AN EFFICIENT APPROACH TO SIMULTANEOUS TRANSISTOR AND INTERCONNECT SIZING

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. We define a class of optimization problems as CH-posynomial programs and reveal a general dominance property for all CH-posynomial programs (Theorem 1). We show that the STIS problems under a number of transistor delay models are CH-posynomial programs and propose an efficient and near-optimal STIS algorithm based on the dominance property. When used to solve the simultaneous driver/buffer and wire sizing problem for real designs, it reduces the maximum delay by up to 16.1%, and more significantly, reduces the power consumption by a factor of 1.63x, when compared with the original designs. When used to solve the transistor sizing problem, it achieves a smooth area-delay trade-off. Moreover, the algorithm optimizes a clock net of 367 drivers/buffers and $59304\mu m$-long wire in 120 seconds, and a 32bit adder with 1,026 transistors in 66 seconds on a SPARC-5 workstation.

1. INTRODUCTION

The interconnect delay has become the dominating factor in determining the circuit performance in deep submicron designs. We believe that the most effective approach to performance optimization in deep submicron designs 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 a unified methodology and platform for simultaneous logic and interconnect design and optimization, we study the simultaneous transistor and interconnect sizing (*STIS*) problem in this paper.

Most previous works on layout optimization size transistor and interconnect separately, which may lead to suboptimal designs. The transistor sizing problem is to find the optimal width for each transistor under certain objective functions as studied in [14, 18], while the gate sizing problem is to find the optimal width for each gate by assuming all transistor sizes within a gate increase or decrease by a uniform factor [1, 5, 2]. The interconnect sizing problem, also called the wiresizing problem, was first introduced in [10, 11] to determine the optimal width for each wire segment in interconnects and the first polynomial-time optimal algorithm was developed. Later on, alternative wiresizing algorithms were proposed in [17, 23, 25, 6, 3].

Recently, several studies considered simultaneous transistor and interconnect sizing for some special cases. The authors of [8] proposed an efficient algorithm for the simultaneous driver and wire sizing problem. The authors of [16] solved the simultaneous buffer insertion and wire sizing problem by a dynamic programming approach. The authors of [19] studied the simultaneous gate and wire sizing problem and solved it by the sequential quadratic programming technique. Most recently, the authors of [20] solved the simultaneous buffer insertion, wiresizing and tree construction problem. Their gate sizing formulation, however, may lead to suboptimal designs, especially in the full-custom layout. In order to achieve better designs, our STIS formulation is able to optimize the size of *every* transistor and the width of *every* wire segment (under a non-uniform wire segmentation). A very efficient algorithm has been developed based on the general dominance property. It 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.

The remainder of this paper is organized as follows. In Section 2, we present the formulation of the *STIS* problem. In Section 3, we reveal the dominance property for a class of optimization problems named CH-posynomial programs and show that our STIS problems under a number of transistor models are CH-posynomial programs. In Section 4, we propose a polynomial-time algorithm to compute a set of lower and upper bounds of the optimal solution to the STIS problem. In Section 5, experimental results show the algorithm often achieves the identical lower and upper bounds, which lead to the optimal solution. Thus, it is a near-optimal algorithm in practice. A novel contribution of our work is that the algorithm is applicable to any optimization problem if it is a CH-posynomial program. Proofs of theorems are given in a technical report [7].

2. FORMULATIONS

2.1. Delay model for Transistors and Interconnects

We model a transistor by the source-drain *effective resistance* r_d, and the gate, source and drain capacitances c_g, c_s and c_d. Let x be the transistor width, r_d, c_g, c_s and c_d can be written as the following:

$$r_d = r_{d0}/x \quad (1)$$
$$c_g = c_{g0} \cdot x + c_{g1} \quad (2)$$
$$c_s = c_{s0} \cdot x + c_{s1} \quad (3)$$
$$c_d = c_{d0} \cdot x + c_{d1} \quad (4)$$

where c_{g0}, c_{s0} and c_{d0}, as well as c_{g1}, c_{s1} and c_{d1} are constants determined by the technology. In addition, r_{d0} is the

*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.

unit effective resistance as defined in the following: assuming an n-type transistor is driven by a rising input. If the transistor size is x, the capacitance loading C_L, and the 50% delay τ, we define the effective resistance r_d and the unit effective resistance r_{d0} of the transistor as:

$$r_d = \tau / C_L \tag{5}$$
$$r_{d0} = \tau / C_L \cdot x \tag{6}$$

The unit effective resistance of a p-type transistor can be defined similarly under the falling input. Let $\mathbf{X} = \{x_1, \cdots, x_n\}$ denote the sizes for all transistors and wires. We point out that, in general, r_{d0} is a function of $\mathbf{X}$ (i.e., the unit effective resistance of a transistor depends on the sizes of all transistors and wires in the design). We use r_{d0} instead of $r_{d0}(\mathbf{X})$ for the simplicity of presentation.

