AN EFFICIENT TECHNIQUE FOR DEVICE AND INTERCONNECT OPTIMIZATION IN DEEP SUBMICRON DESIGNS

Jason Cong *Lei He*

Department of Computer Science
University of California, Los Angeles, CA 90095
cong@cs.ucla.edu, helei@cs.ucla.edu

ABSTRACT

In this paper, we formulate a new class of optimization problem, named the general CH-posynomial program, and reveal the general dominance property. We propose an efficient algorithm based on the extended local refinement operation to compute lower and upper bounds of the exact solution to the general CH-posynomial program. We apply the algorithm to solve the simultaneous transistor and interconnect sizing (STIS) problem under the table-based device model, and the global interconnect sizing and spacing (GISS) problem with consideration of the crosstalk capacitance. Experiment results show that our algorithm can handle many device and interconnect modeling issues in deep submicron designs and is very efficient.

1. INTRODUCTION

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

Several recent studies [2, 3, 4, 5, 6, 7] considered the simultaneous device and interconnect sizing problem. However, most of these works used over simplified models for devices and interconnects, which are not capable of modeling many DSM issues. For example, a gate of size d is modeled by an effective resistor $r_d = r_0/d$, where r_0 is the effective resistance of the unit-size gate, and is assumed *independent* of the size, input waveform slope, and output load of the gate. Moreover, the capacitance for 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 fringe capacitance for the wire. Both are assumed to be *constants.*

These assumptions, however, are no longer realistic, especially for DSM designs. For example, in Table 1, we computed the effective driver resistance r_0 via HSPICE simulation for an inverter under the rising input (i.e., r_0 of the n-transistor in the inverter) based on the representative

size = 100x						
	n-transistor			p-transistor		
c_l / t_t	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_t	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 1. Unit-size resistance r_0 for a n-transistor of different sizes, input transition times (t_t) and output loads (c_l).

$0.18\mu m$ technology in SIA roadmap [8] for two different sizes ($100x$ and $400x$ of the minimum size). Different combinations of input transition times and output loads are used for measuring. As one can see, r_0 is clearly *not* a constant. Its value may differ by a factor of 2. We also computed the capacitance of a *victim* wire centered between two neighboring wires in the same layer and both top and down grounds two-layer away from the victim (see Figure 1). We use a 3D field solver FastCap [9] and geometric parameters for the $0.18\mu m$ technology in [8]. Figure 2.(a) depicts the ground capacitance (c_g) between the victim and grounds, with each curve for different wire widths under a specific spacing as shown in Figure 1. It is seen that neither c_a nor c_f is a constant because none of these curves is linear and different curves have different intercepts. The total capacitance of the victim is $c_{total} = c_g + c_x$, where c_x is the crosstalk capacitance between the victim and the neighboring wires. One can define the effective-fringe capacitance $c_{ef} = c_f + c_x$ as in [10], and compute $c_{total} = c_a \cdot w \cdot l + c_{ef} \cdot l$. We also obtained c_{ef} under fixed pitch-spacings[1] for different widths (see Figure 2.(b)). Clearly, 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. In contrast, a device model is a *general* model if it can handle a non-constant r_0, and a capacitance model is a *general*

[1] As shown in Figure 1, *spacing* means edge-to-edge spacing, which is distinguished from *pitch-spacing*.

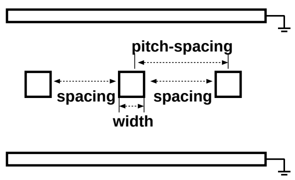

Figure 1. The geometric structure for capacitance extraction.

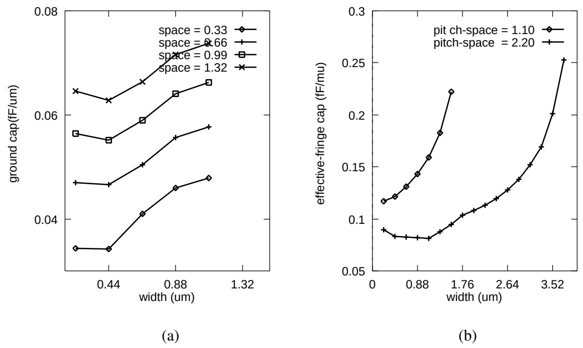

Figure 2. (a) Ground capacitances given by FastCap; (b) Effective-fringe capacitance for fixed pitch-spacings.

model if it does *not* assume that c_a and c_{ef} are constants (variable c_{ef} is necessary to handle c_x). Simple device and capacitance models were used in most previous work, except a few recent works where more accurate models were used. In [4], a sequential quadratic programming method is used to solve the simultaneous gate sizing and wire sizing problem under both simple gate model and a voltage-ramp gate model (a general model). The latter model achieves better results but is 10x slower. Two very recent works [11, 10] consider crosstalk capacitance between neighboring wires. However, both assumes that c_a and c_f are constants (but allow variable c_x). Moreover, the runtime at these algorithms is already high. For example, it took 1379 seconds to optimize a 16-bit bus of 320 wire segments in [10].

