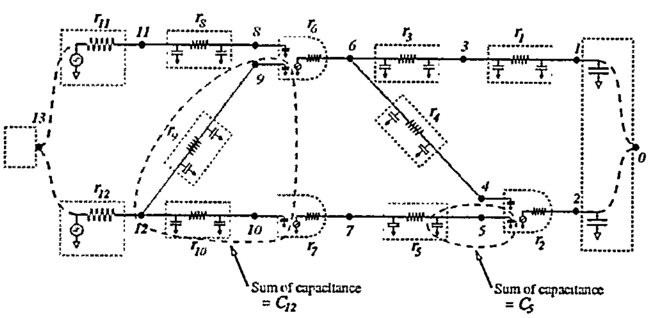

Figure 5: *The circuit in Figure 2 after replacing the gates and wire segments by the RC models.*

Let $D_i = r_i C_i$ be the delay associated with resistor i. We represent a signal path passing through resistors $i_1, \ldots, i_k$ by the set $p = \{i_1, \ldots, i_k\}$. Let $\mathcal{P}$ 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 $p \in \mathcal{P}$, the Elmore delay along path p is $\sum_{i \in p} D_i$.

3 Minimizing total area subject to maximum delay bound

In this section, we will solve the problem of minimizing the total component area with respect to component sizes $x_1, \ldots, x_n$ 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). We will formulate the problem as a constrained optimization problem and then solve it using 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, 14].

We call the constrained optimization problem to be solved the primal problem ($\mathcal{PP}$). In Lagrangian relaxation, "troublesome" constraints in $\mathcal{PP}$ 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 λ 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 λ ($\mathcal{LRS}/\lambda$). It can be shown that there exists a vector λ such that the optimal solution of $\mathcal{LRS}/\lambda$ is also the optimal solution of the original problem $\mathcal{PP}$. The problem of finding such a vector λ is called the Lagrangian dual problem ($\mathcal{LDP}$). So if we can solve both $\mathcal{LRS}/\lambda$ and $\mathcal{LDP}$, then the optimal solution of $\mathcal{PP}$ will be given by $\mathcal{LRS}/\lambda$ where λ is the optimal solution of $\mathcal{LDP}$.

In Section 3.1, we will show how to formulate the gate and wire sizing problem as a constrained optimization problem with a polynomial number of constraints. This formulation is our primal problem ($\mathcal{PP}$). In Section 3.2, we will show how $\mathcal{PP}$ is relaxed to obtain the $\mathcal{LRS}/\lambda$. We will use the Kuhn-Tucker conditions (see [1] for a reference) to greatly simplify $\mathcal{LRS}/\lambda$. We call the simplified version $\mathcal{LRS}/\mu$. In Section 3.3, we will show how to solve $\mathcal{LRS}/\mu$ (i.e. $\mathcal{LRS}/\lambda$) for any fixed vector μ. In Section 3.4, we will show how to solve the $\mathcal{LDP}$ by the classical method of subgradient optimization. Due to space limitation, all the proofs in this section have been omitted. They can be found in [5].

3.1 Problem formulation

The total component area can be written as $\sum_{i=1}^{n} \alpha_i x_i$ for some constants $\alpha_1, \ldots, \alpha_n$. So the problem of minimizing total area subject to maximum delay bound can be formulated directly as the mathematical program:

$$\begin{array}{lll} \text{min.} & \sum_{i=1}^{n} \alpha_i x_i & \\ \text{s.t.} & \sum_{i \in p} D_i \leq A_0 & \forall p \in \mathcal{P} \\ & L_i \leq x_i \leq U_i & i \in \mathcal{G} \cup \mathcal{W} \end{array}$$

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:

$$\begin{array}{lll} \mathcal{PP}: & \text{min.} & \sum_{i=1}^{n} \alpha_i x_i \\ & \text{s.t.} & a_j \leq A_0 \quad\quad j \in input(0) \\ & & a_j + D_i \leq a_i \quad i \in \mathcal{G} \cup \mathcal{W} \wedge \forall j \in input(i) \\ & & D_i \leq a_i \quad\quad i \in \mathcal{D} \\ & & L_i \leq x_i \leq U_i \quad i \in \mathcal{G} \cup \mathcal{W} \end{array}$$

Note that the number of constraints in $\mathcal{PP}$ is linear in n. Also note that for the problem $\mathcal{PP}$, the objective function and the constraints can be rewritten in the form of posynomials [10]. It is well known that under a variable transformation, the problem is convex. So $\mathcal{PP}$ has a unique global minimum and no other local minimum. We will see how to solve $\mathcal{PP}$ in the following.

