Title
Extended global routing with RLC crosstalk constraints

Journal
IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS, 13(3)

ISSN
1063-8210

Authors
Xiong, J J
He, L

Publication Date
2005-03-01

Peer reviewed

Extended Global Routing With *RLC* Crosstalk Constraints

Jinjun Xiong, *Student Member, IEEE,* and Lei He, *Member, IEEE*

***Abstract*—In this paper, we study an extended global routing problem with *RLC* crosstalk constraints. Considering simultaneous shield insertion and net ordering, we propose a multiphase algorithm to synthesize a global routing solution with track assignment to satisfy the *RLC* crosstalk constraint at each sink. The key algorithm phase is global routing synthesis with shield reservation and minimization based on prerouting shield estimation. Experiments using large industrial benchmarks show that compared to the best alternative with postrouting shield insertion and net ordering, the proposed algorithm with shield reservation and minimization reduces the congestion by 18.4% with a smaller runtime. To the best of our knowledge, this is the first in-depth study on global routing synthesis with *RLC* crosstalk constraints.**

***Index Terms*—Global routing, net ordering, *RLC* crosstalk, shielding, signal integrity.**

I. Introduction

As VLSI technology advances, crosstalk becomes increasingly critical [1]. The following works have been developed to reduce the capacitive crosstalk, including net ordering [2], spacing [3], and layer assignment [4] within a routing channel or switchbox. To overcome the limited flexibility of channel or switchbox routing, pseudo pin assignment [5] has also been developed at the full chip level. A global routing adjustment procedure [6] has been developed via iterative region-based crosstalk estimation and reduction considering track assignment. Furthermore, an extended global routing problem has been proposed in [7] to take into account layer/track assignment under the length scaled capacitive crosstalk model.

Given a global routing solution and assuming the crosstalk bound is given for each net segment within a routing region, a few recent works have addressed crosstalk avoidance techniques for both capacitive and inductive crosstalk. Examples include shielding [8], simultaneous shield insertion and net ordering (SINO) [9], staggered repeater placement [10], twisted bundle wires [11], and differential signaling [12]. However, there has been no in-depth study on automatic global routing that is able to consider the above techniques at the full chip level for both capacitive crosstalk and inductive crosstalk reduction. As opposed to capacitive crosstalk that exists only between adjacent wires, inductive crosstalk has a long-range effect, i.e., it may affect both adjacent and nonadjacent wires. Therefore, global routing considering inductive crosstalk is more difficult than global routing considering only capacitive crosstalk [7]. In [14], a postglobal routing optimization technique is proposed to consider the full chip *RLC* crosstalk constraints. However, as the global routing is already given, there are very limited design freedoms for the postglobal routing procedure to leverage.

In this paper, we study an extended global routing synthesis problem with *RLC* crosstalk constraints, and develop a multiphase algorithm to solve it effectively. The proposed algorithm synthesizes an extended global routing solution with no *RLC* crosstalk violation by leveraging the simultaneous SINO technique at the full-chip level. The key algorithm phase is global routing synthesis with shield reservation and minimization based on prerouting shield estimation. Experiments using large industrial benchmarks show that compared to the best alternative with postrouting shield insertion and net ordering, the proposed algorithm reduces the congestion by 18.4% with a smaller runtime. The effectiveness of considering prerouting shield reservation and minimization in routing synthesis stage is the primary contribution of this paper. To the best of our knowledge, this is also the first in-depth study on global routing that is able to accommodate the interconnect synthesis technique with *RLC* crosstalk constraints.

The rest of the paper is organized as follows: in Section II, we introduce the routing model and *RLC* crosstalk model used in this paper and formulate an extended global routing synthesis (GSINO) problem; in Section III we propose a multiphase algorithm to solve the GSINO problem efficiently; in Section IV, we present our experiment results. We conclude this paper with the discussion of our future work in Section V.

II. Modeling and Problem Formulation

A. Routing Model

Over-the-cell (OTC) routing has been studied extensively for large design cases. In this paper, we consider OTC routing style for global interconnects, and use the same routing model as in [15], [16], and [17]. We summarize the notations frequently used in this paper in Table I.

We assume that the routing area is divided by the prerouted power/ground (P/G) networks into rectangular partitions as *global bins*, and all cells along with their connection pins are placed at the center of global bins. Single-source-multisink (SSMS) nets are considered in this paper. Fig. 1(a) shows a circuit after layout that illustrates the definition of global bins and a routing for a net with one source and three sinks.

The routing area can be formally modeled by an undirected graph $G(V, E)$ in Fig. 1(b), where each vertex $v \in V$ represents

Manuscript received July 23, 2003; revised February 11, 2004. This work was supported in part by NSF CAREER award CCR-0093273, a UC MICRO Grant sponsored by Fujitsu Laboratories of America, Intel, and LSI Logic, and by a Faculty Partner Award by IBM.

The authors are with the Department of Electrical Engineering, University of California, Los Angeles, CA 90095 USA (e-mail: jinjun@ee.ucla.edu; lhe@ee.ucla.edu).

Digital Object Identifier 10.1109/TVLSI.2004.842896

TABLE I
NOTATIONS THAT ARE FREQUENTLY USED IN THIS PAPER

R_t	routing regions in a chip
l_t	length of region R_t
C_t	total number of tracks in region R_t
O_t	total number of tracks occupied by obstacles in region R_t
G_t	set of net segments in region R_t
$\|G_t\|$	total number of net segments in region R_t
S_t	set of shield segments in region R_t
$\|S_t\|$	total number of shield segments in region R_t
Ov_t	segment overflow in region R_t
N_i	signal net
p_{i0}	source pin of net N_i
p_{ij}	j^{th} sink of net N_i
l_{ij}	routing length from p_{i0} to p_{ij}
P_i	set of all sinks for net N_i
$\|P_i\|$	total number of sinks for net N_i
N_{it}	net segment of net N_i in region R_t
n_i	total number of net segments in the route for net N_i
r_{it}	physical sensitivity rate of N_{it} in region R_t
H_{ij}	set of regions containing the route for sink p_{ij} of net N_i
L_{it}	self inductance for N_{it}
$M_{it,jt}$	mutual inductance between N_{it} and N_{jt}
$K_{it,jt}$	inductive coupling coefficient between N_{it} and N_{jt}
K_{it}	total inductive coupling for net segment N_{it}
$\overline{K_{it}}$	bound of K_{it}
LSK_{ij}	LSK value of sink p_{ij} of net N_i
$\overline{LSK_{ij}}$	bound of LSK at sink p_{ij} of net N_i
$sign(x)$	$sign(x) = 1$ if $x > 0$, otherwise, $sign(x) = 0$

a global bin, and each edge $e \in E$ represents the routing area between two adjacent bins and has a length l_t. An edge in the routing graph is also called a *routing region* R_t. To model the limited routing resources, we associate each edge in $G(V, E)$ with a *capacity* C_t, which is defined as the maximum number of tracks available for routing. The capacity is decided by the geometry of the design and the technology used. In multilayer designs, an edge may consist of more than one layer. We assume that each track can be used by only one net segment, and we can accommodate multilayer design by increasing the capacity of each edge. Similar to [7], an extended global routing solution not only decides the regions that every net is routed through, i.e., the set of edges connecting all nodes (routing regions) that contain pins for the net, but also determines the track assignment for both signal nets and shields in each region. For a track assignment solution in routing region R_t, a *block* includes all routing tracks between two adjacent shield segments. Fig. 2 shows such a block in R_t, where n_{it} and n_{jt} are track ordering numbers for net segments N_{it} and N_{jt}, respectively, and s_{lt} and s_{rt} are track ordering numbers for the two edge shield segments.

