Simultaneous Transistor and Interconnect Sizing Using General Dominance Property

Jason Cong and Lei He
Department of Computer Science
University of California, Los Angeles, CA 90095

Abstract

In this paper, we study the *simultaneous transistor and interconnect sizing (STIS) problem.* Our contributions include: (1) We formulated the *STIS* problem using a distributed RC circuit model which models the waveform-dependent transistor resistances, the distributed nature of the interconnects and the transistor-interconnect interactions. (2) We showed a general *dominance property* for a large class of posynomial functions (Theorems 1 and 2) and developed efficient algorithms based on recursive *local refinement* or *bundled refinement* for optimizing such functions. Although our intended application is to develop optimal algorithms for the *STIS* problem under a wide range of transistor and interconnect models, it also has direct applications to many other optimization problems in VLSI CAD and other domains. (3) Based on the general dominance property, we developed efficient and optimal algorithms for the *STIS* problem, which are much more efficient than the mathematical programming based methods such as the convex-programming based transistor-sizing and superior to the sensitivity-based heuristics used in many transistor or interconnect sizing works in terms of both global convergence and optimality. The preliminary experiments for both transistor sizing and simultaneous transistor and interconnect sizing are reported. To our knowledge, this is the first in-depth study of the simultaneous transistor and interconnect sizing problem.

1 Introduction

It is well recognized that interconnect delay has become the dominating factor in determining the circuit performance in deep submicron designs. We believe that the most effective way for performance optimization in deep submicron design is to consider both logic and interconnect designs throughout the entire design process (from RTL level to layout design). As part of our effort to develop an unified methodology and platform for simultaneous logic and interconnect design and optimization, we study the simultaneous transistor and interconnect sizing problem (*STIS*) in this paper. The solution to this problem can be used either as a global planning tool after initial floorplan, placement, and global routing to determine the sizes of global interconnects and inter-block drivers/buffers, or as a block-level optimization tool to compute the optimal sizes of all transistors, gates, and wires within a functional block. First, we shall briefly summarize the existing works on device and interconnect sizing. Most of them size device and interconnect independently, and many of them completely ignore the sizing of interconnects.

The *device sizing* problem includes both *gate sizing* and *transistor sizing* problems. The *gate sizing* problem usually assumes that all transistor sizes within each gate scale isotropically [16] (i.e., all transistor sizes increase or decrease by an uniform factor), or each gate has a discrete set of implementations (cells) with different driving capabilities pre-designed as a given cell library for performance optimization. The cell selection problem has been approached using a linear programming technique in [3], in which a continuous solution was first computed and then mapped into a discrete solution. The performance-area (or power consumption) trade-off for the gate sizing problem has been studied in [1] based on another linear programming formulation. The *transistor sizing* problem is more general in the sense that it computes the optimal size for *each* transistor (instead of a gate or cell), which may lead to more optimized designs, especially in full-custom layout. It was formulated as a posynomial programming problem under the RC delay model and solved by a sensitivity-based heuristics in *TILOS* [11]. Since a posynomial programming problem can be transformed into a convex programming problem, the convex programming technique was applied in [15] to obtain an exact solution at higher computational costs when compared with the sensitivity-based method.

Note that none of these above device sizing works dealt with interconnect sizing, although it was roughly pointed out by authors of [11] that their objective function can handle the cases where the wire width is taken into consideration. As the interconnect delay becomes dominant due to the advance of IC technology, the interconnect sizing problem, usually called the wiresizing problem, attracts much attention recently.

The general wiresizing problem was first introduced in [7, 8] where the interconnect is modeled as a distributed RC tree and the driver as a fixed-value resistor to minimize the Elmore delay between the unique source and a set of critical sinks for each net. The authors developed the first polynomial-time optimal wiresizing algorithm based on the *monotone property*, the *separability* and the *dominance property*. Later on, the wiresizing problem to minimize the maximum delay along a net is formulated as a posynomial program in [14], where a continuous solution is obtained by a sensitivity-based algorithm and then mapped into a discrete solution, while simultaneous topology-construction and wiresizing was studied in [13]. These papers assumed a single source for each net until very recently the optimal wiresizing problem for interconnects with multiple sources was formulated and solved optimally in [4]. This optimal multi-source wiresizing algorithm is applicable to the single-source problem as well, and runs much more efficiently when compared with the algorithm in [8]. All these works, however, did not consider the need to size the device again after interconnects have been changed.

Two recent studies consider both device and interconnect sizing. [5] formulated the simultaneous driver and wire sizing (*sdws*) problem to size a single routing tree driven by a chain of cascaded drivers, and revealed an important relationship between the driver sizing and the optimal wire sizing. The authors developed an efficient optimal algorithm, which achieved more substantial delay and power-consumption reduction when compared with separate driver and wire sizing methods. The authors of [16] studied the simultaneous gate and wire sizing problem using a more accurate circuit-level delay model, and use the sensitivity of gate/wire sizing to guide a gradient-based optimization technique. However, they assume all transistors in a gate are scaled isotropically, which often leads to suboptimal designs. Also, the global convergence and optimality are not guaranteed.

In this paper, we study the *simultaneous transistor and interconnect sizing (STIS) problem.* We formulate the *STIS* problem using a distributed RC circuit model which models the waveform-dependent transistor resistances, the distributed nature of the interconnects, and the transistor-interconnect interactions.

Rather than developing *ad hoc* method for our *STIS* formulation, we propose a systematic method based on the general *dominance property* defined in the following:

Definition 1 (Dominance Relation) *For two vectors* $\mathbf{X}$ *and* $\mathbf{X}'$, $\mathbf{X}$ *dominates* $\mathbf{X}'$ *(denoted as* $\mathbf{X} \geq \mathbf{X}'$*) if* $x_i \geq x'_i$ *for all* i.

Definition 2 (Local Refinement Operation) *The local refinement operation of a vector* $\mathbf{X}$*, with respect to any particular variable* x_i *and function* $f(\mathbf{X})$*, is to minimize* $f(\mathbf{X})$ *subject to only evaluating* x_i *while keeping the value of any* $x_j (j \neq i)$.

For simplicity, we will call the resulting vector the *local refinement* of $\mathbf{X}$, and the resulting value of x_i the *local optimum* of x_i (denoted as $\tilde{x}_i$).

Definition 3 (Dominance Property) *Let* $\mathbf{X}^*$ *be the optimal vector to minimize* $f(\mathbf{X})$*. If* $\mathbf{X}$ *dominates* $\mathbf{X}^*$*, a local refinement of* $\mathbf{X}$ *still dominates* $\mathbf{X}^*$*; If* $\mathbf{X}$ *is dominated by* $\mathbf{X}^*$*, a local refinement of* $\mathbf{X}$ *is still dominated by* $\mathbf{X}^*$

We study the dominance property for a large class of posynomial functions of the following form:

$$\begin{aligned} f(\mathbf{X}) &= \sum_{p,q,i,j} a_{ij}^{pq} \cdot \frac{x_j^q}{x_i^p} + \sum_{p,i} b_i^p \cdot \frac{1}{x_i^p} + \sum_{p,i} c_i^p \cdot x_i^p \qquad (1) \\ where \quad & x_i, x_j \in \mathbf{X} = (x_1, \cdots, x_N) \\ & p, q \in (1, 2, \cdots, P), \\ & a_{ij}^{pq},\ b_i^p \ and\ c_i^p \geq 0 \end{aligned}$$

Particularly, we study the cases where this function has certain symmetry:

1. $a_{ij}^{pq} > 0$ if and only if $a_{ji}^{qp} > 0$;

2. $b_i^p > 0$ only if $c_i^p > 0$ or $a_{ij}^{pq} > 0$ for any q and j;

3. $c_i^p > 0$ only if $b_i^p > 0$ or $a_{ij}^{pq} > 0$ for any q and j;

We proved the following theorems:

Theorem 1 *The dominance property holds for the above defined $f(\mathbf{X})$ if its coefficients are constants independent of $\mathbf{X}$.*

Theorem 2 *The dominance property holds for the above defined $f(\mathbf{X})$ if its coefficients are functions of x_i and x_j satisfying the following conditions: a_{ij}^{pq} monotonically increases with respect to x_i while monotonically decreases with respect to x_j, and b_i^p monotonically increases with respect to x_i while c_i^p monotonically decreases with respect to x_i.*

These results greatly generalize the *simple* dominance property in [8, 4], and lead to efficient algorithms with *guaranteed* convergence and optimality, which can be applied to many optimization problems in VLSI CAD and other domains. As an example, our *STIS* algorithm based on the general dominance property is much more efficient than the existing mathematical programming based methods such as convex-programming based transistor-sizing [15] and superior to the existing sensitivity-based heuristics used in many transistor or interconnect sizing works [11, 3, 14, 16] in terms of both convergence and optimality.