We model a routing tree as a distributed RC tree, similar to [11, 6]. Each wire segment is divided into a sequence of uni-segments. A uni-segment is treated as a π-type RC circuit and the wire width is assumed uniform within a uni-segment. For simplicity, we assume that all uni-segments have the same wire length and the unit-width uni-segment has wire resistance r_0, wire area capacitance c_0 and wire fringing capacitance c_1. Then, the resistance r and the capacitance c for a uni-segment with width x are

$$r = r_0 / x \tag{7}$$
$$c = c_0 \cdot x + c_1 \tag{8}$$

Our delay computation is similar to that in the switch level timing analysis tool Crystal [21]. The delay will be computed based on a *stage*, which is a DC-current path from a signal source (either the Vdd or the ground) to the gate of a transistor, including both transistors and wires. The delay of a stage $P(N_s, N_t)$ where N_s is its source and N_t is its sink can be written as Eqn. (9) according to the Elmore delay formulation in [24].

$$\begin{aligned} & t(P(N_s, N_t), \mathbf{X}) \\ = & \sum_{i,j} f_0^{st}(i,j) \cdot \frac{x_j}{x_i} + \sum_{i,j} f_1^{st}(i,j) \cdot \frac{1}{x_i} + \\ & \sum_i g^{st}(i) \cdot \frac{1}{x_i} + \sum_i h_0^{st}(i) + \sum_i h_1^{st}(i) \cdot \frac{1}{x_i} \end{aligned} \tag{9}$$

where x_i is the width for either a transistor or a wire, and $f_0^{st}, f_1^{st}, g^{st}, h_0^{st}$ and h_1^{st} are coefficient functions, which can be determined in a similar way as in [11, 6]. We point out that the unit effective resistance r_{d0} for the transistor always appears as a multiplying factor in these coefficient functions. Furthermore, r_{d0} is multiplied by constants which can be determined before the sizing procedure. This fact is helpful to prove the dominance property for STIS problems in Section 3. Similar delay formulations for transistors have been used in both [21] and [14] except that both model the interconnect as a lumped capacitance instead of a distributed RC tree model as used in our formulation. Note that the unit effective resistance r_{d0} of a transistor in both our formulation and [21] is a function of $\mathbf{X}$. However, it was formulated as a constant independent of $\mathbf{X}$ in [14].

2.2. STIS Problem to Minimize Delay for Multiple Paths

In order to minimize delays along multiple critical paths in a circuit simultaneously, we propose to minimize the weighted delay $t(\mathbf{X})$ of all stages in the set of critical paths, which is denoted as $\mathcal{P}$:

$$t(\mathbf{X}) = \sum_{P(N_s,N_t) \in \mathcal{P}} \lambda^{st} \cdot t(P(N_s, N_t), \mathbf{X}) \tag{10}$$

where the penalty weight λ^{st} indicates the criticality of stage $P(N_s, N_t)$, which can be provided either by the designer (after timing simulation) or computed iteratively using the Lagrangian-relaxation method as in [3] to minimize the maximum delay. Note that although the number of critical paths in a circuit may grow exponentially with respect to the number of logic gates, the number of critical stages grows almost linearly. Let

$$F_0(i,j) = \sum_{P(N_s,N_t) \in \mathcal{P}} \lambda^{st} \cdot f_0^{st}(i,j) \tag{11}$$
$$F_1(i,j) = \sum_{P(N_s,N_t) \in \mathcal{P}} \lambda^{st} \cdot f_1^{st}(i,j) \tag{12}$$
$$G(i) = \sum_{P(N_s,N_t) \in \mathcal{P}} \lambda^{st} \cdot g^{st}(i) \tag{13}$$
$$H_1(i) = \sum_{P(N_s,N_t) \in \mathcal{P}} \lambda^{st} \cdot h_1^{st}(i). \tag{14}$$

and eliminate those terms independent of $\mathbf{X}$, Eqn. (10) becomes

$$\begin{aligned} t(\mathbf{X}) = & \sum_{i,j} F_0(i,j) \cdot \frac{x_j}{x_i} + \sum_{i,j} F_1(i,j) \cdot \frac{1}{x_i} + \\ & \sum_i G(i) \cdot \frac{1}{x_i} + \sum_i H_1(i) \cdot \frac{1}{x_i} \end{aligned} \tag{15}$$

