Extended global routing with RLC
crosstalk constraints

J J. Xiong    L He
Extended global routing with RLC crosstalk constraints

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.
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...
a global bin, and each edge \( e \in E \) represents the routing area between two adjacent bins and has a length \( l_e \). An edge in the routing graph is also called a \( \textit{routing region} \) \( R_t \). To model the limited routing resources, we associate each edge in \( G(V, E) \) with a \( \textit{capacity} \) \( C_t \), 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 \( n_{i,t} \) and \( n_{j,t} \) are track ordering numbers for net segments \( N_{i,t} \) and \( N_{j,t} \), respectively, and \( s_{i,t} \) and \( s_{j,t} \) are track ordering numbers for the two edge shield segments.

### TABLE I

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( R_t )</td>
<td>routing regions in a chip</td>
</tr>
<tr>
<td>( l_e )</td>
<td>length of region ( R_t )</td>
</tr>
<tr>
<td>( C_t )</td>
<td>total number of tracks in region ( R_t )</td>
</tr>
<tr>
<td>( O_t )</td>
<td>total number of tracks occupied by obstacles in region ( R_t )</td>
</tr>
<tr>
<td>( G_t )</td>
<td>set of net segments in region ( R_t )</td>
</tr>
<tr>
<td>(</td>
<td>G_t</td>
</tr>
<tr>
<td>( S_t )</td>
<td>set of shield segments in region ( R_t )</td>
</tr>
<tr>
<td>(</td>
<td>S_t</td>
</tr>
<tr>
<td>( O_{tr} )</td>
<td>segment overflow in region ( R_t )</td>
</tr>
<tr>
<td>( N_t )</td>
<td>signal net</td>
</tr>
<tr>
<td>( p_{i1} )</td>
<td>source pin of net ( N_t )</td>
</tr>
<tr>
<td>( p_{i,j} )</td>
<td>( j )-th sink of net ( N_t )</td>
</tr>
<tr>
<td>( l_{i,j} )</td>
<td>routing length from ( p_{i1} ) to ( p_{i,j} )</td>
</tr>
<tr>
<td>(</td>
<td>P_t</td>
</tr>
<tr>
<td>(</td>
<td>N_t</td>
</tr>
<tr>
<td>( n_{i,t} )</td>
<td>total number of net segments in the route for net ( N_t )</td>
</tr>
<tr>
<td>( r_{i,t} )</td>
<td>physical sensitivity of ( N_t ) in region ( R_t )</td>
</tr>
<tr>
<td>( B_{i,j} )</td>
<td>set of regions containing the route for sink ( p_{i,j} ) of net ( N_t )</td>
</tr>
<tr>
<td>( L_{i,t} )</td>
<td>self inductance for ( N_t )</td>
</tr>
<tr>
<td>( M_{i,j,t} )</td>
<td>mutual inductance between ( N_{i,t} ) and ( N_{j,t} )</td>
</tr>
<tr>
<td>( K_{i,t} )</td>
<td>inductive coupling coefficient between ( N_{i,t} ) and ( N_{j,t} )</td>
</tr>
<tr>
<td>( K_{i,t} )</td>
<td>total inductive coupling for net segment ( N_{i,t} )</td>
</tr>
<tr>
<td>( K_{i,j} )</td>
<td>bound of ( K_{i,t} )</td>
</tr>
<tr>
<td>( LSK_{i,j} )</td>
<td>( LSK ) value of sink ( p_{i,j} ) of net ( N_t )</td>
</tr>
<tr>
<td>( LSK_{i,j} )</td>
<td>bound of ( LSK ) at sink ( p_{i,j} ) of net ( N_t )</td>
</tr>
<tr>
<td>( \text{sign}(x) )</td>
<td>( \text{sign}(x) = 1 ) if ( x &gt; 0 ), otherwise, ( \text{sign}(x) = 0 )</td>
</tr>
</tbody>
</table>

