Theory and Algorithm of Local-Refinement Based Optimization with Application to Device and Interconnect Sizing

Jason Cong and Lei He

Both authors are with the computer science department, the university of California at Los Angeles, CA 90095. E-mails: cong@cs.ucla.edu and helei@cs.ucla.edu. This work is partially supported by DARPA/ETO under contract DAAL01-96-K-3600, the NSF Young Investigator Award MIP-9357582 and a grant from Intel Corporation under the NYI matching award program. This research makes use of the software donated from Avant! Corporation, whose generous donation is greatly appreciated.

Abstract

In this paper we formulate three classes of optimization problems: the simple, monotonically-constrained, and bounded CH-programs. We reveal the dominance property under the local refinement (*LR*) operation for the simple CH-program, as well as the general dominance property under the pseudo-LR operation for the monotonically-constrained CH-program and the extended-LR operation for the bounded CH-program. These properties enable a very efficient polynomial-time algorithm, using different types of LR operations to compute tight lower and upper bounds of the exact solution to any CH-program. We show that the algorithm is capable of solving many layout optimization problems in deep submicron IC and/or high-performance MCM/PCB designs. In particular, we apply the algorithm to the simultaneous transistor and interconnect sizing problem, and to the global interconnect sizing and spacing problem considering the coupling capacitance for multiple nets. We use tables pre-computed from SPICE simulations and numerical capacitance extractions to model device delay and interconnect capacitance, so that our device and interconnect models are much more accurate than many used in previous interconnect optimization algorithms. Experiments show that the bound-computation algorithm can efficiently handle such complex models, and obtain solutions close to the global optimum in most cases. We believe that the CH-program formulations and the bound-computation algorithm can also be applied to other optimization problems in the CAD field.

I. Introduction

The interconnect delay has become the dominant factor in determining circuit performance in deep submicron (*DSM*) designs [1]. Many optimization techniques have been proposed to reduce interconnect delay, including interconnect topology optimization, buffer insertion, and device and interconnect sizing (see [2] for a comprehensive survey). 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). This motivates our study of the simultaneous device and interconnect sizing problem in DSM designs.

Several recent studies considered the simultaneous device and interconnect sizing problem. One class of algorithms minimizes the weighted delay. In [3], the simultaneous driver and wire sizing problem was formulated to minimize the weighted delay between the source and a set of sinks for a single net. Procedures of device sizing and wire sizing are alternately carried out, with device sizes computed by closed-form formulas (via Maple) and wire widths computed by algorithms from [4], [5]. In [6], [7], the simultaneous transistor and interconnect sizing problem was studied to minimize the weighted delay for multiple paths (a path contains multiple nets). The local refinement operation, previously used *only* for wire sizing solutions [4], [3], [5], is applied to optimize both devices and interconnects. It leads to a unified and very efficient algorithm. Recently, the simultaneous buffer insertion and wire sizing problem was also addressed [8]. It is assumed that the number of buffers to insert is given for each wire segment, and that the wire widths between any two buffers are monotonic. Therefore, the problem can be solved as a convex quadratic program to find the lengths of wire segments for different wire widths.

The other class of simultaneous device and interconnect sizing algorithms considers the maximum delay. In [9], the simultaneous gate and wire sizing problem was formulated to minimize the area under the maximum-delay constraint for multiple paths. The problem is shown to be a posynomial program, and is transformed into a convex program solved by a sequential quadratic programming technique. In addition, the simultaneous buffer insertion and wire sizing problem was studied to minimize the maximum delay from the source to a set of sinks for a single net [10]. The potential locations for buffer insertion are *a priori* given. Based on a bottom-up dynamic programming approach, buffers are then inserted with optimal sizes, and optimal wire widths determined simultaneously. In general, the algorithms for minimizing the weighted delay are more efficient. By adjusting the weight assignments, a sequence of such minimizations can be used to minimize the maximum delay under the area constraint or to minimize the area under the delay constraint. In particular, a Lagrangian relaxation technique was proposed in [11] to optimally assign the weights for the sequence of weighted-delay minimizations. The simultaneous buffer and wire sizing problem was also solved [11].

However, most of these works assumed over-simplified models for devices and interconnects. For example, a gate of size d and output load c_l is assumed to have a delay $t_d = t_0 + r_d \cdot c_l$, where t_0 and r_d are the *intrinsic delay* and *effective resistance* of the gate, respectively. In addition, $r_d = r_0/d$, where r_0 is the *unit-size effective-resistance* for the gate. Both t_0 and r_0 are assumed to be *constants*. Moreover, the capacitance for a wire of width w and length l is given by $c_a \cdot w \cdot l + c_f \cdot l$, where c_a and c_f are *unit-area capacitance* and *unit-length fringe capacitance* for the wire. Both are again assumed to be *constants*.

These assumptions are no longer realistic for DSM designs. For example, we computed r_0 for an inverter in Table I. We apply HSPICE simulations, and use device parameters for the $0.18\mu m$ technology in Table 5 of the National Technology Roadmap for Semiconductors (NTRS) [12]. When the inverter is driven by a rising input, we first measure two delay values t_1 and t_2 for a pair of output loads c_1 and c_2 under the same size and input switching time. Using the assumption that $t_1 = t_0 + r_d \cdot c_1$ and $t_2 = t_0 + r_d \cdot c_2$, we can obtain $r_d = (t_1 - t_2)/(c_1 - c_2)$, and $t_0 = t_1 - r_d \cdot c_1$. We then compute t_0 values for different combinations of size, input switching time (t_s) and output load (c_l). Because we assume that the intrinsic delay t_0 is a constant in this paper, we derive the "best" t_0 value by least-square-fitting over t_0 values for different combinations of size, t_s and c_l. Finally, we use the "best" t_0 value to compute $r_0 = (t_d - t_0)/c_l \cdot d$, where t_d is the inverter delay, and d the size for the n-transistor in the inverter. We compute r_0 for the n-transistor under different combinations of size, t_s and c_l. Similarly, when the inverter is driven by a falling input, r_0 for the p-transistor can be determined in the same way under different combinations of size, t_s and c_l. As one can see from Table I, r_0 is clearly *not* a constant. Its value may vary by a factor of 2.

We also computed the capacitance for the *basic geometric structure* (see Figure 1), where the *victim* wire is centered between two neighboring wires on the same layer and both top and down grounds (two layers away from the victim). We assume that wires in the basic geometric structure have same widths, then apply a numerical capacitance extraction tool FastCap [13] to solve the structure, using interconnect geometric parameters for the $0.18\mu m$ technology in Table 22 of the NTRS.[1] Figure 2(a) depicts the unit-

[1] The NTRS gives capacitance values only for the minimum width and spacing. Our extracted capacitance values closely

size = 100x						
	n-transistor			p-transistor		
c_l / t_s	0.05ns	0.1ns	0.2ns	0.05ns	0.1ns	0.2ns
0.225pF	12200	13370	19180	17200	19920	24550
0.425pF	8135	9719	12500	17180	17190	18820
0.825pF	8124	8665	10250	17090	17150	17290
1.625pF	8114	8170	8707	16140	17140	17150
3.225pF	7578	8137	8251	14710	16940	17100
size = 400x						
	n-transistor			p-transistor		
c_l / t_s	0.05ns	0.1ns	0.2ns	0.05ns	0.1ns	0.2ns
0.501pF	12200	15550	19150	18200	19970	27030
0.901pF	11560	13360	17440	17340	19590	24560
1.701pF	8463	9688	12470	17070	17420	18790
3.301pF	7725	8812	10420	17030	16780	17440
4.901pF	7554	8480	10010	16090	17020	17060

TABLE I

UNIT-SIZE EFFECTIVE-RESISTANCE FOR N- AND P-TRANSISTOR

length ground capacitance c_g between the victim and grounds, with each curve for c_g under different wire widths but a fixed edge-to-edge spacing (in short, *spacing*). If we assume $c_g = c_a \cdot w \cdot l + c_f \cdot l$, the curve slope should be c_a, and the curve intercept should be c_f. Because none of these curves is linear, and different curves have different intercepts, neither c_a nor c_f is a constant. The total capacitance of the victim is

$$c_{total} = c_g + c_x \cdot l = c_a \cdot w \cdot l + (c_f + c_x) \cdot l$$

where c_x is the *unit-length coupling capacitance* between the victim and the neighboring wires. One can define the *unit-length effective-fringe capacitance* $c_{ef} = c_f + c_x$, and compute $c_{total} = c_a \cdot w \cdot l + c_{ef} \cdot l$. We also obtained c_{ef} for different widths for the victim, under the assumption that the *center-to-edge spacing* (see Figure 1) from the center of the victim to the edges of its neighboring wires is fixed. As shown in Figure 2(b) for two different center-to-edge spacing, c_{ef} is a not a constant either.

We say that a device model is a *simple* model if it assumes that r_0 is a constant, and a capacitance model is a *simple* model if it assumes that both c_a and c_{ef} are constants. Most existing device and interconnect sizing works assume simple device and capacitance models. Little progress has been made for optimization beyond the simple models. The simultaneous buffer insertion and wire sizing algorithm [10] was extended to consider the impact of the input switching time for the device delay. The unit-size effective-resistance, in essence, is assumed to be $r_0 = r'_0 + \delta \cdot t_s$, where r'_0 is the unit-size effective-resistance

match those given in the NTRS (see [1]).

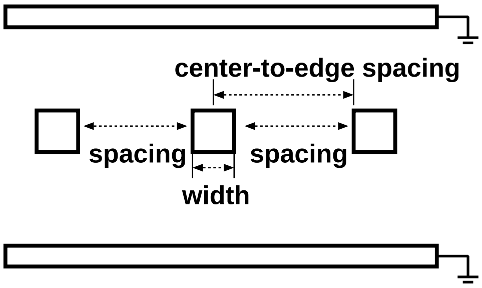

Fig. 1. The basic geometric structure for capacitance extraction.