B. Region Based Crosstalk Model and SINO Technique

P/G network is usually designed as a mesh structure [18]. P/G wires in a mesh structure can not only reduce the capacitive coupling between signal nets, but also provide a closer current return paths for signal switching, and thus reduce inductive coupling. Therefore, for a well designed P/G network, we can assume that there is no crosstalk (coupling) between different regions separated by P/G wires[1]. For simplicity, we further assume that two signal nets are *logically sensitive* to each other[2] if, through logic synthesis or timing analysis [19], a switching event on one signal net (aggressor) causes the other (victim) to malfunction due to extraordinary crosstalk noise, and vice versa. The *logic sensitivity rate* of signal net N_i is defined as the ratio of the number of aggressors for N_i to the total number of signal nets. During the global routing stage, however, two logically sensitive nets are considered to be *physically sensitive* to each other only if they are routed within the same routing region R_t. Therefore, the *physical sensitivity rate* r_{it} of net segment N_{it} in region R_t is defined as the ratio of the number of aggressors in R_t for N_{it} to the total number of net segments in R_t. Similar to [6], we assume that the logic sensitivity information between nets is known and will not change during the shield insertion and net ordering procedure.

Because the capacitive coupling between two sensitive nets decreases dramatically as the distance between them increases, we assume that the capacitive coupling exists between two sensitive net segments if and only if they are physically adjacent to each other (possibly with empty space between them – but without intervening shielding wires). The same capacitive coupling model has been employed in [6] and [7].

According to [20], the inductive coupling coefficient between two net segments N_{it} and N_{jt} in a region R_t is defined as

$$K_{it,jt} = \frac{M_{it,jt}}{\sqrt{L_{it} \cdot L_{jt}}} \tag{1}$$

where $M_{it,jt}$ is the mutual inductance between N_{it} and N_{jt}, and L_{it} and L_{jt} are self inductance for N_{it} and N_{jt} under the loop inductance model, respectively. $K_{it,jt}$ $(= K_{jt,it})$ should be between 0 and 1. It is proposed in [9] that the inductive coupling effect between N_{it} and N_{jt} can be characterized by the inductive coupling coefficient between them. Moreover, a formula-based K model has been developed in [9] to compute $K_{it,jt}$. The computation proceeds as follows. When N_{it} and N_{jt} are in different blocks, $K_{it,jt} = 0$ or a small constant. When the two net segments are in the same block, we consider the following two special cases first:

- when $n_{it} = n_{jt}$, the mutual inductance is reduced to self inductance and $K_{it,it} = 1$ by definition;
- when n_{it} (or n_{jt}) becomes s_{lt} (or s_{rt}), $K_{it,jt} = 0$ because it is now defined for two segments of a same current loop and should be 0 under the loop inductance model.

For other general cases, $K_{it,jt}$ $(= K_{jt,it})$ can be approximated by the mean of two linear interpolations of the above two special cases. Therefore, we have

$$K_{it,jt} = \frac{(f(i,t) + g(j,t))}{2} \tag{2}$$

where $f(i,t) = (n_{it} - s_{lt})/(n_{jt} - s_{lt})$ and $g(j,t) = (s_{rt} - n_{jt})/(s_{rt} - n_{it})$ are two linear interpolations of 0 and 1 as shown in Fig. 2.

According to [9], the K model is reasonably accurate – within $+20\%$ to -10% error range compared to the three-dimensional (3-D) field solver [21] – and tends to be conservative. Furthermore, an effective K model (or in short, K_{eff} model) is proposed in [9] to use the weighted sum of coupling coefficients (K_{it}) as

[1]This assumption is compatible with the K_{eff} model to be presented in Section II-C.

[2]However, our problem formulation and algorithm do not depend on this mutually (or symmetrically) sensitive assumption. In fact, we can consider the nonsymmetrically sensitive cases without changing our algorithm, as the sensitive information between nets is the input to our algorithms.

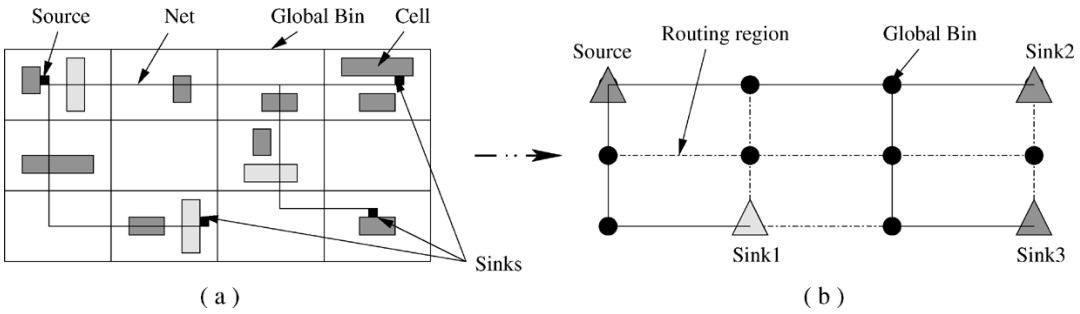

Fig. 1. (a) Layout. (b) The corresponding routing graph.

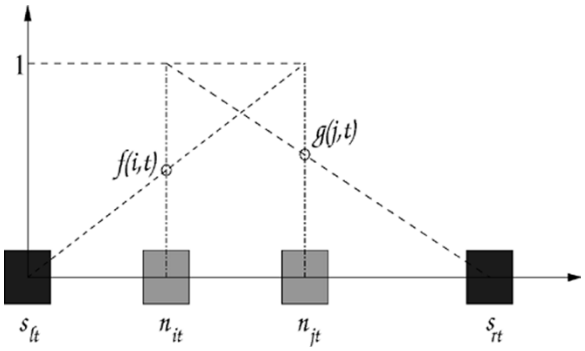

Fig. 2. Illustration of $K_{it,jt}$ computation in a given block.

a figure of merit for the total amount of inductive noise induced on the net segment N_{it}. The K_{it} can be calculated by

$$K_{it} = \sum_{j \neq i} S_{ij} \cdot K_{it,jt} \tag{3}$$

where $S_{ij} = 1$ for all net segments N_{jt} that are sensitive to N_{it}, otherwise $S_{ij} = 0$.

Given a set of parallel nets with uniform wire lengths, the simultaneous SINO problem finds a minimum area track assignment solution for both nets and shields such that all nets are capacitive crosstalk free (i.e., no physically sensitive nets are adjacent[3] to each other) and have inductive crosstalk less than the given bounds [9]. For example, Fig. 3(a) shows an initial routing solution for five parallel nets in a routing region, and (b) shows the final solution after SINO. According to Fig. 3(b), no two sensitive nets are adjacent to each other and no two sensitive nets in this example are within the same block (the leftmost and rightmost P/G networks are not drawn). A SINO solution can be represented compactly as an ordered net sequence. For example, the SINO solution as shown in Fig. 3(b) can be represented as $e = \langle 3, 4, 1, S, 5, 2 \rangle$.

SINO problem has been studied under K_{eff} model in [9]. It has been shown that the K_{eff} model has a high fidelity versus SPICE calculated coupling noise for a SINO solution with a fixed wire length. That is, a signal net in a SINO solution with a higher K value given by the K_{eff} model also has a higher SPICE-computed coupling noise under the distributed *RLC* circuit model. Such fidelity holds under the assumption that no sensitive nets are adjacent to each other in a SINO solution. The K_{eff} model is computationally simple and convenient to use in early design stages.

[3]Two nets are adjacent to each other if there is no net or shield between them.

C. Full-Chip Crosstalk Model: LSK Model

In order to cope with the *RLC* crosstalk constraints at the full-chip level, we treat the set of parallel net segments in each routing region as the input of SINO, hence we can leverage the SINO technique to reduce crosstalk at each routing region. However, at the full-chip level, we often care about the final *RLC* crosstalk induced at each sink while SINO only considers the *RLC* crosstalk within a routing region. Therefore, [14] proposed to use a length-scaled K_{eff} (LSK) model to capture the long range *RLC* crosstalk at the full-chip level effectively.