With respect to Eqn. (15), we define the following STIS problem to minimize delay through multiple critical paths:

Formulation 1 *Given a circuit and the lower and upper bounds for the width of each transistor and wire, the STIS problem is to determine the width for each transistor and wire (or equivalently, a sizing solution $\mathbf{X}$) such that the weighted delay through multiple critical paths given by Eqn. (15) is minimized.*

In practice, it is often the case that we want to size the transistors and wires without increase in the layout area (using the *free* space in the current layout) or with bounded increase in the layout area. Therefore, there is an upper bound associated with each transistor and wire during the optimization. On the other hand, there is a lower bound associated with each device and wire due to the technology feature sizes and reliability concerns (such as electro-migration). Thus, the lower and upper bounds ($\mathbf{L} \le \mathbf{X} \le \mathbf{U}$) are used to handle these constraints. It will be seen later on that the lower and upper bounds are the starting point for our *STIS* algorithm. It is easy to see that the STIS problem is equivalent to the wiresizing problem when the identical lower and upper bounds are given for the widths of all transistors, or equivalent to the transistor sizing problem when identical lower and upper bounds are given for the wire widths of all wires.

3. THE DOMINANCE PROPERTY

Instead of developing *ad hoc* heuristic methods for the STIS problem, we define a class of optimization problems named *CH-posynomial programs* and reveal the *dominance property* for all CH-posynomial programs. Then, we show that the STIS problems under several delay models are CH-posynomial programs and an efficient STIS algorithm will be developed based on the dominance property.

3.1. Dominance Property for CH-posynomial Programs

First, we define the CH-posynomials as functions of form:

$$\begin{aligned} f(\mathbf{X}) &= \sum_{p,q,i,j} a_{pq,ij} \cdot \frac{x_j^q}{x_i^p} + \sum_{p,i} b_{p,i} \cdot \frac{1}{x_i^p} + \\ &\quad \sum_{p,i} c_{p,i} \cdot x_i^p \qquad (16) \\ \textit{where} &\quad x_i, x_j \in \mathbf{X} = (x_1, \cdots, x_n) \\ &\quad \mathbf{L} \le \mathbf{X} \le \mathbf{U} \\ &\quad p, q \in (1, 2, \cdots, m) \\ &\quad a_{pq,ij},\ b_{p,i}\ \textit{and}\ c_{p,i} \ge 0 \end{aligned}$$

and the coefficients satisfy the following symmetric properties:

1. $a_{pq,ij} > 0$ if and only if $a_{qp,ji} > 0$;
2. $b_{p,i} > 0$ only if $c_{p,i} > 0$ or $a_{pq,ij} > 0$ for any q and j;
3. $c_{p,i} > 0$ only if $b_{p,i} > 0$ or $a_{pq,ij} > 0$ for any q and j;

When these coefficients are constants, this type of functions is a subset of posynomials defined in [13]. In this case, we call them *simple CH-posynomials.* Moreover, we define the following *general CH-posynomials*, which are actually no longer posynomials.

Definition 1 *Eqn. (16) is a general CH-posynomial if coefficients are functions of x_i and x_j satisfying the following conditions: $a_{pq,ij}$ monotonically increases with respect to an increase of x_i and monotonically decreases with respect to an increase of x_j, and $b_{p,i}$ monotonically increases with respect to an increase of x_i and $c_{p,i}$ monotonically decreases with respect to an increase of x_i.*

We call an optimization problem to minimize a simple or general CH-posynomial subject to $\mathbf{L} \le \mathbf{X} \le \mathbf{U}$ as a simple or general *CH-posynomial program*, and introduce the following concepts of *dominance relation* and *local refinement operation.*

Definition 2 *For two vectors $\mathbf{X}$ and $\mathbf{X}'$, we define that $\mathbf{X}$ dominates $\mathbf{X}'$ (denoted as $\mathbf{X} \ge \mathbf{X}'$) if $x_i \ge x_i'$ for all i.*

Definition 3 *The local refinement operation of a solution vector $\mathbf{X}$, with respect to any particular variable x_i and function $f(\mathbf{X})$, is to minimize $f(\mathbf{X})$ subject to changing only x_i while keeping the values of other $x_j (j \ne i)$ unchanged.*