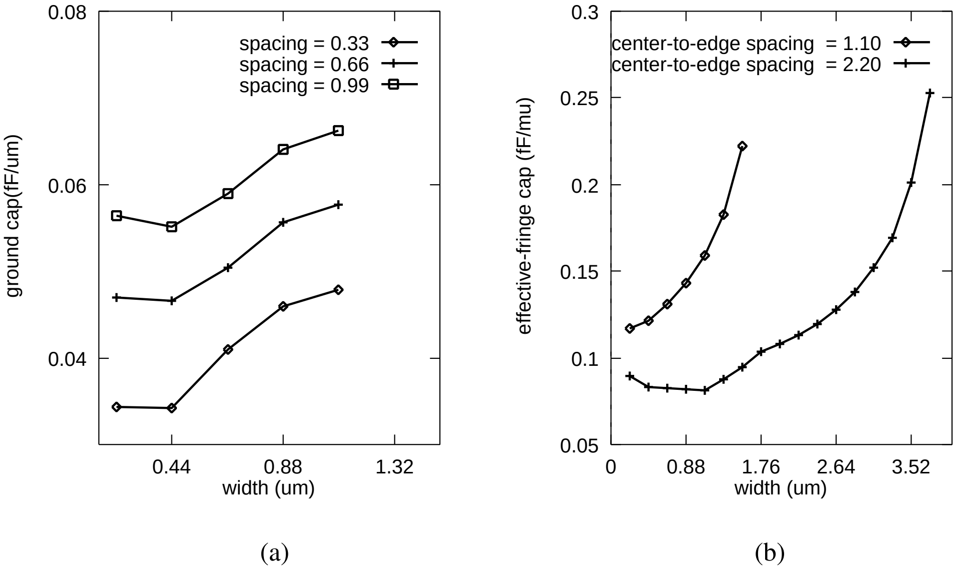

Fig. 2. (a) Ground capacitance and (b) effective-fringe capacitance for the central wire (the victim) in the basic geometric structure shown in Figure 1. Each curve in (a) has the same spacing but different wire widths, and each curve in (b) has the same center-to-edge spacing but different wire widths. The capacitance values are given for the unit-length wire.

under the step input, t_s the input switching time, and δ an empirical constant. The algorithm based on the bottom-up dynamic-programming, however, no longer has a polynomial-time complexity under the extended device model. The posynomial program formulation for the simultaneous gate and wire sizing problem [9] was also extended to accommodate a voltage-ramp gate model, which considers the impacts of the input switching time and output loading under the C_{eff} model [14]. The resulting sizing problem, however, is no longer a posynomial program. It is unknown how far away the solution obtained by solving a posynomial program is from the exact solution under the voltage-ramp model. Two very recent works [15], [16] begin to consider coupling capacitance for multiple nets.[2] Both allow variable c_{ef} but still assume

[2] The formulation in [15] is based on the dominant time constant, which is an approximation to the maximum delay among multiple sinks in a net. Because it is difficult to efficiently minimize the sum of the dominant time constants [15], the Elmore delay model (used in this paper) is more appropriate for path delay minimization.

that r_0 and c_a are constants. Even though all these algorithms still use the simple model for either device delay or interconnect capacitance, their runtime is already high. For example, it took over twenty minutes to optimize a 16-bit bus of 320 wire segments in [16].

We will call the device table, like Table I, *STL-bounded* model, where r_0 is determined by the size, input switching time (t_s) and output load (c_l), and its value is *bounded* (i.e., there exist lower and upper bounds for r_0) for any given ranges of size, t_s and c_l. In addition, a *WS-bounded* capacitance model will be presented in Section IV, where c_a and c_{ef} are determined by the width (w) and spacing (s), and their values are also bounded for any given ranges of w and s. We build tables for the STL-bounded device model via HSPICE simulations, and for the WS-bounded capacitance model via numerical capacitance extractions. These models are more accurate than the simple models, and have been widely used for verification purposes. However, there are virtually no algorithms that allow us to use these models for the device and interconnect sizing problems.

In this paper, we apply the STL-bounded device model and the WS-bounded capacitance model to the simultaneous transistor and interconnect sizing problem (*STIS*), and to the global interconnect sizing and spacing (*GISS*) problem considering the coupling capacitance for multiple nets. In order to efficiently handle the two problems, we formulate three classes of optimization problems: the simple, monotonically-constrained, and bounded CH-programs. We then develop the theory and algorithm based on different local-refinement (LR) operations to optimize three classes of CH-programs. We finally solve the STIS and GISS problems by posing them as CH-programs. Experiments show that we are able to obtain solutions close to the global optimum in the most cases. Based on HSPICE simulations, our algorithm in this paper obtained up to 15.1% and 17% addition delay reductions when compared with STIS results in [7] and GISS results in [16]. Moreover, our algorithm is *extremely* efficient. A speedup of over 100x is achieved compared with the algorithm in [16].

The rest of the paper is organized as follows: we first present the theory and algorithm of LR-based optimization in Section II, then apply the algorithm to the STIS and GISS problems in Sections III and IV, and finally conclude in Section V. Proofs of theorems, together with tables for the device delay and interconnect capacitance used in our experiments, are available from a technical report [17]. Part of preliminary results of this work was presented in two conference papers [7], [18].

II. Theory and Algorithm for CH-Programs

A. Formulations of CH-functions

We first define the CH-function (*Cong-He* function)[3] as a function of a positive vector $\mathbf{X} = \{x_i \mid x_i \geq 0, i = 1, \cdots, n\}$ with the following form:

$$f(\mathbf{X}) \tag{1}$$

[3] CH-function was called CH-posynomial in [7], [18]. As recommended by reviewers, we renamed it to show that it is not, in general, a posynomial.

$$= \sum_{p\geq 0}\sum_{q\geq 0}\sum_{i=1}^{n}\sum_{j=1,j\neq i}^{n}(\frac{a_{p,q,i,j}(\mathbf{X})}{x_i^p})\cdot(b_{p,q,i,j}(\mathbf{X})\cdot x_j^q)$$

where coefficients $a_{p,q,i,j}(\mathbf{X})$ and $b_{p,q,i,j}(\mathbf{X})$, as well as exponents p and q, are positive.

Depending on the coefficient $a_{p,q,i,j}(\mathbf{X})$ and $b_{p,q,i,j}(\mathbf{X})$, we define the following three types of CH-functions:

Definition 1: **(simple CH-function)** Eqn. (1) is a simple CH-function if coefficients $a_{p,q,i,j}$ and $b_{p,q,i,j}$ are constants.

The concept of simple CH-function was first introduced in [6], [7]. It was shown that many previous works on device and interconnect sizing problems, including the single-source and multi-source wire sizing problems [4], [5], continuous wire sizing problem [19], and simultaneous driver and wire sizing problem [3], use simple CH-functions as objective functions.

In some applications however, coefficients $a_{p,q,i,j}(\mathbf{X})$ and $b_{p,q,i,j}(\mathbf{X})$ may vary as functions depending on $\mathbf{X}$. For two vectors $\mathbf{X}$ and $\mathbf{X}'$, we say that $\mathbf{X}$ dominates $\mathbf{X}'$ (denoted by $\mathbf{X}\geq\mathbf{X}'$) if $x_i \geq x_i'$ for $i=1,\cdots,n$. We then define the following *monotonically-constrained CH-function*:

Definition 2: **(monotonically-constrained CH-function)** Eqn. (1) is a monotonically-constrained CH-function, if it satisfies the following monotonic constraints: for any vector $\mathbf{X}'\geq\mathbf{X}$, (i) $\frac{a_{p,q,i,j}(\mathbf{X}')}{x_i'^p} \leq \frac{a_{p,q,i,j}(\mathbf{X})}{x_i^p}$ and $a_{p,q,i,j}(\mathbf{X}')\geq a_{p,q,i,j}(\mathbf{X})$; (ii) $b_{p,q,i,j}(\mathbf{X}')\cdot x_j'^q \geq b_{p,q,i,j}(\mathbf{X})\cdot x_j^q$ and $b_{p,q,i,j}(\mathbf{X}')\leq b_{p,q,i,j}(\mathbf{X})$.

The monotonically-constrained CH-function was defined differently (and called bounded-variation CH-posynomial[4]) in [18], where we say (i) $a_{p,q,i,j}$ is a function depending only on x_i. With respect to an increase of x_i, $\frac{a_{p,q,i,j}(x_i)}{x_i^p}$ monotonically decreases and $a_{p,q,i,j}(x_i)$ monotonically increases; (ii) $b_{p,q,i,j}$ is a function depending only on x_j. With respect to an increase of x_j, $b_{p,q,i,j}(x_j)\cdot x_j^q$ monotonically increases and $b_{p,q,i,j}(x_j)$ monotonically decreases. It is easy to see that Definition 2 subsumes the old definition and covers a wider class of functions, because now each coefficient may vary as a function of all variables in $\mathbf{X}$, instead of a single variable in [18].

We finally remove the monotonic constraints for the CH-function by formulating the following *bounded CH-function*:

Definition 3: **(bounded CH-function)** Eqn. (1) is a bounded CH-function, if its coefficients are bounded: for any p,q,i and j, there exist positive constant $a^L_{p,q,i,j}$, $a^U_{p,q,i,j}$, $b^L_{p,q,i,j}$ and $b^U_{p,q,i,j}$, such that $a^L_{p,q,i,j}\leq a_{p,q,i,j}(\mathbf{X})\leq a^U_{p,q,i,j}$ and $b^L_{p,q,i,j}\leq b_{p,q,i,j}(\mathbf{X})\leq b^U_{p,q,i,j}$.

Clearly, the simple CH-function is a subset of the monotonically-constrained CH-function, which in turn is a subset of the bounded CH-function (see Figure 3). In addition, the simple CH-function is a subset of the posynomial. A posynomial [20] is a function of a positive vector $\mathbf{X}$ having the form $g(\mathbf{X})=\sum_{i=1}^{m}u_i(\mathbf{X})$

[4] According to reviewers' recommendations, we saved the name "bounded" for the type of CH-function defined in Definition 3, which was called the general CH-posynomial in [18].

with

$$u_i(\mathbf{X}) = c_i x_1^{a_{i1}} x_2^{a_{i2}} \cdots x_n^{a_{in}}, \qquad i = 1, 2, \cdots, m \tag{2}$$

where the exponents a_{ij} are real numbers and the coefficients c_i are positive. For example,

$$f(x_1, x_2, x_3) = x_1^{1/2} + 1/x_2 + x_3 \tag{3}$$

is a simple CH-function as well as a posynomial. However,

$$f(x_1, x_2, x_3) = x_1^2 \cdot x_2 \cdot \frac{1}{x_3} \tag{4}$$

is a posynomial but *not* a simple CH-function. On the other hand, the monotonically-constrained and bounded CH-functions may be no longer a posynomial. For example,

$$f(x_1, x_2) = \frac{1}{\ln x_1} \cdot x_1^2 + \frac{x_2}{x_1}, \quad x_1 > 3 \tag{5}$$

is neither a simple CH-function nor a posynomial. However, one can easily verify that it is a monotonically-constrained CH-function by treating $\frac{1}{\ln x_1}$ as the coefficient function for x_1^2.

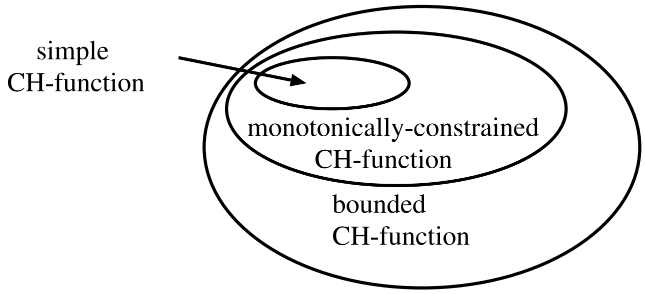

Fig. 3. The simple CH-function is a subset of the monotonically-constrained CH-function, which is in turn a subset of the bounded CH-function.

B. Properties for CH-programs

We define the *CH-program* as an optimization problem to minimize a CH-function subject to $\mathbf{L} \leq \mathbf{X} \leq \mathbf{U}$ (i.e., $l_i \leq x_i \leq u_i$ for $i = 1, \cdots, n$). It may be a simple, monotonically-constrained or bounded CH-program depending on whether its objective function is a simple, monotonically-constrained or bounded CH-function. We will introduce the dominance property for the simple and monotonically-constrained CH-programs, as well as the general dominance property for the monotonically-constrained and bounded CH-programs.