3.2 Lagrangian Relaxation

Following the Lagrangian relaxation procedure, we introduce a non-negative value called the Lagrange multiplier for each constraint on arrival time. For $j \in input(0)$ (i.e. $j = 1, \ldots, t$), we introduce λ_{j0} for the constraint $a_j \leq A_0$. For $i \in \mathcal{G} \cup \mathcal{W}$ and for $j \in input(i)$, we introduce λ_{ji} for the constraint $a_j + D_i \leq a_i$. For $i \in \mathcal{D}$, we introduce λ_{mi} for the constraint $D_i \leq a_i$. Let λ be a vector of all the Lagrange multipliers introduced. Let $x = (x_1, \ldots, x_n)$ and $a = (a_1, \ldots, a_{n+s})$. Let

$$\begin{aligned} L_\lambda(x, a) \;=\; & \sum_{i=1}^{n} \alpha_i x_i \\ & + \sum_{j \in input(0)} \lambda_{j0}(a_j - A_0) \\ & + \sum_{i \in \mathcal{G} \cup \mathcal{W}} \sum_{j \in input(i)} \lambda_{ji}(a_j + D_i - a_i) \\ & + \sum_{i \in \mathcal{D}} \lambda_{mi}(D_i - a_i) \end{aligned} \tag{1}$$

Then the Lagrangian relaxation subproblem associated with the Lagrange multipliers λ will be:

$$\begin{array}{lll} \mathcal{LRS}/\lambda: & \text{min.} & L_\lambda(x, a) \\ & \text{s.t.} & L_i \leq x_i \leq U_i \quad i \in \mathcal{G} \cup \mathcal{W} \end{array}$$

Let (x^*, a^*) be the optimal solution of $\mathcal{PP}$. By Kuhn-Tucker conditions, if the optimal solution of $\mathcal{LRS}/\lambda$ is also the optimal solution of $\mathcal{PP}$, then λ must satisfy the conditions $\frac{\partial L_\lambda}{\partial a_i}(x^*, a^*) = 0$ for $1 \leq i \leq n + s$. Therefore, we can consider only those λ satisfying these conditions.

By rearranging (1), we can write

$$\begin{aligned} L_\lambda(x, a) \;=\; & \sum_{i=1}^{n+s} \left(\sum_{k \in output(i)} \lambda_{ik} - \sum_{j \in input(i)} \lambda_{ji} \right) a_i \\ & + \text{ terms independent of all } a_i\text{'s} \end{aligned}$$

So $\partial L_\lambda / \partial a_i = 0$ for $1 \leq i \leq n + s$ imply the following optimality conditions on Lagrange Multipliers λ:

$$\sum_{k \in output(i)} \lambda_{ik} = \sum_{j \in input(i)} \lambda_{ji} \quad \text{for } 1 \leq i \leq n + s \tag{2}$$

Let $\Omega_\lambda = \{\lambda \geq \mathbf{0} : \lambda \text{ satisfies (2) }\}$. We observe that by considering only those λ in Ω_λ and substituting (2) back to (1), we can greatly simplify the objective function $L_\lambda(x, a)$, and hence the problem $\mathcal{LRS}/\lambda$. This is summarized in the following lemma.