In this paper, we solve the simultaneous transistor and interconnect sizing (STIS) problem under the table-based device model, and the global interconnect sizing and spacing (GISS) problem with consideration of the crosstalk capacitance. Our algorithms are capable to apply arbitrarily accurate models for device delay and wire capacitances, using table-lookup and/or high-order complex characteristic functions, yet still guarantee to compute the lower and upper bounds of the *exact* solution very efficiently. Our implementation uses table-based models, where device-delay tables are generated using HSPICE simulations, and wire-capacitance tables are generated using 3D extractions. These table entries are very accurate, and interpolation and extrapolation are used for data points not in the tables. These table-based models are widely used in industry for verifications, but seldom for layout optimization. Experiment results show that our algorithms are very effective and extremely efficient. Compared with STIS results in [6] and GISS results in [10], up to 16.5% and 11% delay reductions are obtained, respectively. Meanwhile, a 100x speedup over the algorithm in [10] is achieved.

Our algorithm is based on a new class of optimization problem formulated in this paper. We call it the *general CH-posynomial program*, and present its formulation and property in Section 2. We solve the simultaneous transistor and interconnect sizing (STIS) problem under a table-based device model in Section 3., and the global interconnect sizing and spacing (GISS) problem considering the crosstalk capacitance in Section 4. We conclude the paper in Section 5. Proofs of all theorems are given in a technical report [12].

2. THEORY OF CH-POSYNOMIAL PROGRAMS

2.1. Review of simple and bounded-variation CH-posynomial programs

In [6], the CH-posynomial (*Cong-He* posynomial) is defined as a function of positive vector $\mathbf{X} = \{x_i \mid i = 1, \cdots, n\}$ with the following form:

$$f(\mathbf{X}) = \sum_{p=0}^{m}\sum_{q=0}^{m}\sum_{i=1}^{n}\sum_{j=1,j\neq i}^{n}\left(\frac{a_{pi}(\mathbf{X})}{x_i^p}\right)\cdot\left(b_{qj}(\mathbf{X})\cdot x_j^q\right)$$
$$where \quad a_{pi}(\mathbf{X}) \geq 0 \ and \ b_{qj}(\mathbf{X}) \geq 0 \qquad (1)$$

Then, the *simple CH-posynomial* and *bounded-variation CH-posynomial* are defined as the following:

Definition 1 (Simple CH-posynomial) *Eqn. (1) is a simple CH-posynomial if coefficients $a_{pi}(\mathbf{X})$ and $b_{qj}(\mathbf{X})$ are constants.*

Definition 2 (Bounded-Variation CH-posynomial) *Eqn. (1) is a bounded-variation CH-posynomial if coefficients satisfy the following conditions: (i) for any p and i, $a_{pi}(\mathbf{X})$ is a function depending only on x_i. With respect to an increase of x_i, $a_{pi}(\mathbf{X})$ monotonically increases for any p, but $\frac{a_{pi}(\mathbf{X})}{x_i^p}$ still monotonically decreases for any $p \neq 0$. (ii)for any q and j, $b_{qj}(\mathbf{X})$ is a function depending only on x_j. With respect to an increase of x_j, $b_{qj}(\mathbf{X})$ monotonically decrease for any q, but $b_{qj}(\mathbf{X}) \cdot x_j^q$ still monotonically increases for any $q \neq 0$.*

Note that the bounded-variation CH-posynomial is originally called the general CH-posynomial in [6]. In this paper, we rename it and use the name *general CH-posynomial* to refer to a more general formulation defined in Definition 5 later on.

We define the *CH-posynomial program* as an optimization problem to minimize a CH-posynomial Eqn. (1), 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 or bounded-variation CH-posynomial program. The dominance property is revealed based on following concepts:

Definition 3 (Dominance Relation) *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 $i = 1, \cdots, n$.*

Definition 4 (Local Refinement Operation) *For a solution vector (or simply, a solution) $\mathbf{X}'$, the local refinement operation with respect to any particular variable x_i and function $f(\mathbf{X})$ is to minimize $f(\mathbf{X})$ by only varying x_i while keeping all values of other $x_j'(j \neq i)$ in $\mathbf{X}'$ and using coefficients with respect to $\mathbf{X}'$ in case of a bounded-variation CH-posynomial.*

Such an operation is also called *LR* operation in short. The resulting solution is called the *local refinement* of $\mathbf{X}'$ (with respect to x_i).