B.1 Dominance property

We first define the following local refinement operation:

Definition 4: **(local refinement operation)** Given a function $f(\mathbf{X})$ and a solution vector (or simply, a solution) $\mathbf{X}'$, the local refinement operation for any particular variable x_i is to minimize $f(\mathbf{X})$ by only varying x_i while keeping all values of other $x_j (j \neq i)$ in $\mathbf{X}'$ fixed.
Such an operation is also called an *LR operation* in short. The resulting solution vector is called the *local refinement* of $\mathbf{X}'$ (with respect to x_i).

Furthermore, we define

$$\begin{aligned} & g(\mathbf{X}) \\ = & \sum_{i=1}^{n} \sum_{j=1, j \neq i}^{n} A_i(x_i) \cdot B_j(x_j) \end{aligned} \tag{6}$$

where $A_i(x_i)$ is a function depending only on x_i, and it increases with respect to an increase of x_i; $B_j(x_j)$ is a function depending only on x_j, and it decreases with respect to an increase of x_j. We have proved the following Lemma 1 in the technical report [17].

Lemma 1: Let $\mathbf{X}^*$ an exact solution to minimize $g(\mathbf{X})$ (Eqn. (6)). For any solution $\mathbf{X}'$ of $f(\mathbf{X})$, if $\mathbf{X}'$ dominates $\mathbf{X}^*$, any local refinement of $\mathbf{X}'$ leads to a solution that still dominates $\mathbf{X}^*$. Similarly, if $\mathbf{X}'$ is dominated by $\mathbf{X}^*$, any local refinement of $\mathbf{X}'$ leads to a solution that is still dominated by $\mathbf{X}^*$.

Based on Lemma 1, one is easy to verify the following dominance property for the simple CH-program:[5]

Theorem 1: **(Dominance Property)** Let $f(\mathbf{X})$ be a simple CH-function, and $\mathbf{X}^*$ an exact solution to minimize $f(\mathbf{X})$. For any solution $\mathbf{X}'$ of $f(\mathbf{X})$, if $\mathbf{X}'$ dominates $\mathbf{X}^*$, any local refinement of $\mathbf{X}'$ leads to a solution that still dominates $\mathbf{X}^*$. Similarly, if $\mathbf{X}'$ is dominated by $\mathbf{X}^*$, any local refinement of $\mathbf{X}'$ leads to a solution that is still dominated by $\mathbf{X}^*$.

The dominance property under the LR operation was first introduced for the single-source wire sizing problem [4], and was extended to the multi-source wire sizing problem [5]. In [7], it was revealed that the dominance property holds for all simple CH-programs. It was also shown that both wire sizing problems [4], [5], the simultaneous driver/buffer and wire sizing problem, and simultaneous transistor and interconnect sizing problem are all simple CH-programs if simple device and capacitance models are used. Therefore, the dominance property holds for these problems and enables an LR-based algorithm, which uses iterative LR operations to compute optimal sizes for both devices and wires.[6]

When coefficients for variable x_i, like the case of simple CH-program, are all constants, the LR operation of x_i is a single-variable posynomial program that can be solved very efficiently.[7] The LR operation for

[5] Nevertheless, Lemma 1 also reveals that the dominance property holds for the monotonically-constrained CH-program when coefficients are functions of single variables, like the bounded-variation CH-program defined in [18]. The dominance property, however, may not hold for the new defined monotonically-constrained CH-program when coefficients are functions of solution vector $\mathbf{X}$.

[6] The SDWS algorithm for simultaneous driver and wire sizing problem in [3] is different from and less efficient than the LR-based algorithm in [7].

[7] According to [20], a posynomial program is the following minimization problem:

$$min \;\; g_0(\mathbf{X}) \qquad subject \; to \; g_k(\mathbf{X}) \leq 1$$

other CH-programs may be less efficient, however. First, it might be no longer a posynomial program. An example is the LR operation of x_1 to minimize Eqn. (5), where a logarithm function is involved. Second, when a coefficient varies depending on a table rather than a closed-form formula, we may have to enumerate all possible values for x_i in order to find out its local optimal value (an example is given in the technical report [17]).

The usage of the LR operation is also limited by the fact that the dominance property under the LR operation generally does *not* hold for a monotonically-constrained or bounded CH-program. To overcome these limitations, we introduce the pseudo-LR and extended-LR operations, then show a general dominance property.

B.2 General dominance property

The pseudo-LR and extended-LR operations (in short, the *PLR* and *ELR* operations) are defined as the following:

Definition 5: **(pseudo-LR operation)** Given a CH-function $f(\mathbf{X})$ and a solution vector $\mathbf{X}'$, the pseudo-LR operation for variable x_i with respect to $\mathbf{X}'$ is an LR operation using *constant* coefficients $a_{p,q,i,j}(\mathbf{X}')$ and $b_{p,q,i,j}(\mathbf{X}')$ when solving the "local-optimal" x_i for any p, q, i and j.
That is, we fix the coefficients under the *current* solution when performing an PLR operation. The PLR and LR operations are same for a simple CH-program, but may produce different results for a monotonically-constrained CH-program.

Definition 6: **(extended-LR operation)** Given a CH-function $f(\mathbf{X})$ and a solution $\mathbf{X}'$, the extended-LR operation for a particular variable x_i in $\mathbf{X}'$ is the LR operation using the following coefficients for any p, q, $j \neq i$, and $k \neq i$:
(i) When $\mathbf{X}' \geq \mathbf{X}^*$, we replace $a_{p,q,ij}(\mathbf{X}')$ and $a_{p,q,k,j}(\mathbf{X}')$ by $a^U_{p,q,i,j}$ and $a^L_{p,q,k,j}$, and replace $b_{p,q,j,i}(\mathbf{X}')$ and $b_{p,q,k,j}(\mathbf{X}')$ by $b^L_{p,q,j,i}$ and $b^U_{p,q,j,k}$.
(ii) When $\mathbf{X}' \leq \mathbf{X}^*$, we replace $a_{p,q,ij}(\mathbf{X}')$ and $a_{p,q,k,j}(\mathbf{X}')$ by $a^L_{p,q,i,j}$ and $a^U_{p,q,k,j}$, and replace $b_{p,q,j,i}(\mathbf{X}')$ and $b_{p,q,k,j}(\mathbf{X}')$ by $b^U_{p,q,j,i}$ and $b^L_{p,q,j,k}$.
We call the solution given by the PLR or ELR operation as the *pseudo-* or *extended-local refinement* of $\mathbf{X}'$, respectively. Note that the lower and upper bounds are *not* unique for coefficient functions. The definition of the ELR operation is applicable to any valid lower and upper bounds.

According to these definitions, even though coefficients are functions of the variable vector $\mathbf{X}$ in the monotonically-constrained or bounded CH-program, coefficients during each PLR or ELR operation are still treated as constants. Therefore, the PLR or ELR operation for a monotonically-constrained or

$$k = 1, 2, \cdots, p \ and \ \mathbf{X} > 0$$

where each g_k $(k = 0, 1, 2, \cdots, p)$ is a posynomial function. In the case of LR operation of x_i for a simple CH-program, the local optimum is also a global optimum no matter whether x_i has continuous or discrete value. More detailed discussion of posynomial programs can be found in Section II-D.

bounded CH-program again becomes a single-variable posynomial program that can be solved very efficiently, exactly as the LR operation for a simple CH-program. We will illustrate the PLR and ELR operations using the following CH-function:

$$\begin{aligned} f(x_1, x_2) &= \frac{a_1(x_1, x_2)}{x_1} \cdot (b_2(x_1, x_2) \cdot x_2^2) \\ &+ \frac{a_2(x_1, x_2)}{x_2} \cdot (b_1(x_1, x_2) \cdot x_1), \end{aligned} \tag{7}$$

The pseudo-local refinement of x_1 with respect to $\mathbf{X}' = \{x_1', x_2'\}$ is

$$\tilde{x}_1^{PLR} = \sqrt{\frac{a_1(x_1', x_2') \cdot (b_2(x_1', x_2') \cdot x_2'^3)}{a_2(x_1', x_2') \cdot b_1(x_1', x_2')}}. \tag{8}$$

If we assume that $a_1(x_1, x_2) \in [a_1^L, a_1^U]$, $a_2(x_1, x_2) \in [a_2^L, a_2^U]$, $b_1(x_1, x_2) \in [b_1^L, b_1^U]$ and $b_2(x_1, x_2) \in [b_2^L, b_2^U]$. When $\{x_1', x_2'\}$ is dominated by exact solution $\{x_1^*, x_2^*\}$, the extended-local refinement of x_1 concerning $\{x_1', x_2'\}$ is

$$\tilde{x}_1^{ELR} = \sqrt{\frac{a_1^L \cdot (b_2^L \cdot x_2'^3)}{a_2^U \cdot b_1^U}}. \tag{9}$$

Even though we assume continuous variables in this example, our definition for the PLR and ELR operations (as well as the LR operation) applies to both continuous and discrete variables. We proved the following theorem concerning the PLR and ELR operations:

Theorem 2: **(General Dominance Property)** Let $\mathbf{X}^*$ an exact solution to minimize a CH-function $f(\mathbf{X})$.

(i) When $f(\mathbf{X})$ is a monotonically-constrained CH-function, for any solution $\mathbf{X}'$ of $f(\mathbf{X})$, if $\mathbf{X}'$ dominates $\mathbf{X}^*$, any pseudo-local refinement of $\mathbf{X}'$ leads to a solution that still dominates $\mathbf{X}^*$; if $\mathbf{X}'$ is dominated by $\mathbf{X}^*$, any pseudo-local refinement of $\mathbf{X}'$ leads to a solution that is still dominated by $\mathbf{X}^*$.

(ii) When $f(\mathbf{X})$ is a bounded CH-function, for any solution $\mathbf{X}'$ of $f(\mathbf{X})$, if $\mathbf{X}'$ dominates $\mathbf{X}^*$, any extended-local refinement of $\mathbf{X}'$ leads to a solution that still dominates $\mathbf{X}^*$; if $\mathbf{X}'$ is dominated by $\mathbf{X}^*$, any extended-local refinement of $\mathbf{X}'$ leads to a solution that is still dominated by $\mathbf{X}^*$.

The proof can be found in the technical report [17]. Because the simple CH-program is a subset of the monotonically-constrained CH-program, and the PLR operation is same as the LR operation in the case of simple CH-program, Theorem 2 also shows that the dominance property holds under the LR operation for the simple CH-program.

C. LR-based algorithm

Again, let $\mathbf{X}^*$ be an exact solution to a CH-program. We say that a solution $\mathbf{X}$ is the lower bound of $\mathbf{X}^*$ if $\mathbf{X}$ is dominated by $\mathbf{X}^*$, and $\mathbf{X}$ is an upper bound of $\mathbf{X}^*$ if $\mathbf{X}$ dominates $\mathbf{X}^*$. Theorems 1 and 2 enable an algorithm based on different types of LR operations to compute a set of lower and upper bounds for $\mathbf{X}^*$.

Bound-Computation Algorithm
1. Initialize lower and upper bounds;
2. If lower and upper bounds do not meet
3. Perform ELR operation on every x_i of the lower bound iteratively;
4. Perform ELR operation on every x_i of the upper bound iteratively;
5. Goto 2 if there is any improvement in 3 and 4;
6. Return ELR-tight lower and upper bounds.