According to [14], if each routing region assumes a SINO solution, i.e, no capacitive coupling noise exists between any two physically sensitive nets and the inductive coupling noise is less than the given bound via K_{eff} model, then the full-chip *RLC* crosstalk measured at each sink can be controlled by constraining the LSK value (measured at each sink) less than a given bound $\overline{LSK}$. The *LSK* value for a net N_i at its j^{th} sink is defined as

$$LSK_{ij} = \sum_{t \in H_{ij}} l_t \cdot K_{it} \tag{4}$$

where l_t is the length of R_t and K_{it} is the total coupling for N_{it}. It has been validated in [14] that LSK model has high fidelity versus the SPICE computed worst case coupling noise by using the algorithm from [22]. That is, a signal net that has a higher LSK value at a sink also has a higher worst case coupling noise. Because of its simplicity and high fidelity, the LSK model will be employed in this paper to model the full-chip level *RLC* crosstalk for routing. Moreover, as the worst case coupling noise is considered, there is no need to consider the coupling noises under different input scenarios. For the detailed development of the *LSK* model, please refer to [14].

D. Problem Formulation

In the conventional design flow, routing is generally done via two phases: global routing and detailed routing. In global routing, a sequence of regions that each net will go through is decided first; in detailed routing, the actual routes in routing regions are then determined. This design flow makes it very difficult to consider *RLC* crosstalk constraints at the global routing stage, because both capacitive and inductive crosstalk depend on net segments' relative positions in each region, which are not known during global routing. Therefore, a new global routing algorithm that can address *RLC* crosstalk constraints at the full-chip level is in demand.

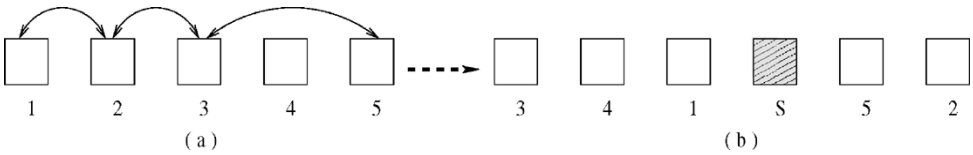

Fig. 3. Illustration of simultaneous shield insertion and net ordering, where arrows above nets indicate that the pointed nets are physically sensitive to each other. (a) Initial solution. (b) Final solution after SINO.

Moreover, integrating SINO [9] into the existing design flow to cope with the full-chip *RLC* crosstalk constraints may ruin the "optimal" solution obtained from global routing, because SINO as a postrouting procedure introduces shields into regions. The added shields would change the *region routing congestion*, defined as the total number of tracks occupied by nets, shields, and obstacles in that region. When the congestion of a region is greater than its capacity, *overflow* occurs in the region. The *overall chip routing congestion* is defined as the number of regions with overflow, because the more regions with overflow, the more difficult it is for a detailed router to find a final feasible routing solution [23]. Therefore, minimizing the total number of overflow regions is of paramount importance at the global routing stage, and it has been studied in [15] and [18] for global routing. We formulate our extended global routing problem as follows:

*Formulation 1—(***GSINO problem***):* Given a placement solution, a netlist and *RLC* crosstalk bounds specified at sinks, the GSINO problem decides a Steiner tree for each net and finds a SINO solution within each routing region such that the *RLC* crosstalk at all sinks is less than the given bounds[4], and the overall chip routing congestion is minimized.

Formally, a GSINO solution is an edge labeling in $G(V, E)$ in the form of $(e) = (e_1, e_2, \ldots, e_t)$, where each e_i is an ordered net sequence to represent a SINO solution in edge e's i^{th} layer, and all edges containing net N_i form a Steiner tree in $G(V, E)$. We consider *RLC* crosstalk constraints under the *LSK* model in this paper.

III. Multiphase GSINO Algorithm

The GSINO problem has high complexity, and its subproblems as Steiner tree generation and SINO are both NP-hard [9]. Therefore, we propose the following three-phase heuristic algorithm. In Phase I, we carry out crosstalk budgeting to distribute the crosstalk bound specified at sinks into potential routing regions for each net, and synthesize a global routing solution with shield area reservation and minimization. In Phase II, we perform SINO inside each routing region. In Phase III, we perform a greedy postrouting local refinement procedure to further reduce the routing congestion. We present the details of each phase algorithm below.

A. Phase I Algorithm

Phase I comprises of crosstalk budgeting and global routing synthesis. In the following, we first introduce a simple yet accurate formula to estimate the number of shields needed by the min-area SINO solution without actually carrying out the SINO algorithm. We then discuss a uniform crosstalk budgeting scheme that distributes the given *RLC* crosstalk bound specified at sinks to net segments in each routing region. Finally, a global routing technique based on the enhanced iterative deletion algorithm with shielding area reservation and minimization is presented.

1) Shield Estimation: According to [9] and [14], the number of shields needed for a min-area SINO solution depends on both the sensitivity rates (r_{it}) and the noise bounds ($\overline{K_{it}}$ in K_{eff} model) of all net segments in a routing region. A robust closed formula has been developed in [14] to estimate the number of shields needed for a min-area SINO solution. The formula is

$$|S_t| = a_1 \cdot \sum_{N_{it} \in G_t} \overline{K_{it}} \cdot r_{it} + a_2 \cdot \sum_{N_{it} \in G_t} r_{it} \tag{5}$$

where $a_1 = -0.104\,91$ and $a_2 = 0.493\,92$ are two constant coefficients. According to [14], (5) has an estimation accuracy of 87% over a large range of design parameters. As a_1 is negative, increasing the crosstalk bound reduces the number of shields needed for a min-area SINO solution.

Note that the number of net segments $|G_t|$ and physical sensitivity rates r_{it}s are fixed in a region R_t for the given global routing solution, hence (5) can be further simplified as a linear combination of the given noise bounds $\overline{K_{it}}$

$$|S_t| = \sum_{N_{it} \in G_t} \alpha_{it} \cdot \overline{K_{it}} + \beta_t \tag{6}$$

where $\alpha_{it} = a_1 \cdot r_{it}$ and $\beta_t = a_2 \cdot \sum_{N_{it} \in G_t} r_{it}$ according to (5). Because a_1 is negative, so are all coefficients (α_{it}s) of $\overline{K_{it}}$s.

2) Uniform Crosstalk Budgeting: To minimize the adverse effect on congestion due to shield insertion, we need to estimate the number of shields and hence properly reserve the right amount of tracks for shielding in the global routing stage. However, to use the estimation formula (5), we need to decide $\overline{K_{it}}$ for all net segments in each routing region first. Such a problem is called a crosstalk budgeting problem and has been solved optimally in [14] based on linear programming (LP). However, [14] assumes that the global routing solution is given *a priori*, and in this paper the global routing solution is unknown and to be decided. Therefore, the LP-based budgeting technique in [14] is not applicable. We employ a simple yet efficient uniform budgeting (UD) scheme as suggested in [14] to obtain the $\overline{K_{it}}$s and we describe it as follows.

For each single-source-multi-sink (SSMS) net N_i, an undirected sub-graph $G_i(V_i, E_i)$ is induced from $G(V, E)$, where V_i includes all *admissible* nodes that the final routing of N_i might possibly connect to. We also call graph $G_i(V_i, E_i)$ as a *complete net connection graph*, because it includes all possible net segments that may be used to route net N_i and hence completely

[4]Note that the study in [7] only minimizes the capacitive crosstalk without considering the capacity constraints.