Lemma 1 *For any $\lambda \in \Omega_\lambda$, the optimal x of $\mathcal{LRS}/\lambda$ is the same as the optimal x of*

$$\begin{array}{lll} \mathcal{LRS}/\mu: & \textit{min.} & L_\mu(x) \\ & \textit{s.t.} & L_i \leq x_i \leq U_i \quad i \in \mathcal{G} \cup \mathcal{W} \end{array}$$

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

To solve $\mathcal{LRS}/\lambda$, we can solve $\mathcal{LRS}/\mu$ to find the optimal x. Then the optimal 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 $\mathcal{PP}$.

3.3 Solving $\mathcal{LRS}/\mu$

In this subsection, for any fixed $\mu \geq \mathbf{0}$, we will show how to solve $\mathcal{LRS}/\mu$ optimally by a greedy algorithm based on iteratively re-sizing the gates and wire segments. Similar techniques have been successfully applied to some other wire or buffer sizing problems before (e.g. [3, 9]).

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 Figure 5, $upstream(1) = \{3, 6\}$ and $upstream(6) = \{8, 9, 11, 12\}$. Let $R_i = \sum_{j \in upstream(i)} \mu_j r_j$ (i.e. R_i is a weighted upstream resistance of component i). For $i \in \mathcal{W}$, let $C_i' = C_i - \widehat{c}_i x_i / 2$, and for $i \in \mathcal{G} \cup \mathcal{D}$, let $C_i' = C_i$. Note that for $1 \leq i \leq n$, C_i' is independent of x_i.

If we re-size component i (i.e. changing x_i) while keeping the sizes of all the other components fixed, we say that it is a local re-sizing of component i. An optimal local re-sizing of component i is a local re-sizing that minimize $L_\mu(x)$, and is given by the following lemma.

Lemma 2 *Let $\tilde{x} = (\tilde{x}_1, \ldots, \tilde{x}_n)$ be a component-sizing solution. An optimal local re-sizing of component i is given by changing the size of component i to*

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

where $A_i(x) = \widehat{c}_i R_i + \alpha_i$ and $B_i(x) = \mu_i \widehat{r}_i C'_i$.

$\mathcal{LRS}/\mu$ can be solved by a greedy algorithm based on iteratively re-sizing the components. In each iteration, the components are examined one at a time; each time a component is re-sized optimally using Lemma 2 while keeping the sizes of the other components fixed. We call the algorithm SOLVE_LRS/μ and it is described below. Note that in order to use Lemma 2 to re-size component i, we need to compute R_i and C'_i first. Our algorithm SOLVE_LRS/μ 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/$\mu$:
Output: $x = (x_1, \ldots, x_n)$ which minimizes $L_\mu(x)$
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
      $C'_i := \begin{cases} C_i^L & \text{if } i \in \mathcal{G} \\ C_i^L + f_i/2 & \text{if } i \in \mathcal{W} \end{cases}$
   for $i := t+1$ to $n$ do
      $C'_i := \begin{cases} 0 & \text{if } i \in \mathcal{G} \\ f_i/2 & \text{if } i \in \mathcal{W} \end{cases}$
      for all $k$ s.t. $i \in input(k)$ do
         $C'_i := \begin{cases} C'_i + \widehat{c}_k x_k & \text{if } k \in \mathcal{G} \\ C'_i + \widehat{c}_k x_k + f_k/2 + C'_k & \text{if } k \in \mathcal{W} \end{cases}$
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 $j \in input(i)$ do
         $R_i := \begin{cases} R_i + \mu_j \widehat{r}_j / x_j & \text{if } j \in \mathcal{G} \\ R_i + \mu_j \widehat{r}_j / x_j + R_j & \text{if } j \in \mathcal{W} \\ R_i + \mu_j R_{j-n}^{\mathcal{D}} & \text{if } j \in \mathcal{D} \end{cases}$
      $x_i := \min\left(U_i, \max\left(L_i, \sqrt{\mu_i \widehat{r}_i C'_i / (\widehat{c}_i R_i + \alpha_i)}\right)\right)$
4. Repeat step 2 and 3 until no improvement.
```

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

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

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

3.4 Solving the $\mathcal{LDP}$

Define the function $Q(\lambda)$ = the optimal value of the problem $\mathcal{LRS}/\lambda$. In this subsection, we will consider the Lagrangian dual problem:

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

As said in Section 3.1, $\mathcal{PP}$ can be transformed into a convex problem. So Theorem 6.2.4 of [1] implies that if λ is the optimal solution of $\mathcal{LDP}$, then the optimal solution of $\mathcal{LRS}/\lambda$ will also optimize $\mathcal{PP}$.

By Theorem 6.3.1 of [1], $Q(\lambda)$ is a concave function over $\lambda \ge 0$. However, $\mathcal{LRS}/\lambda$ 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 λ, the method iteratively moves from the current point to a new point following the subgradient direction. At step k, we first solve $\mathcal{LRS}/\lambda$ (by solving the simpler $\mathcal{LRS}/\mu$). 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 λ. After each time we moved, we project λ back to the nearest point in Ω_λ so that we can solve $\mathcal{LRS}/\mu$ instead of $\mathcal{LRS}/\lambda$ for the next iteration. The procedure is repeated until it converges.

It is well known (see Theorem 8.9.2 of [1]) 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.

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

We conclude Section 3 by giving the algorithm SGWS-LR (Simultaneous Gate and Wire Sizing by Lagrangian Relaxation) below.

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.*