TABLE II

BOUND-COMPUTATION ALGORITHM USING THE ELR OPERATION

Because the bounded CH-program is the most general case, we use the ELR operation to illustrate the bound-computation algorithm (see Table II). Starting with the initial lower and upper bounds (**L** and **U**), the algorithm carries out *interleaved* passes of lower- and upper-bound computations. A *pass* of lower-bound computation will perform an ELR operation on every x_i of a lower bound $\mathbf{X}$ in an *arbitrary* order. Because $\mathbf{X}$ is dominated by $\mathbf{X}^*$, its extended-local refinement becomes closer to $\mathbf{X}^*$ but is still a lower bound. Similarly, a pass of upper bound computation will perform an ELR operation on every x_i of an upper bound $\mathbf{X}$. The iteration of passes is stopped when the lower and upper bounds meet for every x_i, or both bounds are ELR-tight. We say that a lower or upper bound is *ELR-tight* if it can not be improved by any ELR operation.[8] Although the ELR operation may use any valid lower and upper bounds for coefficients according to Definition 6, in general, the closer the lower and upper bounds for coefficients, the smaller the gap between the resulting ELR-tight lower and upper bounds. Because reducing the size of the solution space may narrow the range for coefficients, lower- and upper-bound computations are carried out alternately. The algorithm guarantees that within the resulting ELR-tight lower and upper bounds, there would exist an exact solution to the bounded CH-program.

For a simple or monotonically-constrained CH-program, we may replace the ELR operation in Table II by the LR or PLR operation, respectively. Then, the algorithm computes the LR-tight or PLR-tight lower and upper bounds, where a lower or upper bound of an exact solution is *LR-tight* or *PLR-tight* if it can not be improved by any LR or PLR operation. In essence, the bound-computation algorithm generalizes the greedy wiresizing algorithm GWSA that has been used for computing LR-tight lower and upper bounds for the exact wire sizing solution under fixed c_a and c_{ef} in [4], [5]. When the exact solution

[8] Even though the lower and upper bounds are ELR-tight, there may still be a gap between them. We say that the computation for a variable x_i is *convergent* if its lower and upper bounds are identical. The ELR operation does not guarantee the convergence for all variables. We define the *convergence rate* as the percent of variables that has identical lower and upper bounds. Both *average gap* among all variables and convergence rate will be presented for our experiments in Sections III-D and IV-E.

has the monotone property like those for the single-source and multi-source wire sizing problems [4], [5], the bundled-LR (*BLR*) operation [5] can be used to speed up the LR, PLR or ELR operation. We also use the LR-based algorithm to refer to the bound-computation algorithm, where LR, in general, refers to the LR, PLR, ELR and BLR operations.

The LR-based algorithm has the same worst-case complexity when using different types of LR operations. Let r be the average number of the possible values for variables $x_i (i = \{1, \cdots, n\}) \in \mathbf{X}$ when all variables x_i have discrete values. Because each pass of the lower- and upper-bound computation at least changes the value of one variable to narrow the solution space by at least one unit, the worst-case number of passes is $\Theta(r \cdot n)$. In addition, each pass has at most $2n$ LR operations. Therefore, the bound-computation algorithm needs $\Theta(r \cdot n^2)$ LR operations. We observed in our experiments that the total number of LR operations is much smaller than $\Theta(r \cdot n^2)$ and is *empirically* linear with respect to the number of variables.

D. Comparison with the posynomial program

In order to better appreciate the implications of Theorems 1 and 2, we compare the CH-programs with the posynomial program (defined in Footnote 7). When every variable is of *continuous* value, the posynomial program has the important property that the local optimum is unique, and therefore is also the global optimum. The posynomial program plays an important role in the device and wire sizing works. In [21], the transistor sizing problem was first formulated as a posynomial program and solved by a sensitivity-based method. Later on, the posynomial program formulation was used for transistor sizing [22], wire sizing [23] and simultaneous gate and wire sizing [9], and was solved by being transformed into the convex program.[9] Note that optimality of these solutions depends on the assumption that the local optimum is unique. The assumption holds for the continuous sizing formulation and simple models for the interconnect capacitance and device delay, but may be not true for the discrete sizing formulation and more general models for the interconnect capacitance and device delay.

Our LR-based algorithm is similar to the coordinate descent approach [24] for the posynomial program. The approach iteratively optimizes the value for each variable (i.e., coordinate) while keeping the values for the rest of the variables fixed.[10] Because the local optimum is unique for the posynomial program regarding continuous variables, one may even start with an *arbitrary* solution (see [25]) rather than a lower or upper bound used in the LR-based algorithm. However, when the variables $x_1, x_2, \cdots, x_n$ are of discrete values

[9] Same as the method in [9] that we reviewed in Section 1, methods in [22], [23] minimize the maximum delay.

[10] An alternative method, called the steepest descent approach or the gradient method [24], minimizes the objective function along the direction of the steepest gradient, and may simultaneously change all coordinates. In general, it is $n - 1$ times faster than the coordinate descent approach, where n is again the number of variables [24]. However, because of the special nature of the sizing problems, the LR-based optimization (the coordinate descent approach) turns out to be very efficient in experiments. In fact, it was recently shown that when using the simple device and capacitance models, the LR-based algorithm can be finished in a linear time for the continuous wire sizing problem [25].

for the simple CH-program, or when the coefficients are not constants as in the monotonically-constrained or bounded CH-program (for both continuous or discrete variables), there may be more than one local optimum.[11] Then, the global optimum can not be achieved by the coordinate descent approach starting from an arbitrary solution. However, the LR-based algorithm, which respectively uses the LR, PLR or ELR operations for a simple, monotonically-constrained or bounded CH-program, can still be used to compute lower and upper bounds for the exact (i.e., *globally* optimal) solution. We will apply the ELR operation to the simultaneous transistor and interconnect sizing problem under the STL-bounded device model, and apply the PLR and ELR operations to the global interconnect sizing and spacing (*GISS*) problem considering the coupling capacitance for multiple nets. Both problems are no longer the simple CH-program, and may have multiple local optimum solutions.

III. STIS Problem under STL-Bounded Device Model

A. Problem formulation

Our formulation is similar to that in [7]. The delay is computed based on *a stage*. It is defined as a DC-connected path from a power supply (either the Vdd or the ground) to the gate node of a transistor, containing both transistors and wires. The delay of a stage $P(N_s, N_t)$ with N_s being the source and N_t being the sink can be written as Eqn. (10) under the Elmore delay model.

$$\begin{aligned} & t(P(N_s, N_t), \mathbf{X}) \\ = & \sum_{i,j} f(i,j) \cdot \frac{r_0(i)}{x_i} \cdot c_a(j) \cdot x_j + \sum_{i,j} f(i,j) \cdot \frac{r_0(i)}{x_i} \cdot c_{ef}(j) \\ + & \sum_i g(i) \cdot \frac{r_0(i)}{x_i} + \sum_i r_0(i) \cdot h(i) + \sum_i h(i) \cdot \frac{r_0(i)}{x_i} \end{aligned} \tag{10}$$

where x_i is the width for a transistor M_i or a wire E_i, $r_0(i)$ is its unit-size effective-resistance, and $c_a(i)$ and $c_{ef}(i)$ are its unit-area capacitance and unit-length effective-fringe capacitance. Coefficients $f(i,j), g(i)$ and $h(i)$ are determined by the transistor netlist and routing topology.

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

$$t(\mathbf{X}) = \sum_{P(N_s,N_t)\in\mathcal{P}} \lambda_{st} \cdot t(P(N_s, N_t), \mathbf{X}) \tag{11}$$

where the weight λ_{st} indicates the criticality of stage $P(N_s, N_t)$. After we eliminate those terms independent of $\mathbf{X}$, Eqn. (11) can be re-written as

$$\begin{aligned} & t(\mathbf{X}) \\ = & \sum_{i,j} F(i,j) \cdot \frac{r_0(i)}{x_i} \cdot c_a(j) \cdot x_j \end{aligned}$$

[11] The simple CH-program using continuous variables belongs to the posynomial program, and therefore has a unique local optimum.

$$+ \sum_{i,j} F(i,j) \cdot \frac{r_0(i)}{x_i} \cdot c_{ef}(j)$$
$$+ \sum_{i} G(i) \cdot \frac{r_0(i)}{x_i} + \sum_{i} H(i) \cdot \frac{r_0(i)}{x_i} \quad (12)$$

where $F(i,j), G(i)$ and $H(i)$ are weighted functions of $f(i,j), g(i)$ and $h(i)$, respectively.

We formulate the following simultaneous transistor and interconnect sizing (STIS) problem:

Formulation 1: Given the lower and upper bounds ($\mathbf{L}$ and $\mathbf{U}$) for the width of each transistor and wire, the STIS problem is to determine a width for each transistor and wire (or equivalently, a sizing solution $\mathbf{X}$, $\mathbf{L} \leq \mathbf{X} \leq \mathbf{U}$) such that the weighted delay through multiple critical paths given by Eqn. (12) is minimized.

Note that a sequence of weighted-delay minimization can be used to minimize the maximum delay by adjusting the weight assignment based on the Lagrangian-relaxation method as in [11]. Therefore, we focus on how to minimize weighted delay in this paper. In addition, we assume that the possible width is from a *discrete* width set determined by the technology. The discrete sizing problem is more difficult than the continuous sizing problem, but is more convenient for placement and routing tools and fabrication.

B. Bound computation for the STIS problem

Under the simple models, r_0, c_a and c_{ef} are constants for each wire/transistor, and Eqn. (12) is a simple CH-function. In this case, the STIS problem is a simple CH-program solved in [7]. Because the simple models are no longer valid for DSM designs, we study the STIS problem under the STL-bounded device model that is more suitable for DSM designs. For simplicity of presentation, we assume here that c_a and c_{ef} are constants for each wire segment, but will remove the assumption in Section IV.

In the STL-bounded model, r_0 is pre-computed and stored in tables (e.g., see Table I) indexed by the size, input switching time (t_s), and output load (c_l). It could be very accurate depending on the table size.[12] Because the value for r_0 is bounded, it is easy to verify the following Theorem 3:

Theorem 3: The STIS problem under the STL-bounded device model is a general CH-program.

Note that the STL-bounded model might *not* be monotonic with respect to the sizing solution $\mathbf{X}$. Therefore, the STIS problem is unlikely a monotonically-constrained CH-program, and the LR and PLR operations are not applicable. It can be justified by the following observations: r_0 in our model is a monotonic function of t_s, whereas t_s is *not* monotonic with respect to $\mathbf{X}$, because the optimal wire sizing solution (see [4], [23], [5]) to minimize t_s often has neither minimum nor maximum wire width.

Therefore, the ELR operation is needed in the LR-based algorithm (Table II) to compute lower and

[12] In our experiments, r_0 table for a type of gate (e.g., an inverter) considers the combinations of five different device sizes (from 1x to 800x of the minimum size), three different input switching times, and five different load capacitances. Therefore, the total table size is $5 \times 3 \times 5 \times m = 75m$, where m is the number of gate types. Satisfactory optimization results are obtained according to experiments in Section III-D. For simplicity, we assume that c_l is the lumped capacitance in this paper. Extension to the effective capacitance model [14] is ongoing work and will be discussed briefly in Section V.

upper bounds for an exact solution to the STIS problem. We assume that $r_0(i) \in [r_0^L(i), r_0^U(i)]$ and $r_0(j) \in [r_0^L(j), r_0^U(j)]$. In an ELR operation on a transistor M_i for the lower-bound computation, we use $r_0^L(i)$ instead of $r_0(i)$, and $r_0^U(j)$ instead of $r_0(j)$ for M_j, where M_j is an upstream transistor in the same net for M_i. Symmetrically, in an ELR operation on M_i for the upper-bound computation, we use $r_0^U(i)$ instead of $r_0(i)$ for M_i, and $r_0^L(j)$ instead of $r_0(j)$ for an upstream transistor M_j.