Theorem 1 (Dominance Property) *Let $f(\mathbf{X})$ be a simple or bounded-variation CH-posynomial, and $\mathbf{X}^*$ an exact solution to minimize $f(\mathbf{X})$. For any solution $\mathbf{X}'$ of $f(\mathbf{X})$, if $\mathbf{X}'$ dominating $\mathbf{X}^*$, a local refinement of $\mathbf{X}'$ still dominates $\mathbf{X}^*$; if $\mathbf{X}'$ dominated by $\mathbf{X}^*$, a local refinement of $\mathbf{X}'$ is still dominated by $\mathbf{X}^*$.*

2.2. Theory of general CH-posynomial program

In this paper, we propose the following *general CH-posynomial.*

Definition 5 (General CH-posynomial) *Given a lower bound $\mathbf{L}$ and an upper bound $\mathbf{U}$ of the solutions, Eqn. (1) is a general CH-posynomial, if coefficients are functions of vector $\mathbf{X}$, and for $\mathbf{L} \leq \mathbf{X} \leq \mathbf{U}$, the value of any coefficient is bounded, i.e., for any p and i, there exist a_{pi}^{min} and a_{pi}^{max} such that $a_{pi}^{min} \leq a_{pi}(\mathbf{X}) \leq a_{pi}^{max}$, and for any q and j, there exist b_{qj}^{min} and b_{qj}^{max} such that $b_{qj}^{min} \leq b_{qj}(\mathbf{X}) \leq b_{qj}^{max}$.*

We extend our definition of local refinement operation to consider a general CH-posynomial program to minimize a general CH-posynomial.

Definition 6 (Extended Local Refinement Operation) *For any solution $\mathbf{X}'$, the extended local refinement operation with respect to any particular variable x_i and general CH-posynomial $f(\mathbf{X})$ is to minimize $f(\mathbf{X})$ only by varying x_i while keeping the value of any $x_j'(j \neq i)$ in $\mathbf{X}'$ and using the following coefficients:*
(i) For $\mathbf{X}' \geq \mathbf{X}^*$ and any p, we use a_{pi}^{max} instead of $a_{pi}(\mathbf{X}')$ for $\frac{a_{pi}(\mathbf{X}')}{x_i^p}$ and any i, and a_{pj}^{min} instead of $a_{pj}(\mathbf{X}')$ for $\frac{a_{pj}(\mathbf{X}')}{x_j^p}$ and any $j \neq i$; we also use b_{pi}^{min} instead of $b_{pi}(\mathbf{X}')$ for $b_{pi}(\mathbf{X}') \cdot x_i^p$ and any i, and b_{pj}^{max} instead of $b_{pj}(\mathbf{X}')$ for $b_{pj}(\mathbf{X}') \cdot x_j^p$ and $j \neq i$.
(ii) For $\mathbf{X}' \leq \mathbf{X}^*$ and any p, we use a_{pi}^{min} instead of $a_{pi}(\mathbf{X}')$ for $\frac{a_{pi}(\mathbf{X}')}{x_i^p}$ and any i, and a_{pj}^{max} instead of $a_{pj}(\mathbf{X}')$ for $\frac{a_{pj}(\mathbf{X}')}{x_j^p}$ and any $j \neq i$; we also use b_{pi}^{max} instead of $b_{pi}(\mathbf{X}')$ for $b_{pi}(\mathbf{X}') \cdot x_i^p$ and any i, and b_{pj}^{min} instead of $b_{pj}(\mathbf{X}')$ for $b_{pj}(\mathbf{X}') \cdot x_j^p$ and any $j \neq i$.

We say that the result solution is the *extended local refinement* of $\mathbf{X}'$ (with respect to x_i). Later on, we use *ELR* to denote the extended local refinement.

We have proved the following theorem:

Theorem 2 (General Dominance Property) *Let $\mathbf{X}^*$ be an exact solution to minimize a general CH-posynomial $f(\mathbf{X})$. For any $\mathbf{X}'$ dominating $\mathbf{X}^*$, an extended local refinement of $\mathbf{X}'$ still dominates $\mathbf{X}^*$; For any $\mathbf{X}'$ dominated by $\mathbf{X}^*$, an extended local refinement of $\mathbf{X}'$ is still dominated by $\mathbf{X}^*$.*

We say that a solution $\mathbf{X}$ is the lower bound of the exact solution $\mathbf{X}^*$ if $\mathbf{X}$ is dominated by $\mathbf{X}^*$, and $\mathbf{X}$ is an upper bound of $\mathbf{X}^*$ if $\mathbf{X}$ dominates $\mathbf{X}^*$. A lower or upper bound is *ELR-tight* if it can not be improved by any ELR operation. Based on the general dominance property, we propose a simple ELR-based algorithm (see Table 2) to compute the ELR-tight lower and upper bounds.