We say that the resulting solution vector is the *local refinement* of $\mathbf{X}$ (with respect to x_i). For simplicity, we also use term *solution* instead of solution vector. Let $\mathbf{X}^*$ be the optimal solution minimizing $f(\mathbf{X})$. We say that the problem of optimizing $f(\mathbf{X})$ satisfies the *dominance property* if $\mathbf{X}$ dominates $\mathbf{X}^*$ implies that a local refinement of $\mathbf{X}$ still dominates $\mathbf{X}^*$, and $\mathbf{X}$ is dominated by $\mathbf{X}^*$ implies that a local refinement of $\mathbf{X}$ is still dominated by $\mathbf{X}^*$

We showed the following important theorem which is the foundation of our STIS algorithm to be presented in Section 4.

Theorem 1 *The dominance property holds for both simple and general CH-posynomial programs.*

The authors of [11] first proposed the dominance property for the single-source wiresizing problem. Our results greatly generalize the concept of the dominance property and reveal that the dominance property holds for a much larger class of optimization problems. One can show that the single-source wiresizing problem [11], the multi-source wiresizing problem [6] and the simultaneous driver and wire sizing problem [8] are instances of the simple CH-posynomial program. The dominance property will lead to an efficient polynomial-time algorithm in Section 4. We will show that STIS problems are CH-posynomial programs in order to use the algorithm.

3.2. Dominance Property for STIS Problems

Recall that the unit effective resistance r_{d0} for the transistor is always a multiplying factor in coefficient functions F_0, F_1, G and H_1 in Eqn. (15). It in fact determines whether Eqn. (15) is a CH-posynomial. There are two types of models for the unit effective resistance r_{d0}. The *step model* assumes that the input to the transistor is always a step so that r_{d0} is a constant independent of the sizing solution $\mathbf{X}$. As a result, all coefficients of Eqn. (15) are positive constants independent of $\mathbf{X}$. We have the following theorem:

Theorem 2 *The STIS problem under the step model is a simple CH-posynomial program with the dominance property.*

The step model has been used in [14] for transistor sizing and in wiresizing works [11, 17, 6] and simultaneous driver and wire sizing work [8] to model the driver. However, the step input is just an ideal assumption. For an inverter, let τ_0 be the delay under the step input and τ the delay under the slope input whose transition time is s. The following relation was given in [15]:

$$\tau = \beta \cdot s + \tau_0 \qquad (17)$$

where β is a constant determined by the technology. It shows that the delay is an increasing function of the input transition time. Recalling the definition of the effective resistance (Eqn. (5)), the larger the input transition time, the larger the effective resistance and the unit effective resistance. Furthermore, we associate the input transition time with the transistor size as the following: since increasing the size of a transistor M_i always increases the gate capacitance of M_i, the output of the previous stage, which is the input to M_i, will become slower because M_i contributes a larger loading capacitance. In turn, the slower input to M_i increases its unit effective resistance. To be more precise, we define the following *DP-slope model*:

Definition 4 *The DP-slope model is a transistor model where the unit effective resistance for the transistor is an increasing function of its size.*

Given the definitions, we also have the following theorem:

Theorem 3 *The STIS problem under a DP-slope model is a general CH-posynomial program with the dominance property.*

It worthwhile to mention that if Eqn. (17) is used to compute a gate delay by assuming that the input transition time is twice the Elmore delay of the previous stage as the transistor sizing formulation in [18], the path delay is a simple CH-posynomial and the STIS problem under this delay model is a simple CH-posynomial program. Nevertheless, Theorem 3 is more general in the sense that it is applicable to the DP-slope model of any form or even without a closed form. A DP-slope model without a closed form will be discussed in Section 4.

4. THE STIS ALGORITHM

4.1. Optimization Using Dominance Property

For a CH-posynomial $f(\mathbf{X})$ where $\mathbf{L} \leq \mathbf{X} \leq \mathbf{U}$, a local refinement based algorithm (LRA algorithm) can be used to compute a set of lower and upper bounds for the optimal solution $\mathbf{X}^*$ to minimize $f(\mathbf{X})$. It is a greedy algorithm based on iterative local refinement operations. Beginning with a solution $\mathbf{X} = \mathbf{L}$, we traverse every x_i in a specific order to perform a local refinement operation on it. Because $\mathbf{X}$ is dominated by $\mathbf{X}^*$, its local refinement is still dominated by $\mathbf{X}^*$. This process is repeated and $\mathbf{X}$ becomes increasingly closer to but always dominated by $\mathbf{X}^*$. This process produces a set of lower bounds of $\mathbf{X}^*$ and is stopped when no improvement is achieved in the last round of traversal. Similarly, a set of upper bounds for $\mathbf{X}^*$ can be obtained by performing local refinement operations beginning with $\mathbf{X} = \mathbf{U}$.