We determine $r_0^L(i)$ as follows: Let $\mathbf{X}^L$ and $\mathbf{X}^U$ be lower and upper bounds of the exact solution $\mathbf{X}^*$. We assume that transistor M_i has size $x_i \in [x_i^L, x_i^U]$, input switching time $t_s(i) \in [t_s^L(i), t_s^U(i)]$, and capacitance load $c_l(i) \in [c_l^L(i), c_l^U(i)]$. We often observe in our experiments that $r_0(i)$ increases with respect to an increase of x_i or $t_s(i)$, but decreases with respect to an increase of $c_l(i)$. Therefore, $r_0^L(i)$ for M_i can be obtained by table lookup using x_i^L, $t_s^L(i)$ and c_l^U(i). Symmetrically, $r_0^U(i)$ is determined using x_i^U, $t_s^U(i)$ and $c_l^L(i)$. In addition, contributions of transistors or wires to $c_l^U(i)$ are computed using sizes in $\mathbf{X}^U$, and contributions to $c_l^L(i)$ computed using sizes in $\mathbf{X}^L$. After the ELR operation on M_i, for every stage $P(N_i, N_j)$ (N_i is the source, N_j is the sink) driven by M_i, we will update the lower and upper bounds for the switching time $t_s(j)$ at sink N_j, because $t_s(j)$ is the input switching time for the transistor M_j with gate connected to node N_j. The lower or upper bound of $t_s(j)$ is assumed to be the lower or upper bound of the delay through $P(N_i, N_j)$, respectively. As $\mathbf{X}^L$ and $\mathbf{X}^U$ move closer during the ELR-based optimization procedure, the range of r_0 is also narrowed. In general, the closer the values for r_0^U and r_0^L, the smaller the gap between the lower and upper bounds given by the ELR operations.

Because the unit-size resistance $r_0(i)$ is a constant for each wire segment E_i, we can simply use the LR operation for E_i. Furthermore, in order to achieve better wire sizing solutions, we can divide a wire segment into a sequence of uni-segments, then find a wire width for each uni-segment [5]. We assume that each segment always stays in the same layer, has the fixed r_0, c_a and c_{ef}, as well as same allowable wire widths.[13] With these assumptions, we have proved the following *local monotone property*:

Theorem 4: (**local monotone property**) There exists an optimal STIS solution where the wire widths for uni-segments are monotonic within each wire segment.

The proof is available from the technical report [17]. This theorem enables us to use the BLR operation [5] instead of the LR operation for each wire segment E_i. The BLR operation is shown to be 100x faster than the LR operation for the wiresizing problem [5].

C. Overall algorithm for the STIS problem

Let $\mathbf{L}'$ and $\mathbf{U}'$ be the ELR-tight lower and upper bounds given by the above bound-computation procedure. If $\mathbf{L}'$ and $\mathbf{U}'$ are identical, we obtain the exact solution to the STIS problem under the STL-bounded model. Otherwise, we traverse all wire segments and transistors by iterative PLR operations until there is no improvement in the last round of traversal. Note that the PLR operation is bounded by

[13] Different segments may have different r_0, c_a and c_{ef} if they are in different layers, or have different spacings to neighboring wires.

$\mathbf{L}'$ and $\mathbf{U}'$, and it uses r_0 obtained from the device table. Even though the PLR operation may lead to further improvement over $\mathbf{L}'$ and $\mathbf{U}'$, in general it does *not* lead to a lower or upper bound of the exact solution.[14]

Our experiments in Section III-D.2 show that the ELR-tight lower and upper bounds ($\mathbf{L}'$ and $\mathbf{U}'$) are often close to each other in most cases. Therefore, we can simply treat $\mathbf{L}'$ as the final solution (for smaller area and often lower power-dissipation). Note that the STIS problem to minimize a weighted-sum of delay and area is shown to be a CH-program in [7], with a smooth trade-off obtained between delay and area. A similar approach can be used to better minimize the capacitive power by minimizing the weighted-sum of delay and capacitive power.

D. *Experimental results*

For all experiments in this paper, we computed the delays via HSPICE using the distribute RC model and the level-3 MOSFET model that is also used in HSPICE simulations for device-table generation. The use of HSPICE simulation results not only shows the quality of our sizing solutions, but also verifies the validity of our interconnect and device modeling, and the correctness of our problem formulations.

D.1 Comparison between manual optimization and STIS algorithm

To illustrate the effectiveness of the STIS algorithm, we first compare the sizing solution obtained by our algorithm and the manual optimization applied to a spread spectrum IF transceiver chip in [26]. The design is under the 1.2 μm two-layer metal SCMOS technology. There are two clock nets, *dclk* and *clk*; each uses a chain of four cascade drivers in the clock signal source and chains of four cascade buffers in order to drive long interconnects and register files. The maximum delays of the two nets need to be minimized to reduce the clock skew. Therefore, source drivers and buffers are tuned manually via iterative procedures of layout, extraction and HSPICE simulation. We retain the manual sizing solutions for the first stage drivers at the source and for the drivers of the register files, then apply the STIS algorithm to optimize the sizes for every $10\mu m$-long wire and the rest of the drivers and buffers. We use two formulations under the simple device model, one is simultaneous transistor and wire sizing formulation (*stis/simple*) where optimal sizes are found for p- and n- transistors in each driver/buffer, and the other one is simultaneous gate and wire sizing formulation (*sgws/simple*) where an optimal size is found for each driver/buffer. We also assume that the allowable wire widths are $\{w, 2w, 3w, 4w, 5w\}$ with $w = 1.2\mu m$ being the minimum wire width in the $1.2\mu m$ technology, and the allowable transistor sizes are multiples of $0.6\mu m$ between $1.2\mu m$ and $500\mu m$. The constant value for r_0 in the simple model is determined under the typical input switching time, device size and output load. The fixed ratio between p- and n- transistors in the *sgws/simple* formulation is tuned to make sure that the inverter will have same pull-up and pull-down

[14] In our experiments, we tried to use PLR operations starting from either the minimum or maximum sizing solution. The resulting solutions are often outside the range defined by $\mathbf{L}'$ and $\mathbf{U}'$, and are worse than $\mathbf{L}'$.

resistance values.

net	# of drivers/buffers	wire length (μm)	max delay (ns)			average power (mW)		
			manual	sgws/simple	stis/simple	manual	sgws/simple	stis/simple
dclk	154	41518.2	4.6324	4.3447(-6.2%)	3.9635(-14.4%)	60.85	46.09(-24.3%)	46.29(-24.2%)
clk	367	59304.0	6.2016	6.1578(-0.7%)	5.9035(-4.8%)	499.7	286.8(-42.6%)	285.6(-42.8%)

TABLE III

Comparison between manual optimization and STIS algorithms

Because the simple device model is applied, we use the LR operation to compute the LR-tight lower and upper bounds for devices. Experiments show that the identical LR-tight lower and upper bounds are achieved for almost all devices and wire segments, therefore we use the LR-tight lower bounds as the final sizing solution. We report HSPICE simulation results in Table III. When compared with the manual optimization, *sgws/simple* and *stis/simple* formulations reduce the maximum delay by up to 6.2% and 14.4%, respectively. More significantly, both reduce the power consumption by 42.6% and 42.8%. Because we use the same simple model for two formulations in this experiment, the extra delay reduction (8.2%) of the *stis/simple* formulation comes from the flexibility of the transistor sizing formulation.

D.2 Comparison between simple and STL-bounded models

We then apply our STIS algorithm under different device models. We use the 0.18 μm technology given in the NTRS [12] in order to study the impact of the DSM technologies. The wire sheet-resistance $R_{\square} = 0.0638\Omega$. We generate device and capacitance tables via HSPICE simulations and numerical extractions, respectively, and use c_a and c_{ef} values where the wire is $1.10\mu m$ wide and neighboring wires are $1.65\mu m$ away. We size two global nets, one is a $2cm$ line with five buffers optimally inserted for delay minimization. The other is the above *dclk* net. In addition to different device models (simple model versus STL-bounded model), we also use different sizing formulations (*sgws* versus *stis*). There are four combinations, including *sgws/simple* and *stis/simple* using the LR operation for devices, and *sgws/bounded* and *stis/bounded* using the ELR operation for devices. For simplicity, we assume that the fixed ratio between p- and n- transistors for the gate sizing formulation is 1.0. For both nets, we find the optimal wire width for each 10 μm-long wire, and assume that allowable transistor sizes are multiples of $0.18\mu m$ between $0.18\mu m$ and $144\mu m$, and that allowable wire widths are multiples of $0.56\mu m$ between $0.56\mu m$ and 5.6 μm.

Table IV summarizes experimental comparisons between different formulations. We computed convergence rate under different formulations. For the simple model, the computation for a transistor or wire is *convergent* if its LR-tight lower and upper bounds are identical. For the STL-bounded model, the computation for a transistor or wire is *convergent* if its ELR-tight lower and upper bounds are identical.

net	sgws/ simple	sgws/ bounded	stis/ simple	stis/ bounded	sgws/ simple	sgws/ bounded	stis/ simple	stis/ bounded
	convergence rate for transistors				convergence rate for wire			
dclk	85.8%	83.2%	87.7%	86.7%	99.4%	95.9%	97.1%	95.2%
line	60.0%	100%	70.0%	60.0%)	98.4%	70.9%	88.4%	72.9%
	average width / average gap (for transistors, μm)				average width / average gap (for wires, μm)			
dclk	5.39/0.07	13.0/1.91	17.2/1.53	21.6/2.36	2.50/0.003	2.78/0.025	2.69/0.017	2.82/0.030
line	108/0.108	112/0.0	126/0.97	125/1.98	4.98/0.004	4.99/0.106	5.05/0.032	5.11/0.091
	maximum delay (ns)				runtime (s)			
dclk	1.159(0%)	1.007(-6.4%)	1.132(0%)	0.961(-15%)	1.18	2.32	0.88	3.17
line	0.821(0%)	0.818(-0.4%)	0.751(0%)	0.694(-7.6%)	0.72	0.58	0.55	1.22

TABLE IV

Comparisons between different device and wire sizing formulations

The convergence is not significantly different. For example, computations for about 85% transistor are convergent in *dclk* net under all four formulations. We also computed the average width and the average gap between lower and upper bounds for all wire segments and transistors, respectively. The ELR operation does give larger gap than the LR operation. However, the difference is small. Overall, the average gap is only 1% of the average width, except that net *dclk* has a large gap, nearly 10% of the transistor size.