Starting with the initial lower and upper bounds ($\mathbf{L}$ and $\mathbf{U}$), the algorithm carries out *interleave* passes of lower- and upper-bound computations. A *pass* of lower bound computation is to perform an ELR operation on every x_i of a lower bound $\mathbf{X}$. The ELR operations can be in any 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 is to perform an ELR operation on any 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. Because the range of coefficients in a general CH-posynomial depend on the size of the solution space, lower- and upper-bound computations are carried out alternately to narrow the range of the coefficients. The algorithm is optimal in the sense that there exists an exact solution within the result ELR-tight lower and upper bounds. We will use the algorithm to solve the device and wire sizing problems to be formulated in the next section under general device and capacitance models.

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

Table 2. ELR-based bound-computation algorithm

3. STIS PROBLEM USING GENERAL DEVICE MODEL

3.1. Problem Formulation

We use the transistor sizing formulation in this paper. Similar to [6], our delay formulation is based on the delay for a stage. A stage 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 the source and N_t being the sink can be written as Eqn. (2) under the Elmore delay model.

$$\begin{aligned} & t(P(N_s, N_t), \mathbf{X}) \\ = & \sum_{i,j} f(i,j) \cdot \frac{r_0(i)}{x_i} \cdot c_a(j) \cdot x_j + \sum_{i,j} f(i,j) \cdot \frac{r_0(i)}{x_i} \cdot c_{ef}(j) \\ + & \sum_i g(i) \cdot \frac{r_0(i)}{x_i} + \sum_i r_0(i) \cdot h(i) + \sum_i h(i) \cdot \frac{r_0(i)}{x_i} \quad (2) \end{aligned}$$

where x_i is the width for a transistor M_i or a wire E_i, $r_0(i)$ is the unit-size resistance, and $c_a(i)$ and $c_{ef}(i)$ are the unit-area and effective-fringe capacitances. 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, it is proposed to minimize the weighted delay $t(\mathbf{X})$ of all stages in the set of critical paths denoted as $\mathcal{P}$:

$$t(\mathbf{X}) = \sum_{P(N_s,N_t) \in \mathcal{P}} \lambda \cdot t(P(N_s, N_t), \mathbf{X}) \quad (3)$$

where the weight λ indicates the criticality of stage $P(N_s, N_t)$. After we eliminate those terms independent of $\mathbf{X}$, Eqn. (3) is re-written as

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

$$+ \quad \sum_i G(i) \cdot \frac{r_0(i)}{x_i} + \sum_i H(i) \cdot \frac{r_0(i)}{x_i} \tag{4}$$

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 (**L** and **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. (4) is minimized.*

Note that a sequence of sizing problems to minimize weighted delay can be used to minimize the maximum delay by adjusting the weight assignment based on the Lagrangian-relaxation method as in [5]. Therefore, we focus on how to minimize weighted delay in this paper. In addition, we find *discrete* width from a finite width set determined by the technology. This discrete sizing formulation is more practical and more difficult than the continuous sizing formulation.

3.2. Property and Algorithm

When r_0, c_a and c_{ef} are constants under the simple models, Eqn. (4) is a simple CH-posynomial. In this case, the STIS problem is a simple CH-posynomial problem solved in [6]. Because the simple models are no longer valid for DSM designs, we study the STIS problem under a general device model where r_0 is not a constant. For simplicity, we assume that c_a and c_{ef} are constants, and will remove the assumption in Section 4.

The table-based model is a general model. In our table-based model, as shown in Table 1, values for r_0 are precomputed and stored in three-dimensional tables indexed by the transistor size, input slope and output load. This model could be very accurate depending on the table size. Given the fact that r_0 depends on the transistor size, input transition time and output load and that there is a large range for r_0, r_0 is unlikely a function of any single sizing variable. It is necessary to treat it as a function of the whole sizing solution $\mathbf{X}$. Therefore, we have the following Theorem 3:

Theorem 3 *The STIS problem under the table-based device model is a general CH-posynomial program.*

Based on Theorem 3, the ELR-based algorithm (Table 2) can be used to compute the lower and upper bounds for the exact solution to the STIS problem. The ELR operation is used for transistors. In an ELR operation on a transistor M_i for the lower bound computation, we use $r_0^{min}(i)$ (instead of $r_0(i)$), and $r_0^{max}(j)$ (instead of $r_0(j)$) for any transistor M_j other than M_i, where $r_0^{min}(i)$ is the minimum possible value for $r_0(i)$ and $r_0^{max}(j)$ is the maximum possible value for $r_0(j)$. Symmetrically, in an ELR operation on M_i for the upper bound computation, we use $r_0^{max}(i)$ (instead of $r_0(i)$) for M_i, and $r_0^{min}(j)$ (instead of $r_0(j)$) for any transistor M_j other than M_i, where $r_0^{max}(i)$ is the maximum possible value for $r_0(i)$ and $r_0^{min}(j)$ is the minimum possible value for $r_0(j)$.