```
ALGORITHM SOLVE_LDP:
Output: $\lambda$ which maximizes $\mathcal{LRS}/\lambda$
1. $k := 1$ /* step counter */
   $\lambda :=$ arbitrary initial vector in $\Omega_\lambda$
2. $\mu = (\mu_0, \ldots, \mu_{n+s})$ where $\mu_i = \sum_{j \in input(i)} \lambda_{ji}$
   Solve $\mathcal{LRS}/\lambda$. (Solve $\mathcal{LRS}/\mu$ by SOLVE_LRS/$\mu$,
       and then calculate $a_1, \ldots, a_{n+s}$ as in Section 3.2).
3. /* Move to a new $\lambda$ by adjusting multipliers $\lambda_{ji}$ */
   for $i := 0$ to $n + s$ do
       for all $j \in input(i)$ do
           $\lambda_{ji} := \begin{cases} \lambda_{ji} + \rho_k(a_j - A_0) & \text{if } i = 0 \\ \lambda_{ji} + \rho_k(a_j + D_i - a_i) & \text{if } i \in \mathcal{G} \cup \mathcal{W} \\ \lambda_{ji} + \rho_k(D_i - a_i) & \text{if } i \in \mathcal{D} \end{cases}$
4. Project $\lambda$ onto the nearest point in $\Omega_\lambda$.
5. $k := k + 1$
6. Repeat step 2–5 until $(\sum_{i=1}^{n} \alpha_i x_i - Q(\lambda)) \leq$ error bd.
```