We simply use the ELR-tight lower bound as the final solution under the STL-bounded model, and the LR-tight lower bound as the final solution under the simple model, because lower and upper bounds given by bound computations are very close to each other. Table IV also give the maximum delay via HSPICE simulation. The solutions under the STL-bounded model are consistently better than those under the simple device model. When compared with the *sgws/simple* formulation, the *sgws/bounded* formulation further reduce the maximum delay by up to 6.4%. When compared with the *stis/simple* formulation, the *stis/bounded* formulations further reduce the maximum delay by up to 15%. Note that both *sgws/simple* and *stis/simple* formulations already give very good sizing solutions as shown in the experiment of Section III-D.1. Although ELR operations under the STL-bounded model are more complex, the runtime is still impressively small. It used just 3.17 seconds to optimize *dclk* net of 154 buffers and 41518.2μm wires, when the transistor sizing formulation is used and wire segments are 10μm long. Therefore, our STIS algorithm is extremely efficient.

IV. GISS problem considering coupling capacitance

The unit-area capacitance c_a and unit-length effective-fringe capacitance c_{ef} are assumed to be constants for each wire segment in the STIS problem in Section III. We shall proceed to remove this assumption using the more general WS-bounded capacitance model in this section. For simplicity of presentation, we assume that the device sizes are fixed, and study the global interconnect sizing and spacing (GISS) problem for multiple nets with consideration of the coupling capacitance. However, our algorithm and implementation are able to use the STL-bounded device model and the WS-bounded capacitance model (with consideration of the coupling capacitance) at the same time.

A. Problem formulation

Our GISS formulation was first presented in [16]. We assume that an initial layout is *a priori* given and defines the *initial central-line* for each wire segment. The *initial pitch-spacing*, i.e., the distance between the initial central-lines, remains *unchanged* during the sizing procedure. We consider two wire sizing formulations. One is the *symmetric* wire sizing formulation, where wires are always symmetric with respect to initial central-lines as illustrated in Figure 4(a). In contrast, in the *asymmetric* wire sizing formulation shown in Figure 4(b), wires of same widths are asymmetric with respect to initial central-lines, and have smaller capacitance and less delay. Because neighboring wires are, in general, asymmetrically away from interested nets, the asymmetric wire sizing formulation is capable of further reducing the interconnect delay.

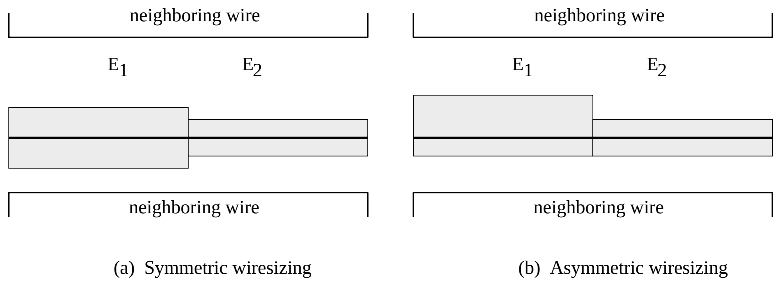

Fig. 4. (a) symmetric wire sizing, and (b) asymmetric wire sizing. The asymmetric wire sizing has smaller capacitance and less delay.

Given the asymmetric formulation, in general, the wire sizing solution for wire segment E_i needs to be represented by a pair of widths $(x_i^{\uparrow}, x_i^{\downarrow})$, where $x_i^{\uparrow}$ is the width of the piece of wire above (or left to) the initial central-line when E_i is a horizontal (or vertical) segment, and $x_i^{\downarrow}$ the width of the piece of wire on the other side of the initial central-line. Similarly, we denote the spacing above (or left to) E_i as $s_i^{\uparrow}$, and spacing on the other side as $s_i^{\downarrow}$. In order to maintain the connectivity, we say that a wire width x_i is *valid* if $x_i^{\uparrow}$ and $x_i^{\downarrow}$ are at least $W_{min}/2$, where W_{min} is the minimum wire width set by the manufacture technology.

With consideration of both symmetric and asymmetric wire sizing formulations, we define the following GISS problem:

Formulation 2: Given multiple nets with initial central-line for each wire segment E_i, the GISS problem is to determine a valid wire width $(x_i^{\uparrow}, x_i^{\downarrow})$ for each E_i with respect to its initial central-line, such that the weighted delay given by Eqn. (12) is minimized for multiple critical paths over these nets.

Note that, as shown in Figure 2, both c_a and c_{ef} are functions of wire widths and spacings. In the following, we shall first consider the symmetric wire sizing formulation, then extend our algorithms to the asymmetric wire sizing formulation.

B. Bound computation for the symmetric GISS problem

Our WS-bounded capacitance model is a table-based model simplified from the 2.5D capacitance model in [27]. In this model, we first use the numerical capacitance extraction to solve the *basic geometric structure* with equal widths and spacings (see Figure 1). We consider different width and spacing combinations, and store $c_a(x,s)$ and $c_{ef}(x,s)$ in two-dimensional tables indexed by widths (x) and spacings (s). Then, for a wire segment E_i with width x_i and spacings $s_i^{\uparrow}$ and $s_i^{\downarrow}$ to its two nearest neighboring wires E_j and E_k, we compute $c_a(i)$ as

$$c_a(i) \quad = \quad \frac{c_a(x_i, s_i^{\uparrow}) + c_a(x_i, s_i^{\downarrow})}{2}, \tag{13}$$

and compute $c_{ef}(i)$ as

$$c_{ef}(i) \quad = \quad c_{ef}^{\uparrow}(i) + c_{ef}^{\downarrow}(i), \tag{14}$$

where $c_{ef}^{\uparrow}(i)$ is the unit-length effective-fringe capacitance between E_i and E_j, and $c_{ef}^{\downarrow}(i)$ the unit-length effective-fringe capacitance between E_i and E_k. They are given as

$$c_{ef}^{\uparrow}(i) \quad = \quad \frac{c_{ef}(x_i, s_i^{\uparrow}) + c_{ef}(x_j, s_i^{\uparrow})}{2}, \tag{15}$$

$$c_{ef}^{\downarrow}(i) \quad = \quad \frac{c_{ef}(x_i, s_i^{\downarrow}) + c_{ef}(x_k, s_i^{\downarrow})}{2}, \tag{16}$$

$$\tag{17}$$

where x_j and x_k are widths for E_j and E_k, respectively.

Because our GISS formulation assumes that the initial central-lines are fixed, $s_i^{\uparrow}$ can be determined by $x_i^{\uparrow}$ and $x_j^{\downarrow}$, and $s_i^{\downarrow}$ by $x_i^{\downarrow}$ and $x_k^{\uparrow}$. Therefore, $c_a(i)$ and $c_{ef}(i)$ are functions of x_i, x_j and x_k. Because their values are obviously bounded, we have the following Theorem 5:

Theorem 5: The GISS problem under the WS-bounded capacitance model is a bounded CH-program.

Note that the GISS problem is easier than the STIS problem in the sense that coefficient c_a or c_{ef} in GISS is a function of just four variables, whereas coefficient r_0 in STIS may depend on all variables.

Based on this theorem, we may use the ELR operation to compute the lower and upper bounds for x_i^*, the optimal width for a wire segment E_i. If we assume that $c_a \in [c_a^L, c_a^U]$ and E_i has two neighboring wires E_j and E_k, in an ELR operation during the lower-bound computation for E_i, we use $c_a^U(i), c_a^U(j)$ and $c_a^U(k)$ instead of $c_a(i), c_a(j)$ and $c_a(k)$ for E_i, E_j and E_k, and use $c_a^L(n)$ instead of $c_a(n)$ for E_n that is a downstream segment of E_i, E_j, or E_k. Similarly, during the upper-bound computation for E_i, we use $c_a^L(i), c_a^L(j)$ and $c_a^L(k)$ for E_i, E_j and E_k, and $c_a^U(n)$ for downstream segment E_n. Furthermore, we re-write

$$c_{ef}^{\uparrow}(i) = c_{ef}^{0\uparrow}(i) \cdot (x_i + x_j), \tag{18}$$

$$c_{ef}^{\downarrow}(i) = c_{ef}^{0\downarrow}(i) \cdot (x_i + x_k). \tag{19}$$

Therefore, the following rules similar to those for c_a are used for c_{ef}^0: during the lower-bound computation, the upper bound of c_{ef}^0 will be used for E_i, E_j and E_k, and lower bound of c_{ef}^0 for downstream segment E_n; during the upper-bound computation, the lower bound of c_{ef}^0 will be used for E_i, E_j and E_k, and upper bound of c_{ef}^0 used for E_n.

The bound-computation for the GISS problem can be simplified when the WS-bounded model is monotonically-constrained. We first define the following *monotonically-constrained capacitance table*:

Definition 7: A capacitance table is monotonically-constrained if the following is true with respect to the basic geometric structure (see Figure 1) for any given pitch-spacing: for any two combinations of widths and spacings (x_1, s_1) and (x_2, s_2), if $x_1 \leq x_2$ (and $s_1 \geq s_2$ under the given pitch-spacing), then $c_a(x_1, s_1) \geq c_a(x_2, s_2)$ and $c_{ef}(x_1, s_1)/x_1 \geq c_{ef}(x_2, s_2)/x_2$, at the same time, $c_a(x_1, s_1) \cdot x_1 \leq c_a(x_2, s_2) \cdot x_2$ and $c_{ef}(x_1, s_1) \leq c_{ef}(x_2, s_2)$.

We say that the WS-bounded model is monotonically-constrained if its capacitance table is monotonically-constrained, and proved the following theorem in the technical report [17]):

Theorem 6: The GISS problem under the WS-bounded capacitance model is a a monotonically-constrained CH-program if the capacitance model is monotonically-constrained.

In this case, the PLR operation can be used instead of the ELR operation. To tighten a lower- (upper-) bound x_i for a wire E_i, we assume that its neighboring wires E_j and E_k have lower- (upper-) bound widths at spacings $s_i^{\uparrow}$ and $s_i^{\downarrow}$ away from E_i. We use c_a and c_{ef}^0 obtained directly using table lookup, and perform an PLR operation on x_i. Compared with the ELR operation, the PLR operation is more efficient and may lead to smaller gaps between lower and upper bounds.

In order to exploit the optimality of the ELR operation and the efficiency of the PLR operation, our implementation of the ELR operation is a hybrid of both operations. When working on a wire E_i, we first check capacitance values with respect to all valid widths and spacings for E_i,[15] then use an PLR operation if Definition 7 is satisfied. Otherwise, we use an ELR operation.

By using the ELR or PLR operation, we obtain lower and upper bounds only for the optimal total-width x_i^*. If the resulting bound is x_i, we assign $x_i^{\uparrow} = x_i^{\downarrow} = x_i/2$ for the symmetric GISS problem. Therefore, starting with the minimum and maximum symmetric wire sizing solutions for all wire segments, and using iterative ELR or PLR operations, we can compute ELR-tight lower and upper bounds for the globally

[15] A dynamic-programming scheme is used based on two-dimensional cache tables, which, similar to our capacitance tables, are indexed by widths and spacings. For given width and spacing, the cache tables return the minimum or maximum values for c_a and c_{ef}^0, or imply that the PLR operation can be used.

optimal solution to the symmetric GISS problem.

C. Bound computation for the asymmetric GISS problem

We first extend the dominance relation to consider the asymmetric wire sizing formulation. We say that the wire sizing solution $\mathbf{X}$ dominates another solution $\mathbf{X}'$ (denote as $\mathbf{X} \geq \mathbf{X}'$), if $(x_i^{\uparrow}, x_i^{\downarrow}) \geq (x'^{\uparrow}_i, x'^{\downarrow}_i)$ (i.e., $x_i^{\uparrow} \geq x'^{\uparrow}_i$ and $x_i^{\downarrow} \geq x'^{\downarrow}_i$) holds for any wire segment E_i. A lower and upper bound of the exact solution to the asymmetric GISS problem will be determined according to the new definition of dominance relation.