We determine the minimum and maximum values for r_0 according to current lower and upper bounds. We assume that r_0 increases with respect to an increase of the transistor size and input transition time (the input t_t), but it decreases with respect to an increase of output load c_l. Therefore, $r_0^{min}(i)$ for M_i can be obtained by table lookup using the lower bound of size x_i, the lower bound of the input t_t and the upper bound of c_l. We use c_l under the current upper bound of the sizing solution as the upper bound of c_l. We set two initial lower and upper bounds for t_t, and update these two bounds during optimization procedure by assuming that the lower bound of the output t_t for M_i occurs when M_i is driven by a lower bound of the input t_t and is driving the upper bound of c_l. Symmetrically, $r_0^{max}(i)$ is determined using the upper bound of x_i, the upper bound of input t_t and the lower bound of c_l. As the lower and upper bounds of sizing solution 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^{max} and r_0^{min}, the tighter 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. In addition, the optimal wire widths are monotonic within each wire segment. Therefore, we use the bundled refinement operation [13] instead of LR operation for wire segment E_i. The bundled refinement operation is a speedup scheme for the LR operation, and shown to be 100x faster than the LR operation for the wiresizing problem.

Let $\mathbf{L}'$ and $\mathbf{U}'$ be the 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 table-based device model. Otherwise, we traverse all wire segments and transistors by iterative LR operations until there is no improvement in the last round of traversal. This procedure is bounded by $\mathbf{L}'$ and $\mathbf{U}'$, and is invoked twice starting with $\mathbf{L}'$ and $\mathbf{U}'$, respectively. We use the better solution from the two runs as the final solution. Even though this type of LR operation may lead to further improvement over $\mathbf{L}'$ and $\mathbf{U}'$, in general, it does *not* leads to a lower or upper bound of the exact solution.

3.3. Experiment results

In this section, we apply our STIS algorithm to two global nets. One is a $2cm$ line with 5 buffers optimally inserted for delay minimization. The other is a buffered tree, the *dclk* net in a spread spectrum IF transceiver chip design [14]. There are 117 drivers and 37 buffers with total wire length of 41518.2 μm. We use parameters based on the 0.18 μm technology given in [8]. The wire sheet-resistance $R_\square = 0.0638\Omega$. Based on parameters given in [8], we generate device tables using HSPICE, and use c_a and c_{ef} values when the wire is $1.10\mu m$ wide and neighboring wires are $1.65\mu m$ away.

We compare sizing solutions obtained under different device models, simple model versus table-based model. We also use different sizing formulations, simultaneous gate and wire sizing (*sgws*) versus simultaneous transistor and wire sizing (*stis*). There are four combinations, including *sgws/simple* and *stis/simple* using the LR-based algorithm as in [6], and *stis/simple* and *stis/table* using new developed ELR-based algorithm. The value for r_0 in the simple model is determined under the typical input, device size and output load. We assume the fixed ratio between p- and n- transistors for the gate sizing formulation is simply 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 allowable wire widths are multiples of $0.56\mu m$ between $0.56\mu m$ and 5.6 μm.

Table 3 gives experimental comparison between different

net	sgws/simple	sgws/table	stis/simple	stis/table	sgws/simple	sgws/table	stis/simple	stis/table
	convergence for transistors				convergence 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	1.007(-6.4%)	1.132(-2.3%)	0.961 (-15.1%)	1.18	2.32	0.88	3.17
line	0.821	0.818(-0.4%)	0.751(-8.6%)	0.694(-16.5%)	0.72	0.58	0.55	1.22

Table 3. Comparisons between different device and wire sizing algorithms: sgws/simple – simultaneous gate and wire sizing under simple model, stis/simple – simultaneous transistor and wire sizing under simple model, and stis/table – simultaneous transistor and wire sizing under table-based model.

sizing formulations. We computed *convergence* for transistors and wires. A transistor or wire is *convergent* if its lower and upper bounds given by the LR- or ELR-based algorithms are identical. It is seen that the convergence are not significantly different. For example, transistors in *dclk* net have about 85% transistor convergent under all four formulations. We also computed the average width and the average gap between lower and upper bounds. The ELR-based algorithm does give larger gap than the LR-based algorithm. However, the difference is small. Overall, the average gap is only 1% of the average width, except for transistors in net *dclk*. Therefore, the ELR-based algorithm gives solutions which are close to the exact solution under the table-based device model.