In essence, the LRA algorithm generalizes the greedy wiresizing algorithm GWSA [11]. We define that a lower or upper bound of $\mathbf{X}^*$ is LR-tight if it can not be tightened by any local refinement operation. Then, the LRA algorithm computes the LR-tight lower and upper bounds of $\mathbf{X}^*$ in the polynomial time $O(r \cdot n^2 \cdot l)$[1], where r is the average number of the possible evaluations for all $x_i (i = \{1, \cdots, n\}) \in \mathbf{X}$, and l is the cost of a single local refinement operation, which be discussed in the next subsection. In practice, we usually find that the LR-tight lower and upper bounds meet, which leads to the optimal solution immediately.

4.2. Computation of Coefficient Functions and Local Refinements

In order to initialize the coefficient functions efficiently, all transistors and interconnects are pre-partitioned into DCCs. A DCC is a set of transistors and wires which are connected by DC-current paths containing only transistor channels or wires, and the DC current can not cross the boundary of a DCC. In most cases, a DCC is just a gate G and a routing tree connecting the output of G to the inputs of all gates driven by G. Since these coefficients defined in Eqn. (11)–(14) are computed basically within a DCC, the computation can be finished in $O(\sum_{i=1}^{k} m_i^2)$ time, where k is the total number of DCC's in a circuit and m_i is the total number of transistors and wire uni-segments in the i-th DCC (roughly, $\sum_{i=1}^{k} m_i = n$, the total number of variables in $\mathbf{X}$).

The cost of the local refinement operation is related to the delay model. We use both the step model and a table-based slope model for comparison. Since the STIS problem under the step model is a simple CH-posynomial program with constant coefficients in its objective function, the local refinement operation can be finished in $O(m_i)$ time, where m_i is still the total number of transistors and wire uni-segments in the i-th DCC. We assume a DP-slope model of the most general form; only unit effective resistance r_{d0} for each transistor size is given. Since the STIS problem under this type of slope model is a general CH-posynomial program, the coefficients of Eqn. (15) in fact are functions of the sizing solution $\mathbf{X}$. When there is no closed form to associate $\mathbf{X}$ with the unit effective resistance (in general, no closed form relation between the variables and the coefficients in the general CH-posynomial), the local refinement operation becomes very expensive or even impossible. We propose the following approach: if the starting solution is $\mathbf{X_0}$, we first update the the coefficient functions with respect to $\mathbf{X_0}$ and then compute "local refinement" with respect to these coefficient functions. A novel contribution of this work is that we proved that the dominance property still holds for a general CH-posynomial program with respect to this type of "local refinement". Since the update of coefficient functions before the local refinement operation can be finished in $O(m_i)$ time, the local refinement operation for a general CH-posynomial program still can be finished in $O(m_i)$ time.

Instead of the simple analytical model in Eqn. (17), a lookup table is pre-computed for values of the unit effective resistance r_{d0} for a transistor. In general, r_{d0} of a transistor depends on its size, its input transition time and its capacitance loading so that a three-dimensional table is needed in a straightforward implementation. The three parameters, however, can be combined into one factor called the *slope ratio* to determine r_{d0} solely [22, 21]. Thus, a one-dimensional table is used instead of a three-dimensional table and a table is built for every type of transistors based on SPICE simulations.

4.3. Overview of Near-Optimal STIS Algorithm

The overall STIS algorithm includes three steps: (i) initialization of the coefficient functions, (ii) computation of LR-tight lower and upper bounds of the optimal solution, and (iii) computation of the optimal solution between the LR-tight lower and upper bounds. Note that the coefficient functions will be updated during the sizing procedure because the unit effective resistance r_{d0} under the DP-slope model depends on the sizing solution $\mathbf{X}$.

Furthermore, we proved that there exists an optimal STIS solution such that the optimal wire widths are monotonic within each wire segment, which enable us to generalize the concept of the bundled refinement operation and the bundled-refinement based wiresizing algorithm (OWBR algorithm) [6] to our STIS formulation. It was shown in [6] that the OWBR algorithm runs 100x time faster than the local-refinement based wiresizing algorithm [11].