We solve the asymmetric GISS problem by augmenting the bound-computation algorithm presented in Section IV-B. Each ELR or PLR operation gives only the total-width x_i, which is a lower or upper bound of the optimal total-width x_i^* for E_i. To obtain an asymmetric wire sizing solution, we need to separate x_i into $x_i^{\uparrow}$ and $x_i^{\downarrow}$, which are respective widths for the "two pieces" of wires around the initial central-line of E_i. This separation is equivalent to embed a wire with total-width x_i around the initial central-line of E_i. It also affects the ELR and PLR operations in the subsequent steps. We propose to perform a *conservative embedding* right after any ELR or PLR operation.

We assume that $x_i^* = (x_i^{\uparrow *}, x_i^{\downarrow *})$ is the width for E_i in the exact asymmetric solution. Let $x_i^{\uparrow L}$ and $x_i^{\uparrow U}$ be the lower and upper bounds for $x_i^{\uparrow *}$, and $x_i^{\downarrow L}$ and $x_i^{\downarrow U}$ the lower and upper bounds for $x_i^{\downarrow *}$. If we obtain a total-width x_i^L in the lower-bound computation, the conservative embedding (*CE*) operation computes $x_i^{\downarrow L} = x_i^L - x_i^{\uparrow U}$, which is a *conservative* lower-bound for $x_i^{\downarrow *}$. Similarly, $x_i^{\uparrow L} = x_i^L - x_i^{\downarrow U}$ is a conservative lower bound for $x_i^{\uparrow *}$. Note that the sum of $x_i^{\uparrow L}$ and $x_i^{\downarrow L}$ may be *less* than x_i^L in the CE operation. Symmetrically, for an upper-bound x_i^U, we compute $x_i^{\uparrow U} = x_i^U - x_i^{\downarrow L}$, and $x_i^{\downarrow U} = x_i^U - x_i^{\uparrow L}$. This augmented algorithm leads to the lower and upper bounds of the exact solution to the asymmetric GISS problem.

We also define a greedy embedding (*GE*) operation. Recall that neighboring wires of E_i have their lower- (upper-) bound widths during lower- (upper-) bound computation for E_i. If the lower or upper bound of wire width for E_i is x_i, we find $x_i^{\uparrow}$ and $x_i^{\downarrow}$ such that $x_i^{\uparrow} + x_i^{\downarrow} = x_i$ and the objective function Eqn. (12) is minimized with respect to the given neighboring wires. Different from the CE operation, the GE operation does not always lead to a lower or upper bound of the exact solution for the asymmetrical GISS problem. We will show, however, that the GE operation has a higher convergence rate than the CE operation in experiments, and achieves satisfactory experimental results in Section IV-E. Again, we say the computation on a wire segment is *convergent* if lower and upper bounds are identical.

D. Overall algorithm for the asymmetric GISS problem

Our overall asymmetric GISS algorithm (denoted as GISS/(E)LR algorithm, see Table V) consists of the following three steps. First, we compute the ELR-tight lower and upper bounds using iterative ELR operations and CE operations. Our ELR implementation invokes PLR operations when PLR operations assure the optimality. Then, if the resulting lower and upper bounds do not meet, we will use iterative

LR operations and GE operations to further improve the lower and upper bounds. We carry out the LR operation and GE operations simultaneously as the following: for a wire segment, we enumerate width choices for two wire-pieces between lower and upper bounds, and the two widths that minimize our multiple-net objective function Eqn. (12) are the LR and GE result. Note that the first step guarantees the optimality in the sense that there exists a global exact solution within the resulting ELR-tight lower and upper bounds. However, this kind of optimality may not hold in the second step. Finally, for each net that still has non-convergent wire segments, we will assume that other nets have lower-bound wire widths, and invoke the single-net interconnect sizing and spacing (*SISS*) algorithm presented in [16] to find the final sizing and spacing solution within its lower and upper bounds. The SSIS algorithm combines the asymmetric wire sizing formulation and the wire sizing algorithm based on the bottom-up dynamic-programming technique [10].[16] We apply the SSIS algorithm in the greedy order such that the more timing-critical net is processed earlier.

GISS/(E)LR Algorithm
1. Compute ELR-tight lower and upper bounds using iterative ELR operations and CE operations; 2. Compute LR-tight "lower" and "upper" bounds using iterative LR operations and GE operations; 3. For all non-convergent nets in the greedy order, invoke single-net dynamic-programming based algorithm within resulting lower and upper bounds.

TABLE V

ASYMMETRIC GISS ALGORITHM BASED ON ELR AND LR OPERATIONS

E. Experimental results

We have tested our GISS algorithm on a 16-bit parallel bus structure. In this bus, each bit is a $1cm$ line with a 119 Ω driver resistance and a 12.0fF sink capacitance. We assume that initially these lines are equally spaced. We will find an asymmetric wire sizing for every $500\mu m$-long wire segment. In addition, the minimum wire width is $0.22\mu m$, and the minimum spacing $0.33\mu m$. The allowable wire widths are from 0.22 to 1.1 μm, with the incremental step of 0.11 μm. The capacitance tables are generated using numerical capacitance extraction for the $0.18\mu m$ technology in Table 22 of NTRS [12].

[16] The SISS problem finds the optimal wire sizing and spacing solution for a single net, under the assumption that all its neighboring wires are fixed. The GISS/(E)LR algorithm, i.e., first computing ELR-tight bounds based on the ELR operation, and then computing the final solution within bounds based on dynamic programming, can also be used to solve the SISS problem. It will be much more efficient than the purely dynamic-programming based approach in [16].

pitch-spacing	Convergence		Average gap (μm)		Average # of operations	
	ELR/CE	LR/GE	ELR/CE	LR/GE	PLR	ELR
2x	71%	100%	0.033	0.0	17.0	2.58
3x	77%	100%	0.040	0.0	29.1	2.34
4x	65%	100%	0.069	0.0	34.4	1.47
5x	57%	100%	0.090	0.0	36.6	4.43
6x	69%	100%	0.066	0.0	37.4	4.35

TABLE VI

Convergence of ELR/CE and LR/GE in GISS/(E)LR algorithm

We optimized the bus for different initial pitch-spacings, from 2x to 6x of the minimum pitch-spacing ($0.55\mu m$). Our GISS/(E)LR algorithm has two bound-computation phases, the first one using ELR/CE operations and the second one using LR/GE operations (see Table V). As shown in Table VI, computations for from 57% to 77% wire segments are convergent, i.e., identical lower and upper bounds are achieved for these segments after the ELR/CE phase. The average gap after the ELR/CE phase is between 0.033 μm and 0.090 μm. Furthermore, the LR/GE phase obtains identical lower and upper bounds for all wire segments in our examples. Therefore, very likely, our bound computation directly leads to the global and asymmetric wire sizing and spacing solution. In addition, we report the average numbers of ELR and PLR operations for a wire segment (our ELR implementation automatically invokes the PLR operation when the PLR operation does not lose the optimality). An important observation is that in most cases the PLR operation is used. It implies that the GISS problem is mainly a monotonically-constrained CH-program.

pitch-spacing	Average Delay (ns)				Run Time (s)	
	SISS	GISS/FAF	GISS/VAF	GISS/(E)LR	GISS/VAF	GISS/(E)LR
2x	1.31	0.82(-37%)	0.82(-37%)	0.79(-39%)	183	3.68
3x	0.72	0.63(-13%)	0.56(-22%)	0.52(-27%)	189	4.69
4x	0.46	0.46(+0.0%)	0.45(-2.2%)	0.42(-8.7%)	511	4.62
5x	0.38	0.39(+2.6%)	0.37(-2.6%)	0.36(-5.3%)	1083	6.82
6x	0.35	0.36(+2.9%)	0.34(-2.9%)	0.32(-8.6%)	1379	9.26

TABLE VII

Comparison of different sizing algorithms

We also presented an alternative GISS algorithm in [16]. Based on an effective-fringe property, it uses a bottom-up dynamic programming technique to compute lower and upper bounds for the global solution to the asymmetric GISS problem when c_a and c_f are constants. We call it GISS/FAF. The algorithm may be extended to use variable c_a and c_f under the WS-bounded capacitance model, and we call it GISS/VAF. In both cases, the exact solution may be *outside* the range defined by the resulting lower and upper bounds. Both GISS/FAF and GISS/VAF algorithms further use the SISS algorithm to obtain final solutions within the lower and upper bounds, whereas the GISS/(E)LR algorithm uses the lower bound as the final solution due to its high convergence. In addition, we also apply the SISS algorithm in a greedy order, which is equivalent to invoking only step 3 in the GISS/(E)LR algorithm (Table V). The SISS algorithm obtains a local-optimal solution for the GISS problem.

We compare the average HSPICE delay for solutions given by these algorithms in Table VII (average delay is our objective function). As seen from the table, the GISS/(E)LR algorithm always achieves results better than the SISS solutions, with up to 39% delay reduction. Therefore, it is important to find the globally optimal solution to the GISS problem. The improvement of the GISS/(E)LR algorithm over the SISS algorithm is reduced when the pitch spacing increases, due to the fact that the coupling capacitance is less significant for larger pitch spacings. Nevertheless, compared with the SISS algorithm, the GISS/(E)LR algorithm still reduces the average delay by 8.6% in the case of maximum pitch spacing. Because neither c_a nor c_f is a constant in DSM designs, both GISS/(E)LR and GISS/VAF algorithms obtain better results than the GISS/FAF algorithm does. The GISS/(E)LR algorithm obtains an extra delay reduction of up to 17% when compared with the GISS/FAF algorithm. Furthermore, compared to the GISS/VAF algorithm, the extra delay reduction of the GISS/(E)LR algorithm is up to 7.1%. More significantly, the GISS/(E)LR algorithm runs 100x faster. It also uses much less memory. Because the GISS/(E)LR algorithm is much faster and always achieves the best results in experiments, we suggest that the GISS/(E)LR algorithm shall be used instead of other algorithms.

V. Conclusions and Discussions

In this paper we formulated three classes of optimization problems: the simple, monotonically-constrained, and bounded CH-programs. We revealed the dominance property (Theorem 1) under the local refinement (*LR*) operation for the simple CH-program, as well as the general dominance property (Theorem 2) under the pseudo-LR (*PLR*) operation for the monotonically-constrained CH-program and under the extended-LR (*ELR*) operation for the bounded CH-program. These properties enable a very efficient polynomial-time algorithm, using the LR, PLR, or ELR operation for computing lower and upper bounds of the exact solution to any CH-program. In addition, we introduced the bundled-LR (*BLR*) operation [5], which may be used to speed up the LR, PLR and ELR operations. We also called the bound-computation algorithm as the LR-based algorithm, where LR, in general, refers to the LR, PLR, ELR or BLR operation.