Given that the ELR-tight lower and upper bounds are close to each other, we simply use the lower bound as the final solution. We computed the maximum delays via HSPICE using the distribute RC model and the level-3 MOSFET model. When compared with the *sgws/simple* formulation, *sgws/simple*, *stis/simple* and *stis/table* formulations reduce the maximum delay by up to 6.4%, 8.6%, 16.5%, respectively. The solutions under the table-based device model are consistently better than those under the simple device model. Although the ELR-based algorithms for the table-based device model has longer runtimes, the maximum runtime is just 3.17 seconds. Therefore, our ELR-based algorithms is effective efficient for the STIS problem.

4. GISS PROBLEM CONSIDERING CROSSTALK CAPACITANCE

The constant c_a and c_{ef} are assumed for the STIS problem in Section 3. We proceed to remove this assumption by using a general capacitance model. For simplicity of presentation, we assume that the device sizes are fixed and study the global (multi-net) wire sizing and spacing (GISS) problem in this section. However, our algorithm and implementation are able to use general models for both device and capacitance at the same time.

4.1. Problem formulation

Our general capacitance model is a 2D model simplified from the 2.5D model in [15]. We consider the area, fringe and crosstalk capacitances for a wire in the 2D model. We assume that c_a and c_{ef} are functions of widths and spacings (see Figure 1). Based on this assumption, we first use a 3D field solver like FastCap to build tables for c_a and c_{ef} under different width and spacing combinations. Then, table lookup is used during layout optimization to obtain c_a and c_{ef} for the given wire width and spacing.

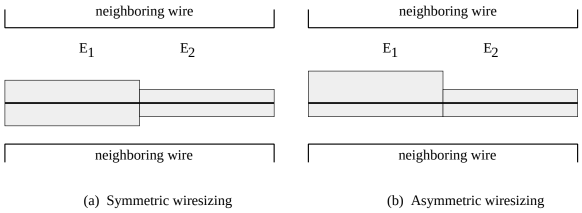

Figure 3. (a) Symmetric wire sizing, and (b) Asymmetric wire sizing.

Our GISS formulation was first presented in [10]. It assumes that an initial layout is *a priori* given and that the *initial central-lines* and *initial pitch-spacings* defined by the initial layout remain *unchanged* during the sizing procedure. Even though $c_a(i)$ and $c_{ef}(i)$ for a wire segment E_i are functions of width x_i and spacings in the 2D capacitance model, they are functions only explicitly depends on width x_i. Therefore, we can still use the delay formulation Eqn. (4).

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 3(a). In contrast, in the *asymmetric* wire sizing formulation shown in Figure 3(b), wires of same widths are asymmetric with respect to initial central-lines, and has smaller capacitance and delay. Given that neighboring wires are in general asymmetrically away from interested nets, the asymmetric wire sizing formulation is capable to further reduce the interconnect delay.

In the asymmetric formulation, the wire sizing solution for wire segment E_i is needed to be represented by a pair of widths $(x_i^{\uparrow}, x_i^{\downarrow})$, where $x_i^{\uparrow}$ is the width of the wire above (or left to) the initial central-line when E_i is a horizontal (or vertical) segment, and $x_i^{\downarrow}$ is the width of the wire on the other side of the initial central-line. In order to maintain the connectivity, we assume that $x_i^{\uparrow}$ and $x_i^{\downarrow}$ are at least $W_{min}/2$, where W_{min} is the minimum wire width set by the manufacture technology. We first present algorithms for the symmetric wire sizing formulation, then extend the algorithms to consider the asymmetric wire sizing formulation.

4.2. Algorithm for symmetric GISS problem

We often observes the following (like the case of pitch-spacing = 1.10 μm in Figure 2.(b)):

Observation 1 *In a geometric structure as in Figure 1 where the central wire E_i has two neighboring wires at equal and fixed pitch-spacings, if the width x_i of E_i increases symmetrically with respect to its initial central-line, then $c_a(i)$ decreases, but both $c_a(i) \cdot x_i$ and $c_{ef}(i)$ increase.*

We have proved that

Theorem 4 *The GISS problem is a bounded-variation CH-posynomial program if each wire segment satisfies Observation 1 for any valid widths (and spacings).*

In this case, the LR operation can be used to replace the ELR operation in the ELR-based algorithm (Table 2). For example, to tighten a lower bound x_i for a horizontal wire E_i, we assume that its neighboring wires have lower-bound width and define top spacing $s_i^{\uparrow}$ and down spacing $s_i^{\downarrow}$ for E_i. We derive unit area-capacitance $c_a(x_i, s_i^{\uparrow}, s_i^{\downarrow})$ and unit effective-fringe capacitance $c_{ef}(x_i, s_i^{\uparrow}, s_i^{\downarrow})$ according to x_i, $s_i^{\uparrow}$ and $s_i^{\downarrow}$ and perform an LR operation on x_i. The result local refinement of x_i moves closer to but remains smaller than x_i^*, the width of E_i in the exact solution to the GISS problem as a bounded-variation CH-posynomial program. Similarly, we assume that neighboring wires have upper-bound widths in order to perform an LR operation on the upper-bound width of wire E_i.