```
ALGORITHM SGWS-LR:
Output: the optimal gate and wire sizing solution $x$
1. Call SOLVE_LDP to find the optimal $\lambda$.
2. $\mu = (\mu_0, \ldots, \mu_{n+s})$ where $\mu_i = \sum_{j \in input(i)} \lambda_{ji}$
3. Call SOLVE_LRS/$\mu$ to find the optimal $x$.
```

4 Extensions

In this section, we will show how to extend our approach to handle problems with other objectives and with other constraints. For all the extensions, as we will see, only slight modifications to our algorithms presented in Section 3 are needed. Moreover, convergence to global optimal solutions is still guaranteed. Actually, it is not difficult to see that any combination of the problem in Section 3 and the extensions here 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.

4.1 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 (i.e. maximum delay), and we want to minimize a_0. As in Section 3.1, the problem can be formulated as the mathematical program below:

$$\begin{array}{lll} \mathcal{PP}: & \text{min.} & a_0 \\ & \text{s.t.} & a_j \leq a_0 \quad\quad j \in input(0) \\ & & a_j + D_i \leq a_i \quad i \in \mathcal{G} \cup \mathcal{W} \wedge \forall j \in input(i) \\ & & D_i \leq a_i \quad\quad i \in \mathcal{D} \\ & & L_i \leq x_i \leq U_i \quad i \in \mathcal{G} \cup \mathcal{W} \end{array}$$

As in Section 3.2, we relax all the constraints on arrival time to obtain the problem $\mathcal{LRS}/\lambda$. By Kuhn-Tucker conditions, we can focus on those λ in $\Omega_\lambda = \{\lambda \geq 0 : \sum_{k \in output(i)} \lambda_{ik} = \sum_{j \in input(i)} \lambda_{ji}$ for $1 \leq i \leq n + s \wedge 1 = \sum_{j \in input(0)} \lambda_{j0}\}$. Then $\mathcal{LRS}/\lambda$ can be simplified to

$$\begin{array}{lll} \mathcal{LRS}/\mu: & \text{min.} & L_\mu(x) \\ & \text{s.t.} & L_i \leq x_i \leq U_i \quad i \in \mathcal{G} \cup \mathcal{W} \end{array}$$

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

It is easy to see that $\mathcal{LRS}/\mu$ can be solved optimally by the iterative local re-sizing algorithm in Section 3.3 and the corresponding $\mathcal{LDP}$ can be solved optimally by the subgradient optimization method as described in Section 3.4. 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 our Lagrangian relaxation approach. The constraint on area can be relaxed and incorporated into the objective function as well. The function $L_\lambda(x, a)$ will be of the same form as the one in Section 3.2.

4.2 Arrival Time Specifications on Inputs and Outputs

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 $i \in \mathcal{D}$, let A_i be the arrival time specification of the input signal at the $(i - n)$th input driver. For $j \in input(0)$, 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:

$$\begin{array}{lll} \mathcal{PP}: & \text{min.} & \sum_{i=1}^{n} \alpha_i x_i \\ & \text{s.t.} & a_j \leq A_j \quad\quad j \in input(0) \\ & & a_j + D_i \leq a_i \quad i \in \mathcal{G} \cup \mathcal{W} \wedge \forall j \in input(i) \\ & & A_i + D_i \leq a_i \quad i \in \mathcal{D} \\ & & L_i \leq x_i \leq U_i \quad i \in \mathcal{G} \cup \mathcal{W} \end{array}$$

We can obtain exactly the same optimality conditions on Lagrange multipliers as (2) in Section 3.2. The problem $\mathcal{LRS}/\mu$ is also in exactly the same form as the one in Lemma 1. So $\mathcal{LRS}/\mu$ and $\mathcal{LDP}$ can be solved as before.