The remainder of this paper is organized as follows. In Section 2, we will formulate the *STIS* problem under the distributed RC model and the Elmore delay model, and show this formulation satisfied the dominance property even when considering the waveform-slope effect for transistor effective resistances. In Section 3, we will discuss other properties of optimal *STIS* solutions and present the optimal *STIS* algorithms. Section 4 will show the preliminary experimental results for both transistor sizing and simultaneous transistor and interconnect sizing, and Section 5 will conclude the paper with discussions of future work. The proof concerning the dominance property is given as an appendix. To the best of our knowledge, this is the first in-depth study of the simultaneous transistor and interconnect sizing problem.

2 Formulations

Because the transistor channel length is usually fixed when given a fabrication technology, we only allow the channel width to change in transistor sizing. In the following of this paper, we use *width* for both the channel width of a transistor and the wire width of an interconnect. For simplicity of presentation, we assume that only simple gates used in designs are to be optimized, such as AND, OR, NAND, NOR, XOR gates of two to four inputs.

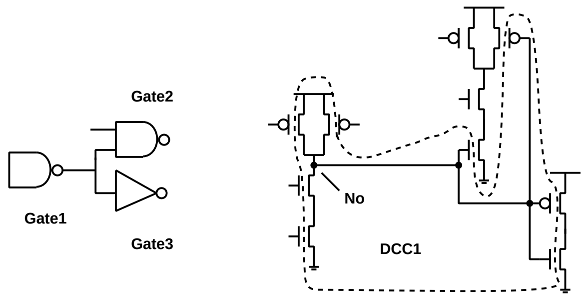

Figure 1: (a) the gate-level circuit diagram where gate2 and gate3 are fanout of gate1. (b) the transistor-level circuit diagram where gate1 and the routing tree linked to its output comprise a *DC-Connected-Component* DCC_1. The output of gate1, denoted as N_o, is the root of the routing tree. The inputs of gate2 and gate3 are sinks of the routing tree.

Let a critical path of a circuit ckt be an acyclic path with the maximum delay from a primary input (or a register output) to a primary output (or a register input) of ckt, and $T(ckt)$ be the delay of a critical path in ckt. We have the following formulation.

Formulation 1 *Given a circuit ckt, a time specification T_{spec} for ckt and lower and upper bounds for the width of each transistor/interconnect, the* STIS *problem is to find an optimal width for each transistor/interconnect such that the critical delay $T(ckt) \leq T_{spec}$.*

A possible framework for the *STIS* problem is as follows: First, a timing analysis is performed to obtain two path sets **CP1** and **CP2**, where **CP1** contains all paths with delay larger than T_{spec} and **CP2** contains paths with delay larger than $T_{spec} \cdot (1 - \epsilon)$ with $\epsilon \in [0, 1]$. If **CP1** is empty, sizing is not necessary. Otherwise, all paths in **CP2** (not just **CP1**) are regarded as critical and both transistors and interconnects along these paths will be sized to tune the path delays.

Since a non-critical path may become critical after sizing critical paths, several iterations of timing analysis and sizing procedure may be carried out before the timing specification is satisfied or failed at last. The parameter ϵ can be adjusted to reduce the total number of iterations required. Note that it is straightforward to apply our algorithm to cases where there is a specific delay requirement for each path.

2.1 Circuit Model

Because an MOS transistor is a voltage-controlled device, an MOS circuit can be partitioned into a number of *DC-Connected Components (DCCs)* [19], which is connected by DC-current path and the DC current can not cross the boundary of a DCC. In general, each DCC consists of both a gate may including pass transistors and a routing tree connecting the output (denoted as N_o) of the gate to a set of inputs of its fanout gates (see Figure 1). We shall use distributed RC tree to model a DCC.

A. Device model

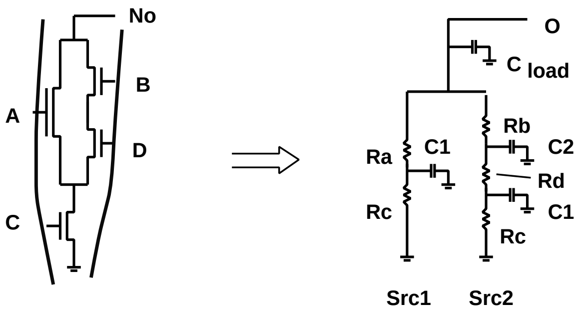

Figure 2: The worst case delay through pulldown network of a CMOS gate is modeled with the maximum Elmore delay among multi-RC paths.

A transistor can be modeled by its source-drain *effective* resistance and the gate, source and drain capacitances. In a simple *step-model*, the resistance and capacitances are computed without consideration of the input/output waveforms. The effective transistor resistance is inversely proportional to the transistor width whereas all these capacitances are proportional to it. In a more accurate *slope-model*, the effective resistance also depends on its input waveform slope and its loading capacitance. It was shown in [17] that the *slope-model* is 15% more accurate than the *step-model*. Our methods apply to both *step-model* and *slope-model*. But for simplicity of presentation, we shall use only the *step-model* in our formulation until Section 2.3.B.

Consider the pulldown (pullup) transistor network of a gate, the worst case delay occurs when the discharging (charging) current goes through a single-transistor chain, defined to be a chain of transistors from the ground (or voltage source Vdd) to N_o, the output of the gate. Since a single-transistor chain is a distributed RC path, its delay can be given by the Elmore delay model [10] through this path and we shall consider only those resistances and capacitances in the path. During our STIS optimization, only those single-transistor chains in the set of critical paths or near-critical paths are considered.

The pulldown network of a CMOS gate is shown in Figure 2. One of the two marked single-transistor chains may be active (for different critical paths of a circuit). For example, chain $C - D - B$ is modeled by the RC path $R_c - c_1 - R_d - c_2 - R_b - C_{load}$ with delay $R_c \cdot (c_1 + c_2 + C_{load}) + R_d \cdot (c_2 + C_{load}) + R_b \cdot C_{load}$. We duplicate the resistance (i.e. R_c) and capacitance (i.e. c_1) which belong to the two paths for simplicity of presentation.

We shall create (duplicate) a source (either the GND or the Vdd in designs) for each unique single-transistor chain [1]. The number of chains in a gate will increase with respect to the number of transistors in the gate, but the number is well under control for most gates with reasonable complexity.