Observation 1 does *not* always hold. For example, for a large enough initial-spacing, if width increases (and spacing decreases), then c_f decreases and c_x increases, which results in $c_{ef} = c_f + c_x$ being a non-monotonic function of wire width and Observation 1 fails (see the case of pitch-spacing $= 2.2\mu m$ in Figure 2.(b)). Therefore, we have Theorem 5:

Theorem 5 *The GISS problem under the general capacitance model is a general CH-posynomial program.*

In the case, the ELR operation is needed in the ELR-based algorithm (Table 2). Let $c_a^{min}(i)$ and $c_a^{max}(i)$ be maximum and minimum values for $c_a(i)$, and $c_{ef}^{min}(i)$ and $c_{ef}^{max}(i)$ be the minimum and maximum values for $c_{ef}(i)$. With respect to these values and delay formulation Eqn. (4), we perform the ELR operation for the lower-bound computation on a wire E_i, by using $c_a^{max}(i)$ and $c_{ef}^{min}(i)$ (instead of $c_a(i)$ and $c_{ef}(i)$) for E_i, and using $c_0^{min}(j)$ and $c_{ef}^{min}(j)$ (instead of $c_0(j)$ and $c_a(j)$) for any edge E_j other than E_i. Similarly, the ELR operation for the upper-bound computation of E_i can be performed by using $c_a^{min}(i)$ and $c_{ef}^{max}(i)$ (instead of $c_a(i)$ and $c_{ef}(i)$) for E_i, and using $c_0^{max}(j)$ and $c_{ef}^{max}(j)$ (instead of $c_0(j)$ and $c_a(j)$) for any edge E_j other than E_i.

We combine LR and ELR operations in our bound-computation algorithm. When working on a wire E_i, we first check capacitance values with respect to all valid widths and spacings of E_i, then use either an LR or an ELR operation according to Observation 1. If the result wire width is x_i for E_i, then $x_i^{\uparrow} = x_i^{\downarrow} = x_i/2$. Therefore, starting with the minimum and maximum symmetric wire sizing solutions for all wire segments, the algorithm leads to lower and upper bounds of the exact global solution to the symmetric GISS problem.

4.3. Algorithm for asymmetric GISS problem

We first extend the dominance relation for 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 4.2. Each LR or ELR 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 map 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. Physically, this mapping is equivalent to embed a wire with total-width x_i around the initial central-line of E_i. This embedding also affects the LR and ELR operations in the subsequent steps.We propose to perform a *conservative embedding* right after any LR or ELR operation. This augmented algorithm will lead to the lower and upper bounds of the exact solution to the asymmetric GISS problem.

In the conservative embedding, without loss of generality, we assume that E_i is a horizontal wire. We keep lower and upper bounds for the widths of both upper-piece and lower-piece of E_i. Let $\underline{x_i^{\uparrow}}$ and $\overline{x_i^{\uparrow}}$ be current lower and upper bounds for the upper-piece width $x_i^{\uparrow *}$, and $\underline{x_i^{\downarrow}}$ and $\overline{x_i^{\downarrow}}$ be current lower and upper bounds for the lower-piece width $x_i^{\downarrow *}$. When we obtain a total-width x_i in the lower-bound computation, we update the lower bound of the lower-piece width as $x_i - \overline{x_i^{\uparrow}}$, which is the difference between the the lower bound of the total-width and the upper bound for the upper-piece width and is a *conservative* lower bound for the lower-piece width. Similarly, we update the lower bound of the upper-piece width as $x_i - \overline{x_i^{\downarrow}}$. Note that the sum of the lower bounds of widths for the two piece wires may be less than the lower bound of the total-width in the conservative embedding. Symmetrically, when we obtain a total-width x_i in the upper-bound computation, the new $\overline{x_i^{\uparrow}} = x_i - \underline{x_i^{\downarrow}}$, and the new $\overline{x_i^{\downarrow}} = x_i - \underline{x_i^{\uparrow}}$. The conservative embedding is also used to prove the asymmetric effective-fringing property in [10].