We showed that the algorithm is very effective and efficient for many layout optimization problems in deep submicron (*DSM*) designs. It unifies solutions to several problems, including the single-source and multi-source wire sizing problems [4], [5], continuous wire sizing problem [19], and simultaneous driver/buffer and wire sizing problem [3], [11], [28]. Because these problems assume the simple models for the device delay and interconnect capacitance, they are all simple CH-program where the LR operation can be used for bound computations. Furthermore, we applied the bound-computation algorithm to the simultaneous transistor and interconnect sizing (*STIS*) problem, and to the global interconnect sizing and spacing (*GISS*) problem with consideration of the coupling capacitance for multiple nets. We used tables pre-computed from SPICE simulations and numerical capacitance extractions to model device delay and interconnect capacitance, so that our device and interconnect models are much more accurate than many used in previous works. We first showed that the STIS and GISS problems are, in general, bounded CH-programs, and that the GISS problem is a monotonically-constrained CH-program when the capacitance model is monotonically-constrained. We then developed the STIS algorithm based on bound-computation using the ELR operation, and the GISS algorithm based on bound-computation using the ELR and PLR operations. According to Theorem 2, our bound-computation guarantees that there exist exact solutions to the two problems between resulting lower and upper bounds. Experiments also showed that our algorithms obtained solutions close to the global optimum in the most cases. Moreover, the algorithms are *extremely* efficient. It took less than 10 seconds to optimize the largest example in this paper.

Solutions to the STIS and GISS problems, as well as other device and wire sizing problems [4], [5], [3], [28], have been integrated in the TRIO package [29]. Routines using the LR, PLR, ELR and BLR operations are shared. Note that our bound-computation algorithm is applicable to any *bounded* model for the device delay and interconnect capacitance. The bounded model simply requires that values for the device delay and interconnect capacitance be bounded. Furthermore, the bounded model can use either table-lookup or high-order complex characteristic functions. In addition, results presented in this paper can be used for both pre-layout interconnect planning, and post-layout interconnect optimization.

In this paper, we assumed that the lumped capacitance is the load capacitance. In the future, we will extend our algorithm to use the effective capacitance (C_{eff}) [14] as the load capacitance for our device model. Because the ELR operation requires only the lower and upper bounds for the load capacitance, we plan to develop methods computing the lower and upper bounds for C_{eff}, which may be more efficient than computing C_{eff} directly. The Elmore delay model is used in this paper. Several recent works [30], [31], [9] have applied the higher-order delay model. We also plan to extend the LR-based algorithm to consider the higher-order delay model.

Note that the coupling capacitance affects not only the interconnect delay, but also the signal integrity. Furthermore, the inductive effect becomes increasingly significant for global interconnects in DSM designs. We plan to develop suitable delay and noise models considering both capacitive and inductive effects, then apply the LR-based algorithm and/or other techniques. The extended algorithm, with consideration of the inductive effect and higher-order delay model, will also be applicable to the device and interconnect sizing problem in PCB and MCM layout designs. Moreover, we believe that our CH-program formulations and the LR-based algorithm can be applied to other optimization problems in the CAD field.

Acknowledgments

The authors would like to thank Dr. J. P. Fishburn at Bell Laboratories, Prof. S. S. Sapatnekar at University of Minnesota, Minneapolis, and the anonymous reviewers for their helpful comments.

References

[1] J. Cong, L. He, K. Khoo, C. Koh, and Z. Pan, "Interconnect design for deep submicron ICs," in *Proc. Int. Conf. on Computer Aided Design*, pp. 478–485, 1997.

[2] J. Cong, L. He, C.-K. Koh, and P. H. Madden, "Performance optimization of VLSI interconnect layout," *Integration, the VLSI Journal*, vol. 21, pp. 1–94, 1996.

[3] J. Cong and C.-K. Koh, "Simultaneous driver and wire sizing for performance and power optimization," in *Proc. Int. Conf. on Computer Aided Design*, pp. 206–212, Nov. 1994.

[4] J. Cong and K. S. Leung, "Optimal wiresizing under the distributed Elmore delay model," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 14, pp. 321–336, Mar. 1995.

[5] J. Cong and L. He, "Optimal wiresizing for interconnects with multiple sources," *ACM Trans. on Design Automation of Electronics Systems*, vol. 1, pp. 478–511, Oct. 1996.

[6] J. Cong and L. He, "Simultaneous transistor and interconnect sizing based on the general dominance property," in *Proc. ACM SIGDA Workshop on Physical Design*, pp. 34–39, April 1996.

[7] J. Cong and L. He, "An efficient approach to simultaneous transistor and interconnect sizing," in *Proc. Int. Conf. on Computer-Aided Design*, pp. 181–186, Nov. 1996.

[8] C. Chu and D. F. Wong, "A new approach to simultaneous buffer insertion and wire sizing," in *Proc. Int. Conf. on Computer Aided Design*, pp. 614–621, 1997.

[9] N. Menezes, R. Baldick, and L. T. Pileggi, "A sequential quadratic programming approach to concurrent gate and wire sizing," in *Proc. Int. Conf. on Computer Aided Design*, pp. 144–151, 1995.

[10] J. Lillis, C. K. Cheng, and T. T. Y. Lin, "Optimal wire sizing and buffer insertion for low power and a generalized delay model," in *Proc. Int. Conf. on Computer Aided Design*, pp. 138–143, Nov. 1995.

[11] C. P. Chen, Y. W. Chang, and D. F. Wong, "Fast performance-driven optimization for buffered clock trees based on Lagrangian relaxation," in *Proc. Design Automation Conf*, pp. 405–408, 1996.

[12] Semiconductor Industry Association, *National Technology Roadmap for Semiconductors*, 1994.

[13] K. Nabors and J. White, "Fastcap: A multipole accelerated 3-d capacitance extraction program," in *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, pp. 1447–1459, Nov. 1991.

[14] J. Qian, S. Pullela, and L. T. Pileggi, "Modeling the "effective capacitance" for the RC interconnect of CMOS gates," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 13, pp. 1526–1535, Dec. 1994.

[15] L. Vandenberghe, S. Boyd, and A. E. Gamal, "Optimal wire and transistor sizing for circuits with non-tree topology," in *Proc. Int. Conf. on Computer Aided Design*, pp. 252–259, 1997.

[16] J. Cong, L. He, C. Koh, and Z. Pan, "Global interconnect sizing and spacing with consideration of coupling capacitance," in *Proc. Int. Conf. on Computer Aided Design*, pp. 628–633, 1997.

[17] J. Cong and L. He, "Theory and algorithm of local refinement based optimization with application to transistor and interconnect sizing," Tech. Rep. 970034 (available at http://cadlab.cs.ucla.edu/ ~helei/publications.html), UCLA CS Dept, Sept. 1997.

[18] J. Cong and L. He, "An efficient technique for device and interconnect optimization in deep submicron designs," in *Proc. Int. Symp. on Physical Design*, pp. 45–51, April 1998.

[19] C. P. Chen and D. F. Wong, "A fast algorithm for optimal wire-sizing under elmore delay model," in *Proc. IEEE Int. Symp. on Circuits and Systems*, pp. 412–415, 1996.

[20] J. G. Ecker, "Geometric programming: Methods, computations and applications," *SIAM Review*, vol. 22, pp. 338–362, July 1980.

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

[22] S. S. Sapatnekar, V. B. Rao, P. M. Vaidya, and S. M. Kang, "An exact solution to the transistor sizing problem for CMOS circuits using convex optimization," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 12, pp. 1621–1634, Nov. 1993.

[23] S. S. Sapatnekar, "Wire sizing as a convex optimization problem: exploring the area-delay tradeoff," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 15, pp. 1001–1011, Aug. 1996.

[24] D. Luenberger, *Linear and Non-linear Programming.* Addison-Wesley, 1989.

[25] C. Chu and D. F. Wong, "Greedy wire-sizing is linear time," in *Proc. Int. Symp. on Physical Design*, pp. 39–44, 1998.

[26] 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," in *Proc. IEEE Int. Solid-State Circuits Conf.*, pp. 30–31, 1994.

[27] J. Cong, L. He, A. B. Kahng, D. Noice, N. Shirali, and S. H.-C. Yen, "Analysis and justification of a simple, practical 2 1/2-d capacitance extraction methodology," in *Proc. Design Automation Conf*, pp. 627–632, 1997.

[28] J. Cong, C.-K. Koh, and K.-S. Leung, "Simultaneous buffer and wire sizing for performance and power optimization," in *Proc. Int. Symp. on Low Power Electronics and Design*, pp. 271–276, Aug. 1996.

[29] J. Cong, L. He, C. Koh, and Z. Pan, *User Manual for TRIO – UCLA Interconnect Optimization Package.* UCLA CS Dept, 1998 (available at http://cadlab.cs.ucla.edu/ ~helei/publications.html).

[30] T. Xue, E. S. Kuh, and Q. Yu, "A sensitivity-based wiresizing approach to interconnect optimization of lossy transmission line topologies," in *Proc. IEEE Multi-Chip Module Conf.*, pp. 117–121, 1996.

[31] N. Menezes, S. Pullela, and L. T. Pileggi, "Simultaneous gate and interconnect sizing for circuit-level delay optimization," in *Proc. Design Automation Conf*, pp. 690–695, June 1995.

Jason Cong received his B.S. degree in computer science from Peking University in 1985, his M.S. and Ph. D. degrees in computer science from the University of Illinois at Urbana-Champaign in 1987 and 1990, respectively. Currently, he is a Professor and Co-Director of the VLSI CAD Laboratory in the Computer Science Department of University of California, Los Angeles.

His research interests include layout synthesis and logic synthesis for high-performance low-power VLSI circuits, design and optimization of high-speed VLSI interconnects, FPGA synthesis, and reconfigurable computing. He has published over 100 research papers and led over 20 research projects supported by DARPA, NSF, and a number of industrial sponsors in these areas. He served as the General Chair of the 1993 ACM/SIGDA Physical Design Workshop, the Program Chair and General Chair of the 1997 and 1998 International Symposium on FPGAs, respectively, and on program committees of many VLSI CAD conferences, including DAC, ICCAD, and ISCAS. He is an Associate Editor of ACM Trans. on Design Automation of Electronic Systems.

Dr. Cong received the Best Graduate Award from the Peking University in 1985, and the Ross J. Martin Award for Excellence in Research from the University of Illinois at Urbana-Champaign in 1989. He received the NSF Research Initiation Award and NSF Young Investigator Award in 1991 and 1993, respectively. He received the Northrop Outstanding Junior Faculty Research Award from UCLA in 1993, and IEEE Trans. on CAD Best Paper Award in 1995. He received the ACM Recognition of Service Award in 1997.

Lei He received his B.S. degree in electrical engineering from Fudan University in 1990. Currently, he is a Ph.D. candidate and a research assistant with the computer science department in the University of California at Los Angeles.

His current research interests are focused on VLSI physical design, and involve inductance and capacitance extraction, device and interconnect modeling, and layout optimization for performance, power and signal integrity. He has 20 technical publications in journals and international conferences.

Mr. He received the Top Student Award in 1987 and 1988, and the Best Graduating Student Award in 1990, all from Fudan University. He received the Motorola Fellowship from Fudan University in 1993. He received the GTE Fellowship and the Chorafas Foundation Prize for Engineering and Technology, both from the University of California at Los Angeles in 1997. He worked with VLSI-CAD Laboratory of Fudan University from 1990 to 1992, with Cadence Design Systems in the summer of 1996, and with Hewlett-Packard Research Laboratories in the summer and fall of 1998.