Let $P(src_s, N_k)$ denote the RC path corresponding to a single-transistor chain between source src_s and node N_k, and c_k^s denote the total capacitance at node N_k due to source/drain capacitances of all transistors linked to N_k. Let $R(P(src_s, N_k))$ be the total resistance of path $P(src_s, N_k)$ and $\mathcal{C}_o$ be the total capacitance due to

[1] Extremely complex gates/cells (such as a multiplier) have to be decomposed before using this approach

the routing tree and its sinks, the Elmore delay between source src_s and node N_o, denoted $t(P(src_s, N_o))$, is

$$t(P(src_s, N_o)) = \sum_{N_k \in P(src_s, N_o)} R(P(src_s, N_k)) \cdot c_k^s \; + \; R(P(src_s, N_o)) \cdot \mathcal{C}_o \tag{2}$$

Note that we can simultaneously size several single-transistor chains in a gate if they are all critical, which is different from [15] where only a maximum delay path between node N_o and a source is considered each time.

B. Interconnect model

A routing tree is modeled similarly as in [8, 4]. We refer to N_o, a sink or a Steiner point in the routing tree as a node. A *segment* connects two *node*s in the routing tree. In order to capture the distributed resistive effect of interconnects, each segment is divided into a sequence of edges and each edge is treated as a π-type RC circuit. Thus, a routing tree is modeled by a distributed RC tree. Note that we assume that the wire width within an edge does not change. Thus, such edge division controls how aggressively we perform wiresizing optimization.

We treat node N_o as the root of the routing tree. Consider node N_k in the routing tree and let T_k denote the subtree rooted at node N_k (including node N_k). We denote the total capacitance in T_k as $\mathcal{C}_k$, including the total wire capacitances within T_k, the total transistor-gate capacitances of sinks within T_k, the total loading capacitances of *PO*s (primary outputs of the circuit, do not exist in every net) and the total via capacitances (if a via is modeled by a capacitance) within T_k. Let r_k be the resistance of the edge E_k with downstream node at N_k, the delay $t(P(N_o, sink_t))$ between node N_o and sink $sink_t$ along the path $P(N_o, sink_t)$ is

$$t(P(N_o, sink_t)) = \sum_{N_k \in P(N_o, sink_t)} r_k \cdot \mathcal{C}_k \tag{3}$$

The distributed RC model for an *DCC* is shown in Figure 3. It is a *multi-source RC tree (MSRT)* which inherits the sources from the gate and the sinks from the routing tree. If src_s is a source of the *MSRT* and $sink_t$ is a sink of the *MSRT*, path $P(src_s, sink_t)$ is called a *stage*.

2.2 Delay Formulation

A. Resistance and Capacitance Calculations

Let r_{c0} be the source-drain resistance of a unit width transistor with the fixed length and x be the width of a transistor, then its source-drain resistance $r_c = r_{c0} \cdot x$. The following capacitances will be considered for a transistor [22]. Let c_{ox} be the "thin oxide" capacitance per unit area, c_{gd} be the gate-drain overlap capacitance per unit width, c_{gs} be the gate-source overlap capacitance per unit width and c_{gb} be the gate-bulk overlap capacitance per unit length. Then, the drain capacitance $c_d = c_{gd} \cdot x$, the source capacitance $c_s = c_{gs} \cdot x$, and the gate capacitance $c_g = c_{ox} \cdot l \cdot x + (c_{gd} + c_{gs}) \cdot x + c_{gb} \cdot 2l = c_{g0} \cdot x + c_{g1}$, where l is the fixed channel length, $c_{g0} = (c_{ox} \cdot l + c_{gd} + c_{gs})$ and $c_{g1} = c_{gb} \cdot 2l$.

Let $r_\square$ be the sheet resistance of a wire, c_0 be the capacitance per unit wire area and c_1 be the total fringing capacitance per unit wire length. Then, for a unit length wire E of width x, its resistance is $r_E = r_\square \cdot \frac{1}{x}$ and

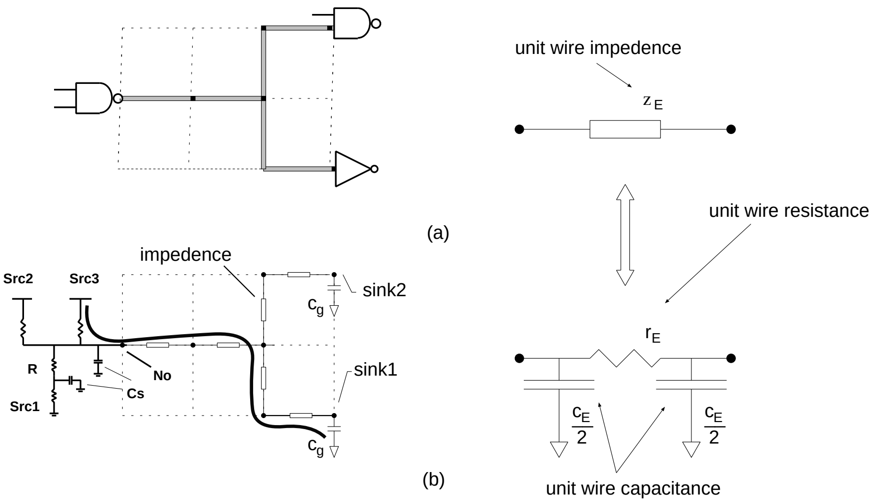

Figure 3: The distributed RC model for a DCC with darken mark for a stage from source src_3 to sink $sink_1$. (a) Layout for DCC_1 in Figure 1. (b) MSRT model for DCC_1. The sink capacitance c_g is the total gate capacitance of all transistor (in the fanout DCCs) linked to a sink. The c_s is the total source/drain capacitance of all transistors linked to a node. Each edge in the routing tree is modeled as a π-type RC circuit to consider the wire resistance, wire area capacitance, and wire fringing capacitance.

its capacitance is $c_E = c_0 \cdot x + c_1$. Both r_E and c_E are used in the π-type RC circuit model for wire E (see Figure 3).

B. Coefficient Functions

Let M_i and M_j be transistors, E_i and E_j be wire edges and c^l be either a loading capacitance of a primary output, a via capacitance or gate-bulk capacitance of a transistor. For simplicity, we use $r_\square(i)$ for both sheet resistance for wire edge E_i and *unit-width* drain-source resistance $r_{c0}(i)$ for transistor M_i. We define the following coefficient functions for stage $P(src_s, sink_t)$:

$$f_0^{st}(i,j) = \begin{cases} r_\square(i) \cdot c_{g0}(j) & \text{if } M_i/E_i \textit{ charges only the gate of } M_j \\ r_\square(i) \cdot c_d(j) & \text{if } M_i/E_i \textit{ charges only the drain of } M_j \\ r_\square(i) \cdot c_s(j) & \text{if } M_i/E_i \textit{ charges only the source of } M_j \\ r_\square(i) \cdot (c_d(j) + c_s(j)) & \text{if } M_i/E_i \textit{ charges both the drain and the source of } M_j \\ r_\square(i) \cdot c_0(j) & \text{if } M_i/E_i \textit{ charges } E_j \\ 0 & \text{otherwise} \end{cases}$$

$$f_1^{st}(i,j) = \begin{cases} r_\square(i) \cdot c_1(j) & \text{if } M_i/E_i \textit{ charges } E_j \\ r_\square(i) \cdot cg_1(j) & \text{if } M_i/E_i \textit{ charges } M_j \\ 0 & \text{otherwise} \end{cases}$$

$$g^{st}(i) = \begin{cases} r_\square(i) \cdot \sum c^l & \text{if } M_i/E_i \textit{ charges } c^l \\ 0 & \text{otherwise} \end{cases}$$

$$h_0^{st}(i) = \begin{cases} r_\square(i) \cdot \frac{c_0(i)}{2} & \text{if } E_i \in P(src_s, sink_t) \\ 0 & \text{otherwise} \end{cases}$$

$$h_1^{st}(i) = \begin{cases} r_\square(i) \cdot \frac{c_1(i)}{2} & \text{if } E_i \in P(src_s, sink_t) \\ 0 & \text{otherwise} \end{cases}$$

Given the logic design and the interconnect topology of a circuit, all these functions can be computed and stored in arrays for later using in the sizing procedure. Note that the definitions can easily handle multiple transistor types and multiple routing layers, and all these functions can be evaluated within each DCC. I.e., most coefficient functions equal to zero for a circuit containing multiple DCCs. Note also that in the *step-model*, $r_\square$ for transistor is a constant whereas $r_\square$ is a variable in the *slope-model*. Therefore, these coefficient functions are constants independent of x in the step model and are variables in terms of x in the *slope-model*.