ID/S Algorithm
1. **For** each SSMS net N_i
2. Construct $G_i(V_i, E_i)$ and initialize all net segments in G_i as non-critical
3. Let Ω be the set of G_i's
4. **Do** {
5. Choose a region R_t with the highest congestion in Ω
6. Choose an arbitrary non-critical net segment N_{it} from R_t
7. Remove N_{it} from the corresponding G_i
8. Update criticality of the remaining net segments in G_i
9. Update R_t's congestion with consideration of both shields and signal nets
10. }**Until**(Ω becomes a net connection forest)

Fig. 4. Phase I algorithm: Global routing synthesis based on iterative deletion. When shields are considered in line 9, it is ID/S, otherwise, it is ID.

determines the search space for net N_is final routing solution. In other words, each edge $e_t \subset E_i$ represents a potential net segment N_{it} in routing region R_t. In the following, we do not distinguish an edge e_{it} from a net segment N_{it} whenever there is no ambiguity. When we choose the admissible nodes for N_i, we consider only those nodes that are within the bounding box of N_i. Nevertheless, if necessary, we can easily expand the admissible nodes outside of the bounding box to explore more routing options. We use the Manhattan distance (half bounding box) between the source s_{i0} and a specific sink s_{ij} of signal net N_i to approximate their final wire length of a routing solution[5], which is denoted as l_{ij}. If the *LSK* crosstalk bound at sink s_{ij} is denoted as $\overline{LSK_{ij}}$, the total inductive coupling bounds for all net segments N_{it} in $G_i(V_i, E_i)$ are determined by:

$$\overline{K_{it}} = \frac{\overline{LSK_{ij}}}{l_{ij}}. \tag{7}$$

If a net segment is shared by multiple paths from the same source to multiple sinks, the minimum $\overline{K_{it}}$ bound determined from all individual paths is chosen.

With the *RLC* crosstalk bounds, we can compute the congestion overflow for routing region R_t in terms of the number of tracks as follows:

$$\begin{aligned} Ov_t &= |S_t| + |G_t| + O_t - C_t \\ &= \left(\sum_{N_{it} \in G_t} \alpha_{it} \cdot \overline{K_{it}} + \beta_t \right) + |G_t| + O_t - C_t \end{aligned} \tag{8}$$

where $|S_t|$ is the estimated number of shields for SINO, $|G_t|$ is the number of net segments in R_t, O_t is the number of tracks preoccupied by obstacles in R_t, and C_t is the capacity of R_t. Equivalently, (8) computes the differences between the required track numbers (the sum of estimated shielding and net segments) and the available free tracks (the difference between the capacity and the preoccupied obstacles). If Ov_t is greater than 1, overflow occurs in R_t. Otherwise, there is no overflow in R_t.

3) Iterative Deletion With Shielding Estimation Algorithm: Existing global routing techniques can be classified as net-order dependent techniques and net-order independent techniques. The net-order dependent routing techniques route each net sequentially and at each step a decision is made based on a greedy cost function. Examples include maze routing [24] and the line-probe algorithm [25]. The net-order independent routing techniques try to consider the routing of all nets simultaneously at each step. Examples include the iterative deletion (ID) algorithm [26], negotiation-based routing [27], [28], and the multicommodity network flow based approach [17], [29].

The ID algorithm was developed in [26] to route nets in standard cell designs for congestion minimization, and the congestion effect due to shielding was not considered yet. In this paper, we will enhance the ID algorithm to minimize congestion with shielding estimation and minimization for OTC routing. We choose the ID algorithm because it is able to consider all signal nets simultaneously. It is less efficient but may lead to better solutions compared to other order-dependent global routing approaches [26]. We denote our enhanced ID algorithm with shield estimation as ID/S. The primary difference between ID/S and ID is that during the calculation of congestion, ID/S considers both shields and signal nets, while ID considers only signal nets. We present the ID/S algorithm in Fig. 4.

A route for a SSMS net N_i is a set of connected edges (or net segments) in routing graph $G_i(V_i, E_i)$ such that there exists one and only one path from the source pin to any one of its sink pins. In other words, the set of connected edges forms a Steiner tree on $G_i(V_i, E_i)$, and the source pin is the root while sink pins are the leaves. G_i is said to be *admissible* if there exists at least one routing Steiner tree in G_i that connects the source to all sinks. A net segment N_{it} is *noncritical* if G_i remains admissible after removing N_{it} from G_i. Otherwise, N_{it} is *critical*.

The ID/S algorithm proceeds as follows. First we construct the complete net connection graphs for all nets, the set of which is denoted as Ω, and initialize all net segments in Ω as noncritical. We then iteratively identify the region in Ω that is most congested as defined by (8), and randomly choose one noncritical net segment in that region to delete. After each deletion, we update the criticality of all affected net segments and re-compute the congestion of that region. Once a noncritical net segment becomes critical, it is "frozen" and can not become noncritical. The iteration stops when the remaining net segments in G_i become critical and form a routing connection tree. In other words, Ω reduces to a routing connection forest. In essence, we minimize the overall chip routing congestion by avoiding the routing of nets in the most congested areas whenever possible. Moreover, because we reserve the right number of free-tracks for shield insertion by utilizing (8), our congestion minimization procedure also takes shield insertion into consideration. Nevertheless, our shield estimation can be also used to enhance the congestion estimation as developed in [15] and [30].

The output of Phase I algorithm is an **un-ordered** labeling of edges in $G(V, E)$ with shield reservation. Fig. 5(a) shows

[5]More accurate routing length estimation techniques can be incorporated into our algorithm framework without difficulty.

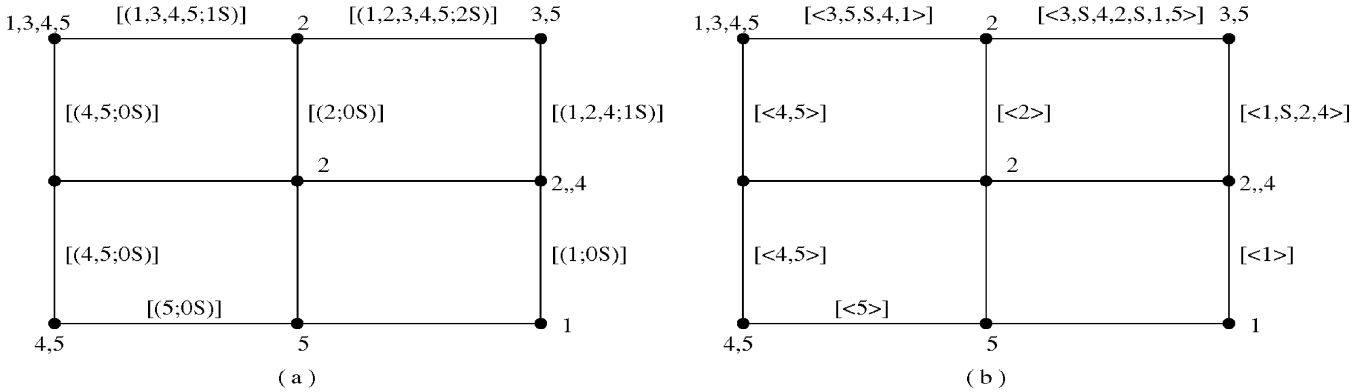

Fig. 5. Labels of each vertex are pins contained in that vertex. Labels of each edge are: (a) output of Phase I and (b) output of Phase II or III.

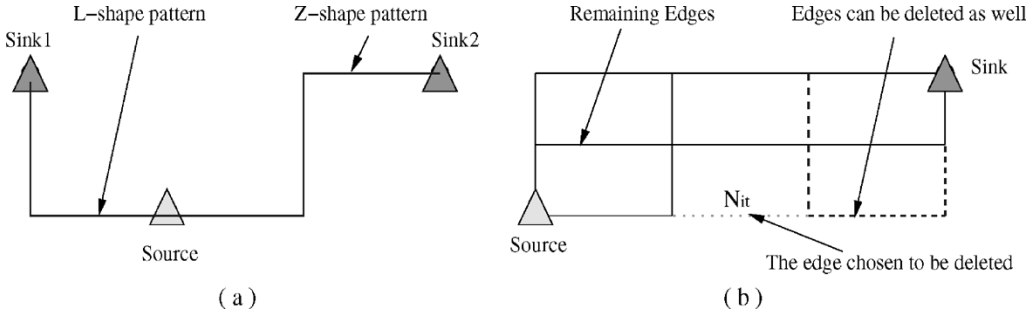

Fig. 6. (a) Illustration of pattern routing. (b) Illustration of the speed-up deletion process for pattern routing.

such an example, where each region has only one layer. We also record the estimated shield numbers required for SINO in the labeling of edges. For example, [(1,2,3,4,5;2S)] means that the region contains net segments for N_1, N_2, N_3, N_4 and N_5 with two free-tracks reserved for shields.