Finally, if the LR-tight lower and upper bounds are identical for every transistor and wire, the optimal solution is achieved immediately. We observed that the identical LR-tight lower and upper bounds are often achieved in our experiments. In this case, since the coefficients and the LR-tight lower and upper bounds are computed in polynomial times $O(\sum_{i=1}^{k} m_i^2)$ and $O(r \cdot n^2 \cdot \sum_{i=1}^{k} m_i)$, respectively, the optimal STIS solution is achieved in the polynomial-time. Moreover, when the LR-tight lower and upper bounds do not meet, the gap between them is very small, often just of one discrete width in our experiments, and the percent of non-identical lower and upper bounds is also very small, thus enumeration can be carried out in reasonable time. During the enumeration of the width for a transistor or wire between its LR-tight lower and upper bounds, widths of other transistors and wires can be determined by local refinement operations rather than enumeration. Thus, the

[1] The complexity of brute-force enumeration is $O(r^n)$.

optimal solution can be achieved efficiently in practice. It is worthwhile to mention that an LR-tight lower or upper bound has *zero* sensitivity, thus the sensitivity based method can not be used to obtain a solution better than an LR-tight lower or upper bound.

5. EXPERIMENTAL RESULTS

5.1. Simultaneous Driver/Buffer and Wire Sizing

We have implemented the near-optimal STIS algorithm in a SUN SPARC-5 workstation and tested the algorithm on a number of examples for advanced IC technologies. First, we use the STIS algorithm to solve the simultaneous driver and wire sizing (SDWS) problem for multi-source nets.[2] These nets are extracted from an Intel high-performance microprocessor design and were used in [12, 6] for topology construction and wiresizing optimization. We assume that a chain of cascade drivers is used for each source and the first stage is a minimum-size (1x) driver. We compare our STIS method with the CDWS method. The CDWS method uses a constant stage ratio $\alpha = (C_L/C_0)^{1/N}$ where C_0 is the gate capacitance of the 1x driver, and C_L the total loading capacitance when the wires have the minimum width. The stage number N is chosen such that α is around e, the base of natural logarithms, for performance optimization. Then, the CDWS method applies the OWBR algorithm [6] to obtain the optimal wiresizing solution. The STIS method uses the same stage number N and assumes the first stage is also a 1x driver, but the sizes of both wires and transistors in other stages are determined by the STIS algorithm. We use parameters of MCNC $0.5\mu m$ CMOS technology, the same as was used in [6]. We assume the nets are driven by a clock of $20MHz$ and report the HSPICE simulation results in Table 1. Even though the OWBR algorithm achieves the optimal wiresizing solutions under the given driver sizing solutions, the STIS formulation consistently outperforms CDWS: the maximum delay is reduced by up to 17.7%, and more significantly, the total device area, the total wire area and the total power dissipation are reduced by factors of 2.3x, 1.2x and 2.6x, respectively. Although we compute the optimal width for every transistor and every $10\mu m$-long wire, the total runtime of the STIS algorithm is just 7.18 seconds.

Then, we use the STIS algorithm to solve the simultaneous buffer and wire sizing (SBWS) problem. A spread spectrum IF transceiver chip is designed recently [4] and the 1.2 μm 2-layer metal SCMOS technology is used. There are two clock nets, named DCLK and CLK. Each uses a chain of 4 cascade drivers in the clock signal source and uses a chain of 4 cascade buffers in order to drive the register file. All clock source drivers and buffers are tuned manually in the original design. We retain the sizes for both the first stage drivers and the input ports for the register files, and apply the STIS algorithm to optimize the sizes for every $0.6\mu m$-long wire and every transistor in other drivers/buffers. The STIS algorithm optimizes the two designs in 61.24 and 120.79 seconds, respectively. We report HSPICE simulation results in Table 2. When compared with the original designs for the two clock nets, the STIS designs reduce the maximum delay by 16.1% and 14.5%, respectively, and more significantly, reduce the power consumption by factors of 1.63x and 1.55x, respectively. Moreover, we compare our results with those in a recent work [9], where the SBWS problem is studied based on the gate sizing formulation and the step model for transistors. For comparison, the STIS algorithm uses the step model in this experiment. The STIS algorithm achieves more delay reduction (see Table 2), mainly since it uses the transistor sizing formulation. The designs given by the STIS algorithm, however, consumes more power for extra delay reductions, when compared with the designs in [9]. We will show a STIS formulation for area-delay trade-off, which can be extended for power-delay trade-off.

[2]The SDWS formulation in [8] is applicable only to single-source nets.