C. Delay of a stage

Using these coefficients, the Elmore delay $t(P(src_s, sink_t))$ of stage $P(src_s, sink_t)$ can be written as:

$$
\begin{aligned}
t(P(src_s, sink_t), \mathbf{X}) &= \sum_{N_k \in P(src_s, N_o)} (R(P(src_s, N_k)) \cdot c_k^s) \; + \; R(P(src_s, N_o)) \cdot \mathcal{C}_o \; + \sum_{N_k \in P(N_o, sink_t)} r_k \cdot \mathcal{C}_k \\
&= \sum_{i,j} f_0^{st}(i,j) \cdot \frac{x_j}{x_i} \cdot l_i \cdot l_j \; + \; \sum_{i,j} f_1^{st}(i,j) \cdot \frac{1}{x_i} \cdot l_i \cdot l_j \; + \; \sum_i g^{st}(i) \cdot \frac{l_i}{x_i} \; + \\
&\quad \sum_i h_0^{st}(i) \cdot l_i^2 \; + \; \sum_i h_1^{st}(i) \cdot \frac{l_i^2}{x_i}
\end{aligned}
\tag{4}
$$

where $l_i = 1$ for a transistor M_i. An interesting observation is that the coefficient function definitions are still valid when the wires inside a gate are also taken into account and the delay of a stage still has the form of (4).

2.3 STIS to Minimize Delay for Multiple Critical Paths

A. Formulation under the step-model

In order to simultaneously minimize delay along multiple critical paths in circuit ckt, we introduce the following weighted delay objective:

$$
t(ckt) = \sum_{P(src_s, sink_t) \in ckt} \lambda^{ij} \cdot t(P(src_s, sink_t)) \tag{5}
$$

where the penalty weight λ^{ij} indicates the criticality of stage $P(src_s, sink_t)$.

The criticality of a stage is given by the timing analysis. A stage is critical if in a critical path and its weight is 1, otherwise, its weight is 0. Note that other criticality scheme can be used with our formulation as well.

Let

$$
\begin{aligned}
F_0(i,j) &= \sum_{P(src_s, sink_t) \in ckt} \lambda^{st} \cdot f_0^{st}(i,j) \\
F_1(i,j) &= \sum_{P(src_s, sink_t) \in ckt} \lambda^{st} \cdot f_1^{st}(i,j) \\
G(i) &= \sum_{P(src_s, sink_t) \in ckt} \lambda^{st} \cdot g^{st}(i) \\
H_1(i) &= \sum_{P(src_s, sink_t) \in ckt} \lambda^{st} \cdot h_1^{st}(i).
\end{aligned}
$$

Eliminating those terms independent of transistor/interconnect widths, (5) becomes

$$t(ckt, \mathbf{X}) \;=\; \sum_{i,j} F_0(i,j) \cdot \frac{x_j}{x_i} \cdot l_i \cdot l_j \;+\; \sum_{i,j} F_1(i,j) \cdot \frac{1}{x_i} \cdot l_i \cdot l_j \;+\; \sum_i G(i) \cdot \frac{l_i}{x_i} \;+\; \sum_i H_1(i) \cdot \frac{l_i^2}{x_i} \qquad (6)$$

With respect to the equation (6), we define the following *STIS* problem to minimize delay through multiple critical paths:

Formulation 2 *Given a circuit ckt and the lower and upper bound for the width for each transistor/interconnect, the STIS problem is to determine an optimal width for each transistor/interconnect (or equivalently, a sizing solution* $\mathbf{X}$*) such that the weighted delay through multiple critical paths given by Eqn. (6) is minimized.*

Because the *step-model* assumes that all those coefficients given in Section 2.2 are constants independent of the transistor/interconnect widths, we have the following corollary of Theorem 1:

Corollary 1 *The dominance property holds for the STIS problem (6) under the step-model.*

In practice, it is often the case that we want to size the transistors and wires with no increase in the layout area (using the *free* space in the current layout) or with bounded increase in the layout area. Therefore, there may be an upper bound associated with each transistor and wire during the optimization. On the other hand, there may be a lower bound associated with each device and wire due to the technology feature sizes and reliability concerns (mainly electro-migration). Thus, there are an upper bound and a lower bound for the sizing solution and our optimal *STIS* algorithm can easily handle these constraints. It will be seen later on that the lower and upper bounds are the starting point for our *STIS* algorithms. Since the transistor/interconnect widths may change considerably during the post-layout optimization, several iterations of placement adjustment together with re-routing and *STIS* procedure may need to be carried out.

B. Extension to Slope-Model

The *step-model* assumption was implicitly made in [8, 5, 14, 4]. For better accuracy, we shall use the *slope-model* for the *STIS* problem. Although these coefficients concerning the $r_\square$'s of transistors are no longer constants under the *slope-model*, all formulations are still valid by just treating these coefficients as functions of waveform slopes.

The *slope-model* similar to those in [18, 17] will be used. For a transistor, the source-drain effective resistance is determined by its width, input waveform slope (roughly, $\frac{rising-time}{voltage-range}$) and capacitance loading. In short, the gate capacitance of a transistor is part of the loading of its previous stage, and its input slope is a monotonically increasing function of, or *simply*, the delay along the previous stage. When the width x_i of a transistor M_i is increased, all gate capacitance, input waveform slope and effective $r_\square$ of M_i will be increased. Thus, the effective $r_\square$ of M_i is monotonically increasing with respect to x_i, which means $F_0(i,j), F_1(i,j), G(i)$ and $H_1(i)$ are monotonically increasing with respect to x_i. According to Theorem 2

Corollary 2 *The dominance property holds for the STIS problem (6) under the slope-model.*

Because the *dominance property* holds for the *STIS* problem, the *local refinement* can be used iteratively to tighten the upper and lower bounds for the optimal width of each transistor/interconnect. We shall study the optimal *STIS* solution properties, and then present the optimal algorithms for the *STIS* problem in the next section.

3 Properties and Algorithms

3.1 Optimal STIS Solution Properties

In this section, we shall study the optimal *STIS* properties, which are similar to those in [8, 4] although we have different formulations and take the slope model into account.

Theorem 3 (Separability) *Given the widths for all transistors in an DCC and the wires in a path P originated from the root of the routing tree, the optimal wire width assignment for each subtree branching off from P can be carried out independently.*

Different from that in [4], both the driver resistances and the loading capacitances are variables in our formulations. In general, the separability may not hold when the transistor sizes are not fixed.

Theorem 4 (Monotone Property) *There exists an optimal wiresizing solution as a part of the optimal STIS solution, such that the wire widths in any path from the root of a routing tree to one of its sinks are monotonically decreased.*

An assumption is implicitly made about Theorem 4 that the lower and upper bounds for the wire-widths for the whole routing tree are uniform. If the lower and upper bounds are uniform only for each segment in the routing tree, the optimal wire-widths will be monotone only within each segment.