4) Speedup ID/S Algorithm: According to Fig. 4, iterations in the "Do-Until" loop account for the major runtime complexity of the ID/S algorithm. In the worst case, the number of potential net segments in $G_i(V_i, E_i)$ for each SSMS net after initialization equals to HV, where H and V are the number of horizontal and vertical routing regions, respectively. If at each iteration only one net segment in the most congested region is deleted and we assume the updating of criticality and congestion can be done in constant time, the ID/S algorithm has time complexity of $O(HVP)$, where P is the total number of SSMS nets. However, deleting one net segment at a time will make the algorithm too slow to apply to large circuits as pointed out by [26] for the original ID algorithm. [26] speeds up their ID algorithm by building a *simplified net connection graph* with less edges than the complete net connection graph, which is made possible by their standard cell routing styles. Because in standard cell routing, direct connection is only allowed for pins within the routing channel or the same cell/feedthrough. In contrast, for OTC routing style as studied in this paper, direct connection between any pair of pins is possible; hence the technique from [26] cannot be employed. Moreover, deleting net segments according to congestion pictures alone may end up producing a routing solution with too many bends along the routing paths. Every bend may introduce a via to connect signals in different routing layers at the final physical layout. Vias not only cause difficulty for detailed routing, but also deteriorate signal integrity and circuit reliability [5]. Therefore, to speed up our ID/S routing algorithm and reduce the number of vias, we employ the pattern routing technique proposed in [16]. The idea of pattern routing is to connect two pin nets by predefined patterns. Simple patterns like L-shaped (one bend) or Z-shaped (two bends) patterns are usually preferred. Fig. 6(a) shows an L-shaped routing pattern and a Z-shaped routing pattern for a net with two sink pins. In this case, each net can have at most $(H+V)$ candidate patterns in its corresponding routing graph $G_i(V_i, E_i)$. Therefore, the ID/S algorithm's complexity can be reduce to $O((H+V)P)$ with pattern routing.

Another advantage of pattern routing is that we may further speed up the ID/S algorithm by pruning out several noncritical net segments at each iteration. For the example as shown in Fig. 6(b), deleting a net segment (dotted line) renders other net segments (dashed line) in G_i noncritical and hence those net segments can be simply deleted.

B. Phase II Algorithm

Phase II algorithm performs SINO within each routing region such that there is no *RLC* crosstalk violation at each sink. The detailed algorithm for Phase II is shown in Fig. 7.

With respect to the budgeted $\overline{K_{it}}$ for all net segments from Phase I, we synthesize a SINO solution in each routing region by using the algorithm developed in [9]. We compute the LSK value at each sink according to (4). Let *LSK slack* of a sink be the gap between $\overline{LSK}$ and *LSK* at the sink. The crosstalk violation at each sink is indicated by a negative LSK slack value. The SINO algorithm from [9] is based on simulated annealing, and the crosstalk and area constraints are implemented as two components of the cost function. Therefore, a very limited number

Phase II algorithm
For(each routing region R_t)
    Perform SA-based SINO;
Compute LSK slack values at sinks;
While(there has crosstalk violation)
    Find net N_i whose j^{th} sink p_{ij} has most severe crosstalk violation;
    Find the region R_t containing N_{it} with the highest K_{it};
    Insert a shield into R_t;
    Perform simultaneous ordering of shields and net segments;
    Update LSK slacks for all affected paths;

Fig. 7. Phase II algorithm.

Local refinement algorithm
While(removing shields is possible){
    Find net N_i whose j^{th} sink p_{ij} has the largest *LSK slack*;
    Find the region R_t containing N_{it} with the least K_{it};
    Remove a shield from R_t;
    Perform simultaneous ordering of shields and net segments;
    If(no violation is found){
        Accept the new solution for R_t;
        Update *LSK slacks* for all affected paths;
    }**else**
        Restore the old solution for R_t;
}

Fig. 8. Phase III local refinement algorithm.

of crosstalk violations may exist after SINO. To implement a "better" SINO algorithm such that all net segments satisfy the partitioned crosstalk bounds within *each* and *every* region may lead to over-design with more shields than needed. Hence we eliminate the remaining crosstalk violations through the following procedures. We first find a net N_i with the most negative LSK slack (i.e., the most severe crosstalk violation) at sink p_{ij}, and locate a routing region R_t with the highest K_{it} for net segment N_{it}. We then insert exactly *one* more shield into R_t and carry out simultaneous ordering of both shields and net segments to obtain the minimum crosstalk for N_{it} but still satisfy crosstalk bounds for all other net segments in R_t. Such a net ordering is implemented as a simpler case of the SINO algorithm [9] without shield insertion or deletion during the simulated annealing process. Inserting a shield in R_t may reduce K_{it} as well as K_{jt} for other segments N_{jt} in R_t, so we need to update the LSK slacks for all nets passing R_t. The iteration stops when there is no crosstalk violation for any net.

C. Phase III Algorithm

Phase III reduces the total shield number by exploiting the remaining LSK slacks to remove the un-needed shields in a greedy fashion. We first find a route that has the largest LSK slack at sink p_{ij} for net N_i, then we find a region R_t with the least K_{it} value for net segment N_{it}. Exactly one shield will be removed from R_t and then simultaneous ordering of both signals and net segments is performed to obtain a solution with the minimum sum of K values for all net segments in R_t. Because removing a shield may increase K values for certain net segments in R_t, we must check if these increments can be compensated by their LSK slacks respectively. If the answer is yes, then no crosstalk violation occurs, and we accept the new solution for R_t and update the affected LSK slacks for all nets passing R_t. Otherwise, we restore our original solution for R_t. The iteration stops when removing shields is no longer possible in any region. The detailed algorithm for Phase III is presented in Fig. 8.

TABLE II
CHARACTERISTICS OF BENCHMARKS

Circuits	# of nets	# of regions	# of pins	Region capacity
prim1.1	1266	8 × 16	2738	H:14 V:18
prim1.2	1266	16 × 16	2738	H:12 V:12
prim1.3	1266	128 × 16	2738	H:12 V:3
prim2.1	3817	8 × 16	10971	H:32 V:42
prim2.2	3817	32 × 32	10971	H:18 V:14
prim2.3	3817	128 × 28	10971	H:16 V:8
ibm01	13056	64 × 64	44266	H:20 V:14
ibm03	26104	80 × 64	75710	H:38 V:28
ibm04	31328	96 × 64	89591	H:30 V:30

The output of either Phase II or Phase III is an ordered labeling of edges with shield insertion in $G(V, E)$. As illustrated in Fig. 5(b), the label $[\langle 3, S, 4, 2, S, 1, 5\rangle]$ of an edge means that the final net order from left to right in that edge are: N_3, *Shield*, N_4, N_2, *Shield*, N_1, N_5.

IV. EXPERIMENTS

A. Experiment Settings

We have implemented the three-phase GSINO algorithm in C++/STL on a Pentium® (833 *MHz*) machine with 256 M memory running RedHat operating systems. For simplicity of presentation, we assume in our experiments that an overall logical sensitivity rate among all nets is 50%, and $\overline{LSK}$ is 1000 for all sinks. The placement results are generated by DRAGON [31]. We use six MCNC benchmarks to validate our GSINO algorithm and three large benchmarks from the ISPD'98/IBM circuit benchmark suite [32] to demonstrate the applicability of the GSINO algorithm to large industrial designs. Same as in [13], we assume two preferred routing directions for all regions, one for horizontal wires and the other for vertical wires. Because there is no physical geometry information in the original benchmark, we derive the routing capacity by the following procedures: we route nets via the minimum spanning tree (MST) algorithm sequentially without considering congestion. The obtained average congestion among all horizontal (or vertical) regions is used as the capacity for horizontal (or vertical) regions. This introduces a challenging routing condition even without considering shield insertion. The characteristics of the benchmarks derived are shown in Table II.