5.2. Area-Delay Trade-off for Transistor Sizing

In order to achieve the area-delay trade-off, we introduce the following objective function:

$$obj(\mathbf{X}, \gamma) = \gamma \cdot \frac{\sum_{x_i \in \mathbf{X}} x_i}{\sum_{x_i \in \mathbf{L}} x_i} + (1-\gamma) \cdot \frac{t(\mathbf{X})}{t(\mathbf{X} = \mathbf{L})} \qquad (18)$$

It is the scaled weighted sum of area and delays, and γ $(0 \leq \gamma \leq 1)$ can be adjusted for area-delay trade-off. One can easily verify that Eqn. (18) is still a CH-posynomial. Again, we can apply the STIS algorithm efficiently.

We use the STIS algorithm to solve the transistor sizing problem for area-delay trade-off[3]. We sized 8bit, 16bit and 32 bit ripple-adders, respectively, and report the total device areas and the maximum delays in Table 3. We still use parameters of MCNC $0.5\mu m$ CMOS technology, and assume that each primary input to these adders comes from a 2x inverter and each primary output drives a 2x inverter. The STIS algorithm optimizes the 32bit adder with 1,026 transistors in 66 seconds. A smooth delay-area trade-off is observed and the maximum delay is reduced by up to 30.8% with about 2x times area when compared with the minimum-size design. We point out that these adders are implemented in CMOS complex gates and we simply assume that every transistor has the same timing criticality and the same weight penalty. We plan to use the Lagrangian relaxation method [3] to obtain the optimal weight penalty assignment and study the impact of weight penalty assignment. Comparison with previous transistor sizing works [14, 18] is also planned.

6. CONCLUSIONS

The major contribution of this work is to reveal the dominance property for all CH-posynomial programs and show that the STIS problems under a number of delay models are CH-posynomial programs. Based on the dominance property, a polynomial time algorithm is proposed to compute the lower and upper bounds of the optimal solution to CH-posynomial programs. The algorithm often achieves identical lower and upper bounds, which leads to the optimal solution of the STIS problems. It is a near-optimal algorithm in practice and is observed to achieve large delay and power reductions when applied to real designs. Furthermore, the algorithm is applicable to any optimization problem that can be formulated as a CH-posynomial program.

ACKNOWLEDGMENTS

The authors would like to thank Professor C.-J. Richard Shi at University of Iowa, Cheng-Kok Koh and Patrick Madden at UCLA for their helpful discussions.

[3]Note that the algorithm based the dominance property was used only for the optimal wiresizing problem in the past [11, 8, 6].

	# of drivers	length (μm)	max delay (ns)		device area (μm^2)		wire area (μm^2)		average power (mW)		runtime (s)
			CDWS	STIS	CDWS	STIS	CDWS	STIS	CDWS	STIS	
net1	15	3600	1.063	0.876(-17.7%)	563.0	188.6	4320	3240	31.9	2.17	0.08
net2	24	6600	1.064	0.928(-12.3%)	4232.4	734.8	19923	7953	31.6	7.02	0.18
net3	36	10070	1.326	1.225(-7.6%)	8773.2	3222.4	23967	19242	45.8	14.2	0.77
net4	24	10570	1.197	1.120(-6.4%)	6071.4	1331.7	40041	22725	47.9	11.4	0.33
net5	30	31980	1.868	1.808(-3.2%)	8406.8	6456.8	139410	139410	53.2	45.8	5.82
total	129	179800			28046.8(2.3x)	11934.3	227661(1.2x)	192570	210.4(2.6x)	80.59	7.18

Table 1. Simultaneous driver and wire sizing for multi-source nets.

net	# of drivers	wire length (μm)	max delay (ns)			average power (mW)			runtime (s)
			original	SBWS	STIS	original	SBWS	STIS	
dclk	154	41518.2	4.6183	4.1897(-10.2%)	3.8718(-16.1%)	25.9(1.63x)	15.5(0.98x)	15.8	61.34
clk	367	59304.0	5.9030	5.1813(-12.2%)	5.0452(-14.5%)	294.7(1.55x)	127.2(0.67x)	189.8	120.79

Table 2. Comparison of buffer and wire sizing for clock nets.