3.2 General Algorithm Schemes

A. LRA Scheme for Lower and Upper Bounds Computation

For an optimization problem $f(\mathbf{X})$ which satisfies the dominance property, let $\underline{\mathbf{X}}$ be the lower bound of its definition domain and $\overline{\mathbf{X}}$ be the upper bound. Since its optimal solution $\mathbf{X}^*$ is bounded by $\underline{\mathbf{X}}$ and $\overline{\mathbf{X}}$, a simple algorithm scheme, the *Local Refinement Algorithm (LRA)* scheme in Table 1, can be used to tighten lower and upper bounds for $\mathbf{X}^*$.

In order to tighten the lower bound, the *LRA* scheme begin with $\underline{\mathbf{X}}$ and traverse the vector in certain order to perform a *local refinement* on each x_i whenever possible. This process is repeated until no improvement is achieved on any x_i in the last round of traversal and we obtain a *tight* lower bound.

An upper bound can be tightened in the similar way by performing local refinements beginning with $\overline{\mathbf{X}}$. If the result lower and upper bounds meet, which is very likely in engineering applications we have studied, we

Local Refinement Algorithm for Lower Bound Based on Dominance Property

```
Function LRA(X̲)
   /*
    *    Given a lower bound X̲, return the tight lower bound.
    *
    */

   X ← X̲;
   repeat
       progress ← false;
       for each x_i of X do
           x ← local_refinement(X, x_i);
           if x ≠ x_i then
               progress ← true;
               x_i ← x;
           end if
       end for;
   until progress = false;
   return X;
end Function;
```

Table 1: The local refinement algorithm based on the dominance property. The upper bound can be tightened similarly.

obtain the optimal solution immediately. Otherwise, other optimization methods can be applied to search for the optimal solution in the much reduced solution space.

B. BRA Scheme for Lower and Upper Bounds Computation

Different from the *local refinement* which only tighten the lower or upper bound for an element $x_i \in \mathbf{X}$, a *bundled refinement* which can tighten the lower or upper bounds for *multiple* elements of $\mathbf{X}$ by a single operation will be described in this section, then the *Bundled Refinement Algorithm (BRA)* scheme for lower and upper bounds computation will be presented as well.

The *bundled refinement* is defined for an interval which is monotonic in $\mathbf{X}^*$. If $[p, q]$ is such a monotone interval (see Figure 4), without loss of generality, we assume that $x_i^* \geq x_j^*$ for $i < j$ and $i, j \in [p, q]$. In order to tighten the lower bound for $\mathbf{X}^*$, we can perform *local refinement* on x_q then use the *local optimum* $\tilde{x}_q$ as a new lower bound for *all* x_i's such that $i \in [p, q]$. This operation is called *bundled refinement for lower bound (BRL)*, which is justified by this fact that $\tilde{x}_q \leq x_q^* \leq x_i^* (i \in [p, q])$.

Similar strategy can be used to tighten the upper bound, i.e., x_p is locally refined and the *local optimum* $\tilde{x}_p$ is regarded as the upper bound for *all* $x_i (i \in [p, q])$. It is called *bundled refinement for upper bound (BRL)*.

The *bundled refinement algorithm (BRA)* may contain multiple *passes*, each just like the *LRA* scheme. In a pass, each interval which is monotonic in $\mathbf{X}^*$ is traversed by *BRL* until no improvement can be obtained in the last traversal, then it is similarly traversed by *BRU*. After a pass, the lower and upper bounds are checked. A divergent interval is the one still with different lower and upper bounds. Each divergent interval is divided

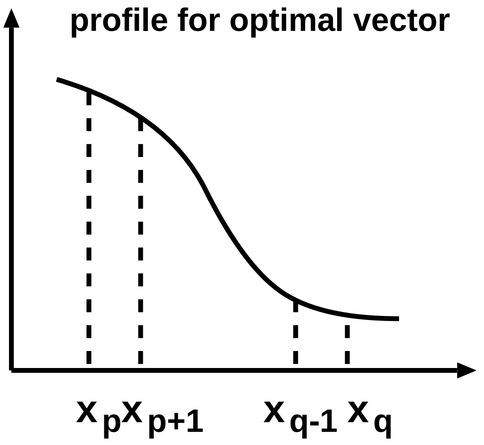

Figure 4: The profile of $\mathbf{X}^*$ with a local monotone interval where the lower bound of x_q is a lower bound for all x_i's ($i \in [p, q]$). and the upper bound of x_p is an upper bound for all x_i's ($i \in [p, q]$),

into two intervals and the lower and upper bounds are passed to the "children". Another pass of *bundled refinement* can be carried out for all such "child" intervals, because they are monotonic in $\mathbf{X}^*$ as well. Such pass iteration will be stopped if lower and upper bounds meet everywhere, or each divergent interval contains only one $x_i \in \mathbf{X}$.

C. Convergence and Optimality

We use $\mathcal{A}$-tight bound to refer to a lower or upper bound of $\mathbf{X}^*$ if it can not be tightened any more by an algorithm $\mathcal{A}$. There may be multiple $\mathcal{A}$-tight lower or upper bounds for a function $f(\mathbf{X})$ when $\mathcal{A}$ is not optimal.

The convergence of algorithm $\mathcal{A}$ means how fast an $\mathcal{A}$-tight bound can be achieved, and the optimality means how far an $\mathcal{A}$-tight lower/upper bound is away from the optimal solution. It can be measured by the gap between the $\mathcal{A}$-tight lower and upper bounds.

We can prove the following theorem concerning the optimality of the LRA and BRA schemes.

Theorem 5 *BRA-tight bounds are also LRA-tight.*

Therefore, we use the BRA scheme whenever possible since it converges much faster and produces equal-quality solutions when compared with the LRA scheme.

3.3 Optimal STIS Algorithm

A. Implementation of the slope-model

Although the transistor effective resistance depends on its size, the input waveform slope and the capacitance loading, it can be determined *alone* by the *slope ratio* as proposed in [18, 17]. Let the *intrinsic output rising-time* be the rising-time which would occur at the output when a step waveform drives a transistor, the *slope* *ratio* is defined to be the *input rising-time* over the *intrinsic output rising-time*. Thus, we can build a table, with one rather than three dimensions, for each type of transistors based on SPICE simulation results.

A table is built for a transistor with fixed-length, unit-width and a unit capacitance loading. Each raw has three columns. The first column is the *slope ratio*, the second one is the effective resistance ($r_\square$) and the third one is the *output rising-time*. The first raw is for step input, thus its $r_\square$ is that in the *step-mode* and its *output rising time* is the *intrinsic output rising-time*.

Let transistor M_i drives transistor M_j and their widths be x_i and x_j, respectively. If the *output rising-time* for M_i is $outS(i)$ according to the table and $C_{load}(i)$ is its capacitance loading, the *input rising-time* for M_j is $outS(i) \cdot \frac{C_{load}(i)}{x_i}$ because $outS(i)$ is just for the unit channel-width and unit capacitance-loading case. If the *output rise-time* in the first raw for M_j is $outS0(j)$ and $C_{load}(j)$ is its capacitance loading, the *intrinsic output rising-time* for M_j is $outS0(j) \cdot \frac{C_{load}(j)}{X_j}$. Thus, the $r_\square$ for M_j can be determined by the *slope ratio* $\frac{outS(i) \cdot \frac{C_{load}(i)}{x_i}}{outS0(j) \cdot \frac{C_{load}(j)}{X_j}}$. According to Corollary 2, the *local refinement* can be carried out for transistor M_j with the $r_\square$ under the current width x_j. After a *local refinement*, the result width will be used to compute the new $r_\square$ and to update the coefficient functions.