B. Baseline Case for Comparison

There is no existing work that can consider *RLC* crosstalk constraints at the global routing stage. In order to show the effectiveness of our GSINO algorithm, we implemented an alternative algorithm denoted as GR+SINO. The GR+SINO algorithm also has three phases. Compared to our GSINO algorithm, GR+SINO only differs in Phase I, where the original ID algorithm without shield reservation and minimization is employed. GR+SINO is the best alternative that we can think of and will be used as our baseline case for comparison.

C. Experiment Results

1) Crosstalk Violation: Table III reports the maximum/average *LSK* values among all sinks after Phase II and Phase III. According to Table III, the maximum *LSK* values after Phase II

TABLE III
MAXIMUM/AVERAGE LSK VALUES AFTER PHASE II AND III, WHERE $\overline{LSK}$ IS 1000

Test Circuits		Phase II max/avg	Phase III max/avg
prim1.1	GR+SINO	977.1/138.5	997.4/154.2
	GSINO	999.0/137.1	999.0/150.2
prim1.2	GR+SINO	985.4/148.2	993.8/176.8
	GSINO	990.2/158.4	995.2/186.9
prim1.3	GR+SINO	998.7/175.0	999.3/210.1
	GSINO	990.2/171.1	999.9/210.4
prim2.1	GR+SINO	958.2/95.6	995.1/103.6
	GSINO	964.3/94.8	998.9/105.9
prim2.2	GR+SINO	998.5/154.8	1000.0/198.9
	GSINO	989.0/154.3	999.5/194.8
prim2.3	GR+SINO	997.9/168.3	1000.0/213.9
	GSINO	998.7/169.7	1000.0/216.1

are all smaller than the given bound $\overline{LSK}$, i.e., Phase II algorithm can completely eliminate crosstalk violations. Phase III algorithm is an extra refinement procedure to further improve the overall routing quality by leveraging the *LSK* slacks. Phase III should not introduce any crosstalk violation, and this is indeed true as shown in Table III: the average and maximum *LSK* values after Phase III increase while the maximum *LSK* values still meet the given bound $\overline{LSK}$.

2) Comparisons Between GSINO and GR+SINO: As pattern routing [16] is used in ID/S, there is no routing detour after global routing. Hence both GR+SINO and GSINO have similar wire-length in the sense of global routing. Therefore, we only present results on routing congestion minimization in the following. We measure the overall chip routing congestion by the total number of overflow net segments and the total number of regions with overflows. In Table IV, we report the number of shields, the number of overflow segments ($OvSeg = \sum_{\forall t} Ov_t$) and the number of overflow routing regions ($OvReg = \sum_{\forall t} sign(Ov_t)$) for the six MCNC benchmarks under GR+SINO and GSINO, respectively. Results after each and every phase of the algorithm are reported. The shield numbers from Phase I are based upon our shield estimation formula (6) and are expressed in continuous value. The value in parenthesis is the reduction of GSINO over GR+SINO in percentage after each phase of the algorithm. According to Table IV, GSINO always reduces both the total number of overflow segments and the total number of overflow regions when compared to GR+SINO. For example, for benchmark *prim2.1*, the total number of overflow segments is reduced by 81.0% and 82.4% after Phases II and III, respectively. For the same benchmark, the total number of overflow routing regions is reduced by 72.4.1% and 74.1% after Phases II and III, respectively. Note that GSINO is designed to reduce congestion in most congested regions by routing signal nets and shields to less congested regions, hence GSINO may use more shields than GR+SINO but still achieves less overall congestion (see *prim2.2* for an example).

Phase III effectively reduces the total number of shields without violating the given $\overline{LSK}$. For example, for *prim2.2* benchmark, Phase III reduces shields from 2187 to 1839 for GR+SINO and from 2220 to 1892 for GSINO. The relative reductions are 15.9% and 14.8%, respectively. As a byproduct, Phase III algorithm also reduces the number of overflow segments and overflow regions. For the same benchmark *prim2.2*, when compared with Phase II, Phase III can reduce the number of overflow segments from 326 to 266 for GR+SINO and from 267 to 214 for GSINO, and the relative reduction is 18.4% and 19.9%, respectively (see comparison between column 7 and 10 in Table IV); and can reduce the number of overflow regions from 132 to 114 for GR+SINO and from 116 to 98 for GSINO, and the relative reduction is 13.6% and 15.5%, respectively, (see comparison between column 8 and 11 in Table IV).

In order to examine the impact of shield insertion on routing congestion, we further compare the global routing without shield insertion (denoted as GR) with GR+SINO and GSINO in terms of the number of overflow segments and the number of overflow routing regions in Table V. GR's results are obtained by running the ID algorithms only and provide a lower bound on routing congestion for both GR+SINO and GSINO. According to Table V, shield insertion in GR+SINO can severely increase the routing congestion. For example, for *prim*1.3 circuit, GR+SINO increases the number of overflow segments by 117.4% and the number of overflow regions by 71.9% when compared to GR. In contrast, shield insertion in GSINO has only a moderate impact on routing congestion. For example, for the same *prim*1.3 circuit, GSINO only increases the number of overflow segments by 5.2% and the number of overflow regions by 19.1% when compared to GR. Therefore, the proposed GSINO algorithm with shield estimation and minimization is able to avoid those congested routing regions while routing signal nets, thus keep the overhead due to shield insertion to the minimum.

3) Runtime: We report the runtime for each algorithm phase as well as the total runtime in Table VI. Note that the runtime of Phase I includes the time to compute the coefficients in (5). As the coefficients are computed by a closed formula, their runtime is almost negligible compared to the routing algorithm's. It is interesting to note that although GSINO achieves better routing quality than GR+SINO, it almost always consumes less runtime for all phases when compared to GR+SINO. This convincingly tells us that GSINO with a good shielding estimation and reservation in the early routing design stage can indeed not only improve the overall routing quality, but also help to achieve faster design closure. We also observe that Phase III almost always consumes more runtime than Phase II. For example, for *prim2.2*, Phase III consumes 31% more time than Phase II for GSINO, and accounts for about 55% of the total runtime. Moreover, Phase III can be turned off because the congestion results after Phase II of GSINO are already smaller than those obtained from the completed run of GR+SINO.

4) More Experiment Results: Having validated the effectiveness of our GSINO algorithm, we further explore the applicability of the GSINO algorithm to three IBM designs from ISPD'98/IBM circuit benchmark suite [32]. Phase III is turned off for these benchmarks.

In Table VII, the total number of shields, overflow segments and overflow regions are reported for the three IBM benchmarks. According to Table VII, GR+SINO and GSINO use almost the same number of shields, but GSINO is superior to GR+SINO by having less number of overflow segments

TABLE IV
COMPARISON OF OVERFLOW BETWEEN GR+SINO AND GSINO

1	2	3	4	5	6	7	8	9	10	11
Test Circuit		Phase I			Phase II			Phase III		
		Shield	OvSeg	OvReg	Shield	OvSeg	OvReg	Shield	OvSeg	OvReg
prim1.1	GR+SINO	336.5	98	36.0	200	63	22	2	178	57
	GSINO	339.6	51.3 (-47.7%)	28 (-22.2%)	226	29 (-54.0%)	17 (77.3%)	194	25 (-56.1%)	16 (-27.3%)
prim1.2	GR+SINO	517.6	171.5	84	337	132	56	292	118	53
	GSINO	507.6	101.6 (-40.8%)	53 (-36.9%)	341	72 (-45.5%)	28 (-50.0%)	293	62 (-47.5%)	25 (-52.8%)
prim1.3	GR+SINO	2345.5	982.0	417	1793	872	284	1587	787	251
	GSINO	2381.8	483.6 (-50.8%)	308 (-26.1%)	1958	470 (-46.1%)	215 (-24.3%)	1707	381 (-51.6%)	174 (-30.7%)
prim2.1	GR+SINO	666.6	192.5	42	479	137	29	410	119	27
	GSINO	645.5	49.2 (-74.4%)	16 (-61.9%)	460	26 (-81.0%)	8 (-72.4%)	394	21 (-82.4%)	7 (-74.1%)
prim2.2	GR+SINO	3108.8	503.4	227	2187	326	132	1839	266	114
	GSINO	3078.6	428.8 (-14.8%)	215 (-5.3%)	2220	267 (-18.1%)	116 (-12.1%)	1892	214 (-19.5%)	98 (-14.0%)
prim2.3	GR+SINO	7952.1	2362.8	787	6220	1962	557	5381	1781	502
	GSINO	7810.9	1647.1 (-30.3%)	630 (-19.9%)	6190	1307 (-33.4%)	449 (-19.4%)	5355	1162 (-34.8%)	399 (-20.5%)