We also propose a greedy embedding. We assume that neighboring wires of E_i have their lower- (upper-) bound widths during lower- (upper-) bound computation for E_i, and then find $x_i^{\uparrow}$ and $x_i^{\downarrow}$ such that $x_i^{\uparrow} + x_i^{\downarrow} = x_i$ and the capacitance for E_i is minimized. This heuristic embedding leads to good experimental results as discussed in [12].

4.4. Experiment 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 these lines are initial equally spaced and find an asymmetric wire sizing for every $500\mu m$-long wire segment. In addition, the minimum wire width is $0.22\mu m$. The minimum spacing is $0.33\mu m$. 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 3D field solver FastCap for the $0.18\mu m$ technology in [8].

We call the GISS algorithm presented in this paper GISS/ELR algorithm. An alternative GISS algorithm was presented in [10] based on bottom-up dynamic programming technique. It computes lower and upper bounds for the exact solution to the asymmetric GISS problem when c_a and c_f are constants, and we denote it as GISS/FAF. It may be extended to use variable c_0 and c_f in a general capacitance model. In this case, the exact solution may be

pitch-spacing	Average Delay (ns)				Run Time (s)	
	MIN	GISS/FAF	GISS/VAF	GISS/ELR	GISS/VAF	GISS/ELR
2x	1.51	0.79(-48%)	0.79(-48%)	0.79(-48%)	183	3.68
3x	1.32	0.56(-58%)	0.53(-60%)	0.52(-61%)	189	4.69
4x	1.27	0.46(-64%)	0.42(-67%)	0.42(-67%)	511	4.62
5x	1.24	0.39(-69%)	0.37(-70%)	0.36(-71%)	1083	6.82
6x	1.22	0.36(-70%)	0.34(-72%)	0.32(-74%)	1379	9.26

Table 4. Comparison of different sizing algorithm when sizing 16-bit buses under 2x-6x minimum pitch-spacing. MIN is the minimum wire width (and thus maximum spacing) solution; GISS/FAF and GISS/VAF are bottom-up dynamic programming algorithms; GISS/ELR is the algorithm presented in this paper.

outside the range defined by the resulting lower and upper bounds, and we denote it as GISS/VAF.

We optimized the bus for different initial pitch-spacings, from 2x to 6x of the minimum pitch-spacing ($0.55\mu m$). We report the average HSPICE delay among all sinks in Table 4. The MIN is the solution with minimum wire width and thus largest spacing to reduce the coupling capacitance. It serves as the base for delay comparison. GISS/FAF and GISS/VAF further use a greedy algorithm to obtain final solutions within the lower and upper bounds, whereas GISS/ELR uses the lower bound as the final solution due to its higher convergence. All GISS algorithms lead to solutions much better than the MIN solution. Because the GISS problem is no longer a bounded-variation CH-posynomial program in case of large pitch-spacings, GISS/ELR achieves more improvement (11% better than GISS/FAF and 5.9% better than GISS/VAF for 6x minimum pitch-spacing). GISS/ELR is also 100x faster and uses much less memory. Detailed analysis on memory usage and convergence of bounds is included in [12].

5. CONCLUSIONS

We formulated a new class of optimization problem, named the general CH-posynomial program, and propose an algorithm to compute lower and upper bounds of the exact solution to the general CH-posynomial program. We applied the algorithm to solve device and wire sizing problems, with consideration of DSM issues like the table-based models for device delay and interconnect capacitances including crosstalk capacitance between neighboring wires. Our algorithm achieves more delay reduction when compared with previous work, and is also *extremely* efficient. We plan to extend the algorithm to consider the higher-order delay model in the future. We believe that our general CH-posynomial formulation and the bound-computation algorithm can also be applied to other optimization problems in the CAD field.

ACKNOWLEDGMENTS

This work is partially supported the NSF Young Investigator Award MIP-9357582 and a grant from Intel Corporation under the California MICRO program. The authors would like to thank the anonymous reviewers for helpful comments.

REFERENCES

[1] 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.

[2] 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.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

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

[9] 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.

[10] J. Cong, L. He, C. Koh, and Z. Pan, "Global interconnect sizing and spacing with consideration of coupling capacitance," Tech. Rep. 970031, UCLA CS Dept, 1997.

[11] 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.

[12] J. Cong and L. He, "Theory and algorithm of local refinement based optimization with application to transistor and interconnect sizing," Tech. Rep. 970034, UCLA CS Dept, Sept. 1997.

[13] J. Cong and L. He, "Optimal wiresizing for interconnects with multiple sources," in *Proc. Int. Conf. on Computer Aided Design*, pp. 568–574, Nov. 1995.

[14] 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 $1.2\mu m$ CMOS," in *Proc. IEEE Int. Solid-State Circuits Conf.*, pp. 30–31, 1994.

[15] J. Cong, L. He, A. B. Kahng, D. Noice, S. N., 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.