B. Optimal Wiresizing Algorithm OWBR

Let δ be the minimum wire length permitted by the design rules. The *STIS* problem requires that the optimal wire width be determined for each wire with length δ. According to Theorem 4, each segment is a monotone interval in the optimal solution $\mathbf{X}^*$. Thus, the *bundled refinement* can be used within each segment, and the optimal wiresizing algorithm using the BRA scheme (called OWBR) is shown below:

Rather than dividing the routing tree into edges of length δ, a recursively refined edge division scheme is applied to achieve the required wiresizing accuracy for much less computational costs. In the first pass, each edge is just a segment in the routing tree and the *bundled refinement* are carried out iteratively through the routing tree to tighten the lower and upper bounds of the optimal wiresizing solution.

After a pass, each edge still with different lower and upper bounds will be divided into two edges. Then, another pass of bundled refinement followed by edge division refinement is iteratively carried out until the upper and lower bounds meet everywhere or each divergent edge is just δ long. More details about the *OWBR* algorithm can be found in [4]. In fact, the *OWBR* is a special version of the *BRA* scheme to take advantage of the simplicity of the Elmore delay model.

The *OWBR* algorithm is shown in [4] to run more than 100x faster and still achieve the upper and lower bounds same tight as those given by the *local refinement* algorithm under the edge division with unit of δ. All analyses concerning the *OWBR* algorithm in [4] are still true under our *STIS* formulations.

C. Overview of Optimal STIS algorithm

The optimal STIS algorithm includes two parts, the *BRA* scheme to tighten upper and lower bounds and the enumeration algorithm with the feature of dynamic programming to achieve the optimal solution between the tight upper and lower bounds.

The *STIS* algorithm works as follows. First, a circuit together with the interconnect topology is partitioned into *DCC*s and the coefficient functions will be computed within each DCC and before the sizing procedure.

Then, the lower and upper bounds of the optimal sizing solution is computed by the *BRA* scheme. Note that a transistor M_i can be viewed as a "segment" with length δ in a routing tree so that then the *OWBR* algorithm can be applied just as in a single routing tree in [4]. The initial lower bound is determined by the technology feature sizes and the reliability considerations and the initial upper bound is based on the chip area specification. The lower and upper bounds will be tightened iteratively.

The optimal sizing solution may be achieved by obtaining the identical lower and upper bounds. Otherwise, the enumeration procedure will be carried out. According to Theorems 3 and 4, we first enumerate the width assignments for transistors, then the *OWSA* algorithm in [8], which is a polynomial-time dynamic programming method, can be used to compute an optimal wiresizing solution between the lower and upper bounds for each wire.

4 Experimental Results

We have implemented the optimal *STIS* algorithm in ANSI C for the Sun SPARC station environment and tested our algorithm on a number of examples for the advanced IC technologies, including several critical nets with their drivers extracted from the layout of an Intel high performance microprocessor.

In this section, we shall present the comparison of different sizing solutions. The delays to be reported in the following are computed using HSPICE. The use of HSPICE simulation results, instead of calculated Elmore delay values, not only shows the quality of our *STIS* solutions, but also verifies the validity of our transistor/interconnect modeling and the correctness of our *STIS* problem formulation.

Parameters based on MCNC $0.5\mu m$ CMOS process technology are used in the optimal *STIS* algorithms and HSPICE. The wire width choices are $\{ W_1, 2W_1, 3W_1, 4W_1, 5W_1\}$, where W_1 is the minimum wire width ($0.95\mu m$ in the $0.5\mu m$ IC technology). The transistor widths used are $\{0.5\mu m, 0.6\mu m, \cdots, 2000.0\mu m\}$ with step of of $0.1\mu m$, which is determined by the design rules. Note that the *local refinement* does not depend how feasible widths are defined, thus our algorithms are applicable to any width scheme for wires and transistors.

4.1 SDWS

The *simultaneous driver and wire sizing (sdws) problem* is first used to show the effectiveness of the *slope-model*. Cascade drivers are used to drive a $1pF$ capacitance through an $1cm$ interconnect in IC technology. We relax the constraint in real designs that the total stage number shall be even because we just want to compare the modeling approaches. Three models are used (see Table 2), including the *fixed-model*, the *step-model* and the *slope-model*. The *fixed-value* model assumes a fixed-resistance for a driver, just the one used in previous works [8, 5, 4, 14], while the latter two are used in this paper for better modeling of transistors. Three configurations are used: five-stage cascade inverters, four-stage cascade inverters and three-stage cascade inverters. According to this table, the *fixed-model* and the *step-model* perform similarly in case of inverters. When the stage number is large, the *slope-model* outperforms the other two significantly and more than 60% extra delay reduction is observed. When the total stage number is small, the *fixed-model* and the *step-model* behavior like the *slope-model* because we assume a step-waveform in the input of the first stage, the waveform is still near step in this case. If we consider the transistors/interconnects together in a large netlist, the *slope-model* shall be of increasingly importance because multi-stages of transistors and interconnects tend to increase the slope of waveforms.

Model	FET Type	FET Width Stage1	Stage2	Stage3	Stage4	Stage5	Average Wire Width	Total Delay
fixed	NMOS	1.0	4.3	19.2	84.4	370.4	4.733	1.6358ns
	PMOS	2.0	8.6	38.4	168.8	740.8		
step	NMOS	1.0	4.0	17.4	75.4	327.1	4.571	1.5935ns
	PMOS	2.0	9.4	40.8	177.0	767.6		
slope	NMOS	1.0	6.7	48.6	232.5	720.6	4.713	0.6543ns
	PMOS	2.0	15.8	116.6	557.9	1730.0		
fixed	NMOS	1.0	6.3	40.2	254.8	-	4.665	0.6513ns
	PMOS	2.0	12.6	80.4	509.6	-		
step	NMOS	1.0	5.8	36.1	224.9	-	4.455	0.6434ns
	PMOS	2.0	13.6	84.7	527.7	-		
slope	NMOS	1.0	8.0	68.0	384.0	-	4.611	0.6334ns
	PMOS	2.0	18.7	163.2	921.4	-		
fixed	NMOS	1.0	11.6	126.0	-	-	4.471	0.6474ns
	PMOS	2.0	23.2	252.0	-	-		
step	NMOS	1.0	10.4	117.0	-	-	4.111	0.6533ns
	PMOS	2.0	24.4	274.5	-	-		
slope	NMOS	1.0	12.0	153.2	-	-	4.286	0.6455ns
	PMOS	2.0	28.1	367.4	-	-		

Table 2: Comparison between the *fixed-model*, the *step-model* and the *slope-model* in case of *sdws*.

4.2 Transistor Sizing

The *dominance property* was only used for the optimal wiresizing problem in the past. In this experiment, we design the following testing set to show the efficiency of our *STIS* algorithm based on the *dominance property* with the *transistor sizing problem*. The full adders from 2-bit to 16-bit are sized by assuming that all transistors in these circuits are critical. Although we use an extreme upper bound (2000.0 μm) for each transistor, the performance of the *local refinement* is still quite good. In Table 3, *nFET* is the total transistor number in a circuit, *average nLoc* is the average number of *local refinement* operations for a transistor, the *Time* is the total CPU time to size a circuit and *average width* is the average channel width for all transistors in the circuit. A transistor usually reaches its optimal width after 6–8 *local refinement* operations, and the time to size 514 transistors is just 19.78 seconds in a Sparc-10 workstation. Note that the $F_0(i,j) = 0$ and $F_1(i,j) = 0$ for E_i/M_i and E_j/M_j in different DCCs, which can be exploited to further speed up our *STIS* program in the future. The critical delay reduction by the optimal transistor sizing is up to 23.7%.