TABLE V
COMPARISON OF SHIELD INSERTION OVERHEAD BETWEEN GR, GR+SINO AND GSINO

Test Circuit	GR		GR+SINO		GSINO	
	OvSeg	OvReg	OvSeg	OvReg	OvSeg	OvReg
prim1.1	22	12	178 (709.0%)	57 (375.0%)	25 (13.6%)	16 (33.3%)
prim1.2	36	20	118 (227.8%)	53 (165.0%)	62 (72.2%)	25 (25.0%)
prim1.3	362	146	787 (117.4%)	251 (71.9%)	381 (5.2%)	174 (19.2%)

TABLE VI
COMPARISON OF RUNTIME IN SECONDS BETWEEN GR+SINO AND GSINO

Test Circuit		Phase I	Phase II	Phase III
prim1.1	GR+SINO	7.7	1457.0	1719.9
	GSINO	8.4	1397.0	1614.2
prim1.2	GR+SINO	16.8	1623.7	1721.2
	GSINO	19.7	1601.7	1586.7
prim1.3	GR+SINO	124.5	6957.3	6611.2
	GSINO	118.8	6671.3	6014.4
prim2.1	GR+SINO	98.2	8700.0	18470.4
	GSINO	104.4	8194.2	17134.7
prim2.2	GR+SINO	930.4	12176.6	16344.0
	GSINO	889.2	11979.0	15740.7
prim2.3	GR+SINO	2095.7	26528.9	29616.0
	GSINO	1777.1	25597.1	27975.6

and overflow regions. For example, compared to GR+SINO, GSINO reduces the number of overflow segments by 13.0%, 12.5% and 20.8%, and reduces the number of overflow regions by 12.0%, 8.9%, and 18.4% for the three benchmarks, respectively. In Table VIII, we report the total runtime for the three benchmarks. Combining with Table VII, we find that GSINO achieves better routing quality with less runtime compared to GR+SINO. All the above observations are consistent with what we have observed from the six MCNC benchmarks.

V. DISCUSSION AND FUTURE WORK

RLC crosstalk will become increasingly critical as more system-on-chip designs operate at giga-hertz clocks in the near future. In order to meet the increasingly restrictive *RLC* crosstalk constraints, we have formulated an extended global routing (called GSINO) problem and developed an effective three-phase algorithm to solve it. The algorithm leverages simultaneous SINO to completely eliminate *RLC* crosstalk violations. Moreover, shield reservation and minimization based on prerouting shield estimation is applied for faster design closure. Experiment results from large industrial benchmarks have showed that, when compared to the best alternative method with postrouting shield insertion and net ordering, the proposed GSINO algorithm can reduce the overall routing congestion up to 18.4% with less runtime.

To synthesize a global routing solution, we employed the enhanced iterative deletion algorithm (ID/S) in this paper. Nevertheless, we recognize that because of the nature of iterative deletion, our routing algorithm is slower than other order dependent routing techniques like [13], [15]. However, as our whole GSINO framework is not limited to the ID/S algorithm, any global routing algorithm can be easily extended by incorporating estimated shields into congestion calculation. For example, we have applied the multilevel routing technique, a newly developed efficient routing technique [33], [34], to solve the power network and signal network codesign problem in [35]. Clearly, if power network design is not considered, the routing algorithm from [35] can also be used to replace the ID/S algorithm in this paper.

We employed the LSK model in this paper. This model is extremely efficient and has a high fidelity versus the SPICE computed worst case coupling noise [22]. Because of the high fidelity and its efficiency, the LSK model is suitable to model the *RLC* crosstalk during full-chip routing synthesis[6]. The LSK model is developed under the assumption that there is no crosstalk (coupling) between different routing regions separated by P/G wires. This assumption has been used in a number of works, including inductance modeling in [38]. Nevertheless, this assumption may not always be true. Therefore, we plan to develop a more accurate yet still efficient *RLC* crosstalk model in the future. Moreover, in this paper, we only synthesized an extended global routing with track assignment for both signal nets and shields, but not detailed routing. In the future, we

[6]Similarly, the Elmore delay model [36] has high fidelity but not accurate for measuring the interconnect delay. But due to its efficiency, the Elmore delay has been widely used for performance-driven routing, like [28], [37].

TABLE VII
COMPARISON OF OVERFLOW BETWEEN GR+SINO AND GSINO

Test Circuit		Phase I			Phase II		
		Shield	OvSeg	OvReg	Shield	OvSeg	OvReg
ibm01	GR+SINO	12578.1	4127	1072	8978	3217	784
	GSINO	12489.3	3611 (-12.5%)	988 (-7.8%)	9029	2798 (-13.0%)	690 (-12.0%)
ibm03	GR+SINO	27046.1	6632	1250	19721	4964	944
	GSINO	26774.8	5848 (-11.8%)	1155 (-7.6%)	19672	4345 (-12.5%)	860 (-8.9%)
ibm04	GR+SINO	28744.8	4280	1013	21159	3003	716
	GSINO	28481.3	3399 (-20.6%)	868 (-14.3%)	21123	2377 (-20.8%)	584 (-18.4%)

TABLE VIII
COMPARISON OF RUNTIME IN SECONDS BETWEEN GR+SINO AND GSINO

Circuits		Phase I	Phase II
ibm01	GR+SINO	5791.8	8267.6
	GSINO	5708.5	8072.6
ibm03	GR+SINO	15329.7	36300.9
	GSINO	15350.2	35660.3
ibm04	GR+SINO	26645.6	33906.2
	GSINO	28910.9	33348.4

will develop detailed routing algorithms with accurate parasitic extraction, and present experiment results on the comparison between the LSK model and the SPICE calculated coupling noise.

REFERENCES

[1] C. K. Cheng, L. Lillis, S. Lin, and N. Chang, *Interconnect Analysis and Synthesis*. New York: Wiley, 2000.

[2] D. A. Kirkpatrick and A. L. Sangiovanni-Vincentelli, "Techniques for crosstalk avoidance in the physical design of high-performance digital systems," in *Proc. Int. Conf. Computer-Aided Design*, Nov. 1994, pp. 616–619.

[3] K. Chaudhary, A. Onozawa, and E. S. Kuh, "A spacing algorithm for performance enhancement and cross-talk reduction," in *Proc. Int. Conf. Computer-Aided Design*, 1993, pp. 697–702.

[4] S. Thakur, K.-Y. Chao, and D. F. Wong, "An optimal layer assignment algorithm for minimizing crosstalk for three layer VHV channel routing," in *Proc. IEEE Int. Symp. Circuits Syst.*, 1995, pp. 207–210.

[5] C.-C. Chang and J. Cong, "Pseudo pin assignment with crosstalk noise control," in *Proc. Int. Symp. Phys. Design*, 2000, pp. 343–346.

[6] T. Xue and E. S. Kuh, "Post global routing crosstalk synthesis," *IEEE Trans. Computer-Aided Design Integr. Circuits Syst.*, vol. 16, no. 12, pp. 1418–1430, Dec. 1997.

[7] H. Zhou and D. F. Wong, "Global routing with crosstalk constraints," *IEEE Trans. Computer-Aided Design Integr. Circuits Syst.*, vol. 18, no. 11, pp. 1683–1688, Nov. 1999.