4.3 Power Consideration

For each i, the power consumption of component i is proportional to its size x_i. Therefore, the total power consumption can be written as $\sum_{i=1}^{n} \beta_i x_i$ for some constants $\beta_1, \ldots, \beta_n$. It is of the same form as the total component area. So it is easy to see that it can be handle in exactly the same way as component area.

4.4 More Accurate Gate Model

For higher precision timing requirements, more accurate gate models are desirable. Although in Section 2, 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:

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

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, $\widehat{s}_i, \widehat{q}_i, \widehat{r}_i, \tilde{s}_i, \tilde{q}_i$ 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 $\widehat{c}_i x_i$ to its parents' capacitance load. Hence the delay of gate i can be rewritten as follows:

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

where $\widehat{s}_i{}' = \widehat{s}_i + \widehat{p}_i(\tilde{s}_j + \tilde{p}_j t_j + \tilde{q}_j x_j)$, $\widehat{q}_i{}' = \widehat{q}_i + \widehat{p}_i \widehat{c}_i \frac{\tilde{r}_j}{x_j}$, and component j is the parent of component i. It is not hard to see that after the substitution, $A_i(x) = \widehat{c}_i R_i + \alpha_i + \widehat{q}_i{}'$. Hence our algorithm in Section 3 will still converge to the optimal solution under this modification.

5 Experimental Results

We implemented our algorithms in an RS/6000 workstation. We ran our algorithms on adders of different sizes ranging from 8 bits to 512 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 27,648. 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. The stopping criteria of our algorithm is the solution is within 1% of the optimal solution.

Table 1 shows the runtime and storage requirements of our algorithm. Even for a circuit with 27,648 sizable components, the runtime and storage requirements of our algorithm are only about half an hour and 23 MB respectively. The maximum delays for the solution of minimum gate and wire sizes, and for our solution are also listed.

Figure 6 and Figure 7 show the runtime and storage requirements of our algorithm respectively. Figure 6 shows that the runtime increases roughly three times when the circuit size is doubled. Hence the empirical runtime of our program is about $n^{\log 3/\log 2} \approx n^{1.6}$. Figure 7 shows that the storage requirement is linear to the circuit size. The storage requirement for each sizable component is about 0.8 KB.

Figure 8 shows the convergence sequence of our algorithm SOLVE_LDP on a 128-bit adder. It shows that our algorithm converges steadily to the optimal solution. The solid line and the dotted line represent respectively the upper bound and lower bound of the optimal delay. The lower bound values come from the optimal value of $\mathcal{LRS}/\lambda$ at current iteration.

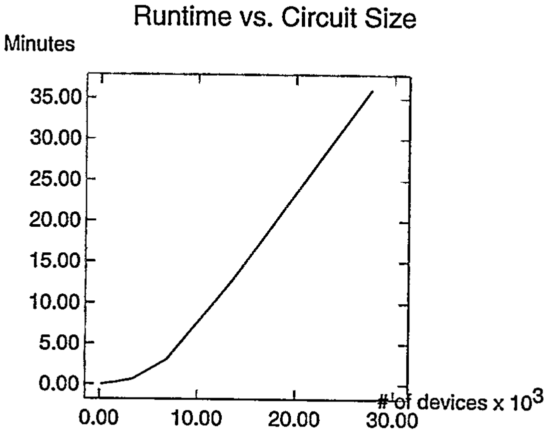

Figure 6: *The runtime requirement of our algorithm vs. circuit size.*

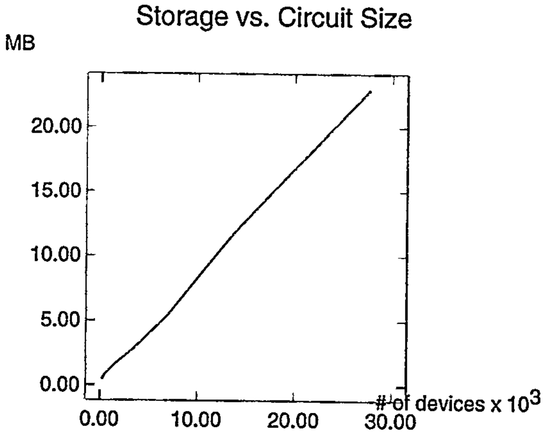

Figure 7: *The storage requirement of our algorithm vs. circuit size.*

Note that the optimal solution is always inbetween 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 program.