Adder	nFET	Average nLoc	Time (s)	Average Width	Critical Delay (ns)	
					min_width	opt_width
2bit	66	6.82	0.22	2.030	0.6021	0.4597 (-23.7%)
4bit	130	7.55	1.23	1.923	1.3511	1.0592 (-21.6%)
8bit	258	7.90	4.40	1.868	2.8681	2.4440 (-14.8%)
16bit	514	8.07	19.78	1.840	5.8878	4.6227 (-21.5%)

Table 3: Critical delay comparison between the minimum-width solution and the optimal-width solution.

4.3 STIS

A 4-bit adder is used to driven a 1cm wire in IC technology. Three sizing schemes, i.e., minimum transistor/interconnect width, transistor sizing alone, and simultaneous transistor and interconnect sizing are used (see Table 4). The *STIS* achieves 27.56% delay reduction.

	Total FET Area	Total Wire Area	Critical Delay
min FET + min Wire	828	1000	2.3159ns(0.00%)
opt FET + min Wire	1119	1000	2.0012ns(-13.58%)
STIS	1268	4493	1.6776ns(-27.56%)

Table 4: Critical delay comparison among minimal sizing scheme, transistor sizing alone, and the STIS solution.

5 Conclusions

In this paper, we study the *simultaneous transistor and interconnect sizing (STIS) problem.* We formulated the *STIS* problem using a distributed RC circuit model with consideration of the slope effects for transistors. We showed a general dominance property for a large class of posynomial functions (Theorem 1 and 2) and developed efficient optimal algorithms based on recursive *local refinement* and *bundled refinement* for optimizing such objective functions. The preliminary experiments for both transistor sizing and simultaneous transistor and interconnect sizing have been shown. To our knowledge, this is the first in-depth study of the simultaneous transistor and interconnect sizing problem.

Note that current experiments are carried out under this assumption that all paths in a circuit are critical. Our algorithm can only size critical ones of these paths to achieve more delay reduction and less area penalty if timing critical information is given. We plan to integrated our STIS algorithm with a timing analysis tool like Crystal [17] and test more circuits in the future.

Acknowledgments

This work is partially supported by ARPA/CSTO under contract J-FBI-93-112, the NSF Young Investigator Award MIP-9357582 and a grant from Intel Corporation under the NYI matching award program. The authors would like to thank Professor C.-J. Richard Shi at University of Iowa, Kei-Yong Khoo and Cheng-Kok Koh at UCLA for their helpful discussions and assistance.

Appendix Proof of the Dominance Property

For simplicity, we only prove the following simpler and less general case (Theorem 6). It is easy to prove Theorems 1 and 2 similarly.

Theorem 6 *The dominance property holds for the following function (7) if its coefficients are constants or the coefficient a_{ij}^{pq} monotonically increases with respect to x_i while monotonically decreases with respect to x_j.*

$$f(\mathbf{X}) \;=\; \sum_{p=1}^{M}\sum_{q=1}^{M}\sum_{i=1}^{N}\sum_{j=i}^{N} a_{ij}^{pq}\cdot\frac{x_j^q}{x_i^p} \tag{7}$$

$$\begin{aligned} \textit{where} \quad & M \textit{ and } N \textit{ are any integers} \\ & \mathbf{X} \;=\; (x_1, x_2, \cdots, x_N) \\ & a_{ij}^{pq} \;>\; 0 \end{aligned}$$

Proof: Respect to any particular x_i^p, define the follows:

$$\begin{aligned} \Psi_p(\mathbf{X},\mathbf{i}) &= \sum_{q\neq p}\sum_{j\neq i} a_{ji}^{qp}\cdot\frac{1}{x_j^q} \\ \Phi_p(\mathbf{X},\mathbf{i}) &= \sum_{q\neq p}\sum_{j\neq i} a_{ij}^{pq}\cdot x_j^q \\ \Theta(\mathbf{X},\mathbf{i}) &= \sum_{q}\sum_{r}\sum_{j\neq i}\sum_{k\neq i} a_{jk}^{qr}\cdot\frac{x_k^r}{x_j^q} \end{aligned}$$

Note that for any pair of vectors $\mathbf{X}$ and $\mathbf{X}'$ such that $\mathbf{X}'$ dominates $\mathbf{X}$, we have $\Psi_p(\mathbf{X},\mathbf{i}) \geq \mathbf{\Psi_p}(\mathbf{X}',\mathbf{i})$ and $\Phi_p(\mathbf{X},\mathbf{i}) \leq \mathbf{\Phi_p}(\mathbf{X}',\mathbf{i})$.

Ψ_p's, Φ_p's and Θ_p's are regarded as constants during local refinement operation with respect to x_p and the objective function (7) can be written as:

$$f(\mathbf{X},\mathbf{p},\mathbf{i}) \;=\; \sum_{p=1}^{M}(\Psi_p(\mathbf{X},\mathbf{i})\cdot\mathbf{x_i^p} \;+\; \mathbf{\Phi_p}(\mathbf{X},\mathbf{i})\cdot\frac{\mathbf{1}}{\mathbf{x_i^p}}) + \mathbf{\Theta}(\mathbf{X},\mathbf{i}) \tag{8}$$

Let the optimal vector be $\mathbf{X}^*$ and $\tilde{x}_i$ be the *local optimum* of x_i with respect to $\mathbf{X}$, we have:

$$\begin{aligned} & \sum_{p=1}^{M}(\Psi_p(\mathbf{X},\mathbf{i})\cdot\tilde{\mathbf{x}}_\mathbf{i}^\mathbf{p} \;+\; \mathbf{\Phi_p}(\mathbf{X},\mathbf{i})\cdot\frac{\mathbf{1}}{\tilde{\mathbf{x}}_\mathbf{i}^\mathbf{p}}) \;+\; \mathbf{\Theta}(\mathbf{X},\mathbf{i}) \\ \leq\; & \sum_{p=1}^{M}(\Psi_p(\mathbf{X},\mathbf{i})\cdot\mathbf{x_i^{*p}} \;+\; \mathbf{\Phi_p}(\mathbf{X},\mathbf{i})\cdot\frac{\mathbf{1}}{\mathbf{x_i^{*p}}}) \;+\; \mathbf{\Theta}(\mathbf{X},\mathbf{i}) \end{aligned} \tag{9}$$

Since x_i^* is the *local optimum* with respect to $\mathbf{X}^*$, we have:

$$\begin{aligned} & \sum_{p=1}^{M}(\Psi_p(\mathbf{X}^*,\mathbf{i})\cdot\mathbf{x_i^{*p}} \;+\; \mathbf{\Phi_p}(\mathbf{X}^*,\mathbf{i})\cdot\frac{\mathbf{1}}{\mathbf{x_i^{*p}}}) \;+\; \mathbf{\Theta}(\mathbf{X}^*,\mathbf{i}) \\ \leq\; & \sum_{p=1}^{M}(\Psi_p(\mathbf{X}^*,\mathbf{i})\cdot\tilde{\mathbf{x}}_\mathbf{i}^\mathbf{p} \;+\; \mathbf{\Phi_p}(\mathbf{X}^*,\mathbf{i})\cdot\frac{\mathbf{1}}{\tilde{\mathbf{x}}_\mathbf{i}^\mathbf{p}}) \;+\; \mathbf{\Theta}(\mathbf{X}^*,\mathbf{i}) \end{aligned} \tag{10}$$