[8] L. He, N. Chang, S. Lin, and O. S. Nakagawa, "An efficient inductance modeling for on-chip interconnects," in *Proc. IEEE Int. Conf. Custom Integr. Circuits*, 1999, pp. 457–460.

[9] K. M. Lepak, M. Xu, J. Chen, and L. He, "Simultaneous shield insertion and net ordering for capacitive and inductive coupling minimization," *ACM Trans. Design Automation of Electronic Syst.*, vol. 9, no. 3, pp. 290–309, 2004.

[10] Y. Cao, C. M. Hu, X. Huang, A. B. Kahng, S. Muddu, D. Stroobandt, and D. Sylvester, "Effects of global interconnect optimizations on performance estimation of deep submicron design," in *Proc. Int. Conf. Computer-Aided Design*, 2000, pp. 56–61.

[11] G. Zhong, H. Wang, C.-K. Koh, and K. Roy, "A twisted bundle layout structure for minimizing inductive coupling noise," in *Proc. Int. Conf. Computer-Aided Design*, 2000, pp. 406–411.

[12] Y. Massoud, J. Kawa, D. MacMillen, and J. White, "Modeling and analysis of differential signaling for minimizing inductive cross-talk," in *Proc. Design Automation Conf.*, 2001, pp. 804–809.

[13] R. Kastner, E. Bozorgzadeh, and M. Sarrafzadeh, "An exact algorithm for coupling-free routing," in *Proc. Int. Symp. Phys. Design*, 2001, pp. 10–15.

[14] J. Xiong and L. He, "Full-chip routing optimization with RLC crosstalk budgeting," *IEEE Trans. Computer-Aided Design Integr. Circuits Syst.*, vol. 23, pp. 366–377, Mar. 2004.

[15] R. Hadsell and P. Madden, "Improved global routing through congestion estimation," in *Proc. Design Automation Conf.*, 2003, pp. 28–31.

[16] R. Kastner, E. Bozorgzadeh, and M. Sarrafzadeh, "Predictable routing," in *Proc. Int. Conf. Computer-Aided Design*, 2000, pp. 110–114.

[17] C. Albrecht, "Provably good global routing by a new approximation algorithm for multicommodity flow," in *Proc. Int. Symp. Phys. Design*, 2000, pp. 19–25.

[18] H. Su, J. Hu, S. Sapatnekar, and S. Nassif, "Congestion-driven codesin of power and signal networks," in *Proc. Design Automation Conf.*, 2002, pp. 64–69.

[19] P. Chen, Y. Kukimoto, C.-C. Teng, and K. Keutzer, "On convergence of switching windows computation in presence of crosstalk noise," in *Proc. Int. Symp. Phys. Design*, 2002, pp. 84–89.

[20] N. Lda, *Engineering Electromagnetics*: Springer-Verlag, 2000.

[21] M. Kamon, M. Tsuk, and J. White, "Fasthenry: A multipole-accelerated 3d inductance extraction program," *IEEE Trans. Microwave Theory Tech.*, vol. 42, no. 9, pp. 1750–1758, Sep. 1994.

[22] J. Chen and L. He, "Worst-case crosstalk noise for nonswitching victims in high-speed buses," IEEE Trans. Computer-Aided Design Integr. Circuits Syst., to be published.

[23] A. Rohe and U. Brenner, "An effective congestion driven placement framework," in *Proc. Int. Symp. Phys. Design*, 2002, pp. 6–11.

[24] J. Soukup, "Fast maze router," in *Proc. Design Automation Conf.*, 1978, pp. 100–102.

[25] D. Hightower, "A solution to line routing problems on the continuous plane," in *Proc. Design Automation Conf.*, 1969, pp. 1–24.

[26] J. Cong and B. Preas, "A new algorithm for standard cell global routing," in *Proc. Int. Conf. Computer-Aided Design*, Nov. 1988, pp. 176–179.

[27] R. Nair, "A simple yet effective technique for global wiring," *IEEE Trans. Computer-Aided Design Integr. Circuits Syst.*, vol. 6, no. 2, pp. 165–172, Jun. 1987.

[28] L. McMurchie and C. Ebeling, "Pathfinder: A negotiation-based performance-driven router for FPGAs," in *Int. Symp. Field Programmable Gate Arrays*, 1995, pp. 111–117.

[29] R. C. Carden and C.-K. Cheng, "A global router using an efficient approximate multicommodity multiterminal flow algorithm," in *Proc. Design Automation Conf.*, 1991, pp. 316–321.

[30] S. Lin and Y. Chang, "A novel framework for multilevel routing considering routability and performance," in *Proc. Int. Conf. Computer-Aided Design*, 2002, pp. 44–50.

[31] M. Wang, X. Yang, and M. Sarrafzadeh, "DRAGON2000: Standard-cell placement tool for large industry circuits," in *Proc. Int. Conf. Computer-Aided Design*, 2000, pp. 260–263.

[32] C. Alpert, "The proc. int. symp. on physical design98 circuit benchmark suite," in *Proc. Int. Symp. Physical Design*, 1998, pp. 80–85.

[33] J. Cong, M. Xie, and Y. Zhang, "An enhanced multilevel routing system," in *Proc. Int. Conf. Computer-Aided Design*, 2002, pp. 51–58.

[34] S.-P. Lin and Y.-W. Chang, "A novel framework for multilevel routing considering routability and performance," in *Proc. Int. Conf. Computer-Aided Design*, 2002, pp. 44–50.

[35] J. Xiong and L. He, "Full-chip multilevel routing for power and signal integrity," in *Proc. Design Automation and Test in Europe*, 2004, pp. 1116–1121.

[36] W. C. Elmore, "The transient response of damped linear networks with particular regard to wide-band amplifiers," *J. Appl. Phys.*, vol. 19, no. 1, pp. 55–63, Jan. 1948.

[37] T.-Y. Ho, Y.-W. Chang, S.-J. Chen, and D. T. Lee, "A fast crosstalk- and performance-driven multilevel routing system," in *Proc. Int. Conf. Computer-Aided Design*, 2003, pp. 382–387.

[38] K. Shepard and Z. Tian, "Return-limited inductances: A practical approach to on-chip inductance extraction," *IEEE Trans. Computer-Aided Design Integr. Circuits Syst.*, vol. 19, pp. 425–436, Apr. 2000.

Jinjun Xiong (S'04) received the B.E. (with honors) degree in precision measurement and control in 1998, the B.E. degree in industrial engineering in 1998, and the M.E. degree in microelectronic engineering in 2000, all from Tsinghua University, China. He received the M.S. degree in electrical and computer engineering from the University of Wisconsin at Madison, in 2001.

He is currently pursuing the Ph.D. degree from the University of California, Los Angeles (UCLA). He has been with the Mechanical and Industrial Engineering Department, University of Illinois at Urbana-Champaign, from August 2000 to August 2001. His current research interests include design automation for VLSI circuits and systems, large-scale optimization, and combinatorial mathematics.

Mr. Xiong received the Distinguished Graduate Fellowship from the University of Wisconsin at Madison in 2001, and from UCLA in 2002.

Lei He (S'94–M'99) received the B.S. degree in electrical engineering from Fudan University, China, in 1990 and the Ph.D. degree in computer science from the University of California, Los Angeles (UCLA) in 1999.

He is currently an Assistant Professor with the Electrical Engineering Department, UCLA. From 1999 to 2001, he was a faculty member with the University of Wisconsin, Madison. He held industrial positions with Cadence, Hewlett-Packard, Intel, and Synopsys. His research interests include computer-aided design of VLSI circuits and systems, interconnect modeling and design, programmable logic and interconnect, and power-efficient circuits and systems.

Dr. He received the Dimitris N. Chorafas Foundation Prize for Engineering and Technology in 1997, the Distinguished Ph.D. Award from the UCLA Henry Samueli School of Engineering and Applied Science in 2000, the NSF CAREER award in 2000, the UCLA Chancellor's Faculty Development Award in 2003, and the IBM Faculty Award in 2003.