Figure 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 5 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.

References

[1] M. S. Bazaraa, H. D. Sherali, and C. M. Shetty. *Nonlinear Programming: Theory and Algorithms*. John Wiley & Sons, Inc., second edition, 1993.

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

Circuit Name	Circuit Size			Maximum Delay (ns)		Runtime	Memory
	# Gates	# Wires	Total	Min. Size	Our Alg.	(min:sec)	(MB)
adder (8 bits)	120	96	216	8.55	4.70	0:01	0.48
adder (16 bits)	240	192	432	17.23	8.12	0:02	0.76
adder (32 bits)	480	384	864	33.36	16.00	0:07	1.15
adder (64 bits)	960	768	1728	66.07	31.90	0:15	1.75
adder (128 bits)	1920	1536	3456	131.51	63.70	0:39	2.82
adder (256 bits)	3840	3072	6912	262.43	127.32	3:05	5.37
adder (512 bits)	7680	6144	13824	524.08	256.21	13:09	11.83
adder (1024 bits)	15360	12288	27648	1047.53	508.95	36:12	22.92

Table 1: *The runtime and storage requirements of our algorithm on test circuits of different sizes.*

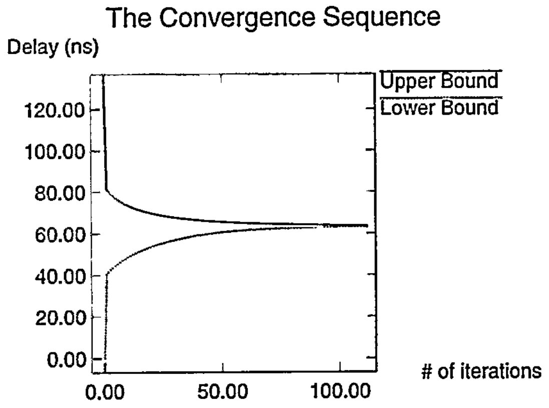

Figure 8: *The convergence sequence for a 128-bit adder.*

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

[4] Chung-Ping Chen, Hai Zhou, and D. F. Wong. Optimal nonuniform wire-sizing under the Elmore delay model. In *Proc. IEEE ICCAD*, pages 38–43, 1996.

[5] Chris C. N. Chu, Chung-Ping Chen, and D. F. Wong. Fast and exact simultaneous gate and wire sizing by Lagrangian relaxation. Tech. Rep. TR98–06, UT Austin CS Dept., February 1998.

[6] M. A. Cirit. Transistor sizing in CMOS circuits. In *Proc. ACM/IEEE DAC*, pages 121–124, 1987.

[7] Jason Cong and Lei He. An efficient approach to simultaneous transistor and interconnect sizing. In *Proc. IEEE ICCAD*, pages 181–186, 1996.

[8] Jason Cong and Cheng-Kok Koh. Simultaneous driver and wire sizing for performance and power optimization. In *Proc. IEEE ICCAD*, pages 206–212, 1994.

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

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

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

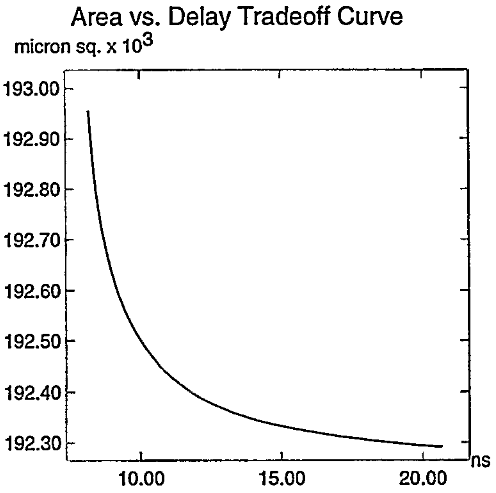

Figure 9: *The area vs. delay tradeoff curve for a 16-bit adder.*

[12] J. P. Fishburn and A. E. Dunlop. TILOS: A posynominal programming approach to transistor sizing. In *Proc. IEEE ICCAD*, pages 326–328, 1985.

[13] M. L. Fisher. An application oriented guide to lagrangian relaxation. *Interfaces*, 15(2):10–21, March–April 1985.

[14] D. G. Luenberger. *Linear and Nonlinear Programming.* Addison Wesley, second edition, 1984.

[15] David P. Marple and Abbas El Gamal. Optimal selection of transistor sizes in digital VLSI circuits. In *Proc. 1987 Stanford Conf.*, pages 151–172, 1987.

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

[17] N. Menezes, S. Pullela, F. Dartu, and L. T. Pileggi. RC interconnect syntheses-a moment fitting approach. In *Proc. IEEE ICCAD*, pages 418–425, 1994.

[18] N. Menezes, S. Pullela, and L. T. Pileggi. Simultaneous gate and interconnect sizing for circuit level delay optimization. In *Proc. ACM/IEEE DAC*, pages 690–695, 1995.

[19] Sachin S. Sapatnekar. RC interconnect optimization under the Elmore delay model. In *Proc. ACM/IEEE DAC*, pages 387–391, 1994.

[20] Sachin 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 optimizaiton. *IEEE Trans. on CAD*, 12(11):1621–1634, November 1993.