Summing up (9) and (10), we obtain:

$$\sum_{p=1}^{M}\{\{\Psi_p(\mathbf{X},\mathbf{i}) \;-\; \mathbf{\Psi_p}(\mathbf{X}^*,\mathbf{i})\}\cdot\{\tilde{\mathbf{x}}_{\mathbf{i}}^{\mathbf{p}} - \mathbf{x}_{\mathbf{i}}^{*\mathbf{p}}\} \;+\; \{\mathbf{\Phi_p}(\mathbf{X},\mathbf{i}) \;-\; \mathbf{\Phi_p}(\mathbf{X}^*,\mathbf{i})\}\cdot\{\frac{\mathbf{1}}{\tilde{\mathbf{x}}_{\mathbf{i}}^{\mathbf{p}}} - \frac{\mathbf{1}}{\mathbf{x}_{\mathbf{i}}^{*\mathbf{p}}}\}\}$$
$$= \sum_{p=1}^{M}\{\{\Psi_p(\mathbf{X},\mathbf{i}) \;-\; \mathbf{\Psi_p}(\mathbf{X}^*,\mathbf{i}) \;+\; \frac{\mathbf{\Phi_p}(\mathbf{X}^*,\mathbf{i}) \;-\; \mathbf{\Phi_p}(\mathbf{X},\mathbf{i})}{\tilde{\mathbf{x}}_{\mathbf{i}}^{\mathbf{p}}\cdot\mathbf{x}_{\mathbf{i}}^{*\mathbf{p}}}\}\cdot\{\tilde{\mathbf{x}}_{\mathbf{i}}^{\mathbf{p}} - \mathbf{x}_{\mathbf{i}}^{*\mathbf{p}}\}\}$$
$$= \{\sum_{p=1}^{M}\{\{\Psi_p(\mathbf{X},\mathbf{i}) \;-\; \mathbf{\Psi_p}(\mathbf{X}^*,\mathbf{i}) \;+\; \frac{\mathbf{\Phi_p}(\mathbf{X}^*,\mathbf{i}) \;-\; \mathbf{\Phi_p}(\mathbf{X},\mathbf{i})}{\tilde{\mathbf{x}}_{\mathbf{i}}^{\mathbf{p}}\cdot\mathbf{x}_{\mathbf{i}}^{*\mathbf{p}}}\}\cdot\sum_{\mathbf{m}=\mathbf{1}}^{\mathbf{p}}(\tilde{\mathbf{x}}_{\mathbf{i}}^{\mathbf{p}-\mathbf{m}}\cdot\mathbf{x}_{\mathbf{i}}^{*\mathbf{m}-\mathbf{1}})\}\}\cdot\{\tilde{\mathbf{x}}_{\mathbf{i}} - \mathbf{x}_{\mathbf{i}}^*\}$$
$$\leq \; 0$$

If $\mathbf{X}$ dominates $\mathbf{X}^*$, we have $\Psi_p(\mathbf{X},\mathbf{i}) \leq \mathbf{\Psi_p}(\mathbf{X}^*,\mathbf{i})$ and $\Phi_p(\mathbf{X}^*,\mathbf{i}) \leq \mathbf{\Phi_p}(\mathbf{X},\mathbf{i})$, therefore, $\tilde{x}_i - x_i^* \geq 0$ (i.e. the local refinement of $\mathbf{X}$ still dominates $\mathbf{X}^*$). Similarly, if $\mathbf{X}^*$ dominates $\mathbf{X}$, then $\mathbf{X}^*$ dominates the local refinement of $\mathbf{X}$. □

References

[1] M. Berkelaar, P. Buurman and J. Jess, "Computing the Entire Active Area/Power Consumption versus Delay Trade-off Curve for Gate Sizing with a Piecewise Linear Simulator", *Proc. IEEE Int'l. Conf. on Computer-Aided Design*, 1994, pp. 474-480.

[2] M. Borah, R. M. Owens, and M. J. Irwin, "Transistor Sizing for Minimizing Power Consumption of CMOS Circuit under Delay Constraint", *International Symposium Proceedings of Lower Power Design*, 1995, pp. 167-172.

[3] W. Chuang, S. S. Sapatnekar and I. N. Hajj, "Timing and Area Optimization for Standard-Cell VLSI Circuit Design", *IEEE Tran. on Computer-Aided Design*, March 1995, pp. 308-320.

[4] J. Cong and L. He, "Optimal Wiresizing for Interconnects with Multiple Sources", *Proc. IEEE Int'l. Conf. on Computer Design*, Nov. 1995 (full version as *UCLA Computer Science Dept. Tech. Report CSD-00031, 1995*).

[5] J. Cong, and C.-K. Koh, "Simulataneous Driver and Wire Sizing for Performance and Power Optimization", *IEEE Trans. on VLSI*, 2(4), December 1994, pp. 408-423.

[6] J. Cong, and P. H. Madden, "Performance-Driven Routing with Multiple Sources", *Proc. IEEE ISCAS*, 1995.

[7] J. Cong, K. S. Leung, and D. Zhou, "Performance-Driven Interconnect Design Based on Distributed RC Delay Model", *Proc. ACM/IEEE Design Automation Conf.*, 1993, pp. 606-611.

[8] J. Cong and K. S. Leung, "Optimal Wiresizing Under the Distributed Elmore Delay Model", *IEEE Trans. on CAD*, 14(3), March 1995, pp. 321-336 (extended abstract in *Proc. IEEE Int'l. Conf. on Computer-Aided Design*, 1993, pp. 634-639).

[9] J. G. Ecker, "Geometric Programming: Methods, Computations and Appicaltions", *SIAM Review*, Vol. 22, No. 3, July 1980, pp. 338-362.

[10] W. C. Elmore, "The Transient Response of Damped Linear Network with Particular Regard to Wideband Amplifier", *J. Applied Physics*, 19(1948), pp. 55-63.

[11] J. P. Fishburn and A. E. Dunlop, "TILOS: A Psoynomial Programming Approach to Transistor Sizing", *Proc. IEEE Int'l. Conf. on Computer-Aided Design*, 1985, pp. 326-328.

[12] N. Hedenstierna, and K. O. Jeppson, "CMOS Circuit Speed and Buffer Optimization", *IEEE Tran. on CAD*, 1987, pp. 270-281.

[13] T. D. Hodes, et. al. "Dynamically-Wiresized Elmore-Based Routing Constructions," *ISCAS'94*, pp. 463-466 (Vol. I).

[14] S. S. Sapatnekar, "RC Interconnect Optimization Under the Elmore Delay Model", *Proc. ACM/IEEE Design Automation Conf.*, 1994, pp. 387-391.

[15] 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 Tran. on CAD*, November 1993, pp. 1621-1634.

[16] N. Menezes, S. Pullela, and L. T. Pilegi, "Simultaneous Gate and Interconnect Sizing for Circuit-Level delay Optimization", *Proc. ACM/IEEE DAC*, 1995, pp. 690-695.

[17] J. K. Ousterhout, "A Switch-Level Timing Verifier for Digital MOS VLSI, " *IEEE Trans. on CAD*, 4(3) (1983) pp. 336-349.

[18] D. J. Pilling and J. G. Skalnik, "A Circuit Model for Predicting Transient Delays in LSI Logic Systems", *Proc. 6th Asilomar Conf. on Circuits and Systems*, 1972, pp. 424-428.

[19] V. B. Rao and I. Hajj, "Switch-Level Timing Simulation of MOS VLSI Circuits", *Proc. IEEE ISCAS*, 1985, pp. 229-232.

[20] P. M. Vaidya, "A New Algorithm for Minimize Convex Function over Convex Sets", *Proc. IEEE Symposium. on Foundations of Computer Science*, 1989, pp. 338-343.

[21] H. J. M. Veendrick, "Short-Circuit Dissipation of Static CMOS Circuitry and Its Impact on the Design of Buffer Circuits", *IEEE Journal of Solid-State Circuits*, vol. SC-19, no. 4, August 1984, pp. 468-473.

[22] N. H. E. Weste and K. Eshraghian, *Principles of CMOS VLSI Design: a Systems Perspective – 2nd ed*, Addison-Wesley, 1993.