γ	8bit adder, 258 transistors			16bit adder, 514 transistors			32bit adder, 1026 transistors		
	delay(ns)	area(μm^2)	runtime(s)	delay(ns)	area(μm^2)	runtime(s)	delay(ns)	area(μm^2)	runtime(s)
0.00	2.558(-30.8%)	312.0	9.72	5.611(-25.3%)	623.2	29.42	11.241(-25.8%)	1245.6	66.38
0.01	2.918(-21.1%)	298.2	6.67	6.173(-17.8%)	479.2	16.42	12.324(-18.6%)	904.4	40.42
0.02	3.018(-18.4%)	241.2	5.61	6.549(-12.8%)	416.7	14.55	13.314(-12.1%)	840.9	29.95
0.10	3.597(-2.78%)	156.4	2.87	7.345(-2.3%)	310.4	4.92	14.842(-2.0%)	618.4	15.20
1.00	3.700	152.8	–	7.515	303.3	–	15.145	604.0	–

Table 3. Delay-area trade-off for transistor sizing.

REFERENCES

[1] M. Berkelaar and J. Jess, "Gate Sizing in MOS Digital Circuits with Linear Programming", *Proc. EDAC*, 1990, pp. 217-221.

[2] G. Chen, H. Onodera and K. Tamaru, "An Iterative Gate Sizing Approach with Accurate Delay Evaluation", *Proc. ICCAD*, 1995, pp. 422-427.

[3] C. P. Chen, Y. W. Chang, and D. F. Wong, "Fast Performance-Driven Optimization for Buffered Clock Trees Based on Lagrangian Relaxation," *Proc. DAC*, 1996, pp. 405–408.

[4] C. Chien, P. Yang, E. Cohen, R. Jain, and H. Samueli, "A 12.7Mchip/s All-Digital BPSK Direct Sequence Spread-Spectrum IF Transceiver in 1.2μm CMOS," *Proc. IEEE Int'l Solid-State Circuits Conf.*, 1994, pp. 30–31.

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

[6] J. Cong and L. He, "Optimal Wiresizing for Interconnects with Multiple Sources", to appear in *ACM TODAES* (the extended abstract included in *Proc. ICCAD*, Nov. 1995).

[7] J. Cong and L. He, "Simultaneous Transistor and Interconnect Sizing Based on the General Dominance Property", *UCLA Computer Science, Technical Report 95-00046*, Dec. 1995 (at http://ballade.cs.ucla.edu/~ cong /publications.html).

[8] 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.

[9] J. Cong, and C.-K. Koh, "Simulataneous Buffer and Wire Sizing for Performance and Power Optimization", to appear in *Proc. Int'l. Symp. on Low Power Electronics and Design*, August 1996.

[10] J. Cong, K. S. Leung, and D. Zhou, "Performance-Driven Interconnect Design Based on Distributed RC Delay Model", *Proc. DAC*, 1993, pp. 606-611.

[11] 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.

[12] J. Cong and P. H. Madden, "Performance Driven Routing with Multiple Sources," *Proc. ISCAS*, 1995, pp. 1157–1169.

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

[14] J. P. Fishburn and A. E. Dunlop, "TILOS: A Psoynomial Programming Approach to Transistor Sizing", *Proc. ICCAD*, 1985, pp. 326-328.

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

[16] J. Lillis, C. K. Cheng and T. T. Y. Lin, "Optimal Wire Sizing and Buffer Insertion for Low Power and a Generalized Delay Model", *Proc. ICCAD* Nov. 1995, pp. 138-143.

[17] S. S. Sapatnekar, "RC Interconnect Optimization Under the Elmore Delay Model", *Proc. DAC* 1994, pp. 387-391.

[18] 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.

[19] N. Menezes, R. Baldick, and L. T. Pileggi, "A Sequential Quadratic Programming Approach to Concurrent Gate and Wire Sizing", *Proc. IEEE ICCAD*, 1995, pp. 144-151.

[20] T. Okamoto and J. Cong, "Buffered Steiner Tree Construction with Wire Sizing for Interconnect Layout Optimization," to appear in *Proc. ICCAD*, Nov. 1996.

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

[22] 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.

[23] N. Menezes, S. Pullela, F. Dartu, and L. T. Pillage, "RC Interconnect Synthesis — A Moment Fitting Approach", *Proc. IEEE ICCAD*, 1994, pp. 418-425.

[24] J. Rubinstein, P. Penfield, and M. A. Horowitz, "Signal Delay in RC Tree Networks", *IEEE Trans. on CAD*, 2(3) (1983) pp. 202-211.

[25] T. Xue and E. S. Kuh, "Post Routing Performance Optimization via Multi-Link Insertion and Non-Uniform Wiresizing," *Proc. ICCAD*, 1995, pp. 575–580.