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 \( \textit{logic sensitivity rate} \) of signal net \( N_t \) is defined as the ratio of the number of aggressors for \( N_t \) to the total number of signal nets. During the global routing stage, however, two logically sensitive nets are considered to be \( \textit{physically sensitive} \) to each other only if they are routed within the same routing region \( R_t \). Therefore, the \( \textit{physical sensitivity rate} \) \( r_{b,t} \) of net segment \( N_{b,t} \) in region \( R_t \) is defined as the ratio of the number of aggressors in \( R_t \) for \( N_{b,t} \) 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 \( N_{i,t} \) and \( N_{j,t} \) in region \( R_t \) is defined as:

\[
K_{i,j,t} = \frac{M_{i,j,t}}{\sqrt{L_{i,t} \cdot L_{j,t}}} \tag{1}
\]

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

- when \( n_{i,t} = n_{j,t} \), the mutual inductance is reduced to self inductance and \( K_{i,j,t} = 1 \) by definition;
- when \( n_{i,t} \) (or \( n_{j,t} \)) becomes \( s_{i,t} \) (or \( s_{j,t} \)), \( K_{i,j,t} = 0 \) 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, \( K_{i,j,t} \) can be approximated by the mean of two linear interpolations of the above two special cases. Therefore, we have:

\[
K_{i,j,t} = \frac{(f(i,t) + g(j,t))}{2} \tag{2}
\]

where \( f(i,t) = (n_{i,t} - s_{i,t})/(n_{j,t} - s_{i,t}) \) and \( g(j,t) = (s_{j,t} - n_{j,t})/(s_{j,t} - n_{j,t}) \) 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 \( K_{i,t} \) as
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 $K_{\text{eff}}$ (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 $K_{\text{eff}}$ 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 $\text{LSK}$. The $\text{LSK}$ value for a net $N_i$ at its $j$th sink is defined as

$$\text{LSK}_{ij} = \sum_{l \in B_{ij}} l_t \cdot K_{it}$$

where $l_t$ is the length of $B_t$ 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.
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 post-routing 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 \((e) = (e_1, e_2, \ldots, e_\ell)\), 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 \((V, E)\). 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 post-routing 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 \(K_{it}\) in RLC 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

\[
|S_i| = a_1 \cdot \sum_{N_{it} \in G_i} K_{it} \cdot r_{it} + a_2 \cdot \sum_{N_{it} \in G_i} r_{it}
\]

(5)

where \(a_1 = -0.10491\) and \(a_2 = 0.49392\) 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 \(|G_i|\) and physical sensitivity rates \(r_{it}\) are fixed in a region \(G_i\) for the given global routing solution, hence (5) can be further simplified as a linear combination of the given noise bounds \(K_{it}\).

\[
|S_i| = \sum_{N_{it} \in G_i} \alpha_{it} \cdot K_{it} + \beta_i
\]

(6)

where \(\alpha_{it} = a_1 \cdot r_{it}\) and \(\beta_i = a_2 \cdot \sum_{N_{it} \in G_i} r_{it}\) according to (5). Because \(a_1\) is negative, so are all coefficients \(\alpha_{it}\) and \(\beta_i\).

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 \(K_{it}\) 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 \(K_{it}\) 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
determines the search space for net \( N_i \)'s final routing solution. In other words, each edge \( e_k \in E_i \) 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 \( s_0 \) and a specific sink \( s_{ij} \) of signal net \( N_i \) to approximate their final wire length of a routing solution, which is denoted as \( L_{ij} \). If the LSK crosstalk bound at sink \( s_{ij} \) is denoted as \( LSK_{ij} \), the total inductive coupling bounds for all net segments \( N_{it} \) in \( G_i(V_i, E_i) \) are determined by:

\[
K_{it} = \frac{LSK_{ij}}{L_{ij}}. \tag{7}
\]

If a net segment is shared by multiple paths from the same source to multiple sinks, the minimum \( K_{it} \) 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:

\[
O_{it} = |S_i| + |G_t| + O_t - C_t = \left( \sum_{N_{it} \in G_t} \alpha_{it} \cdot K_{it} + b_t \right) + |G_t| + O_t - C_t \tag{8}
\]

where \( |S_i| \) 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 \( O_{it} \) 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 shielding 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 network connection graphs for all nets, the set of which is denoted as \( \Omega \), and initialize all net segments in \( \Omega \) as noncritical. We then iteratively identify the region in \( \Omega \) 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, \( \Omega \) 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
for all net segments from 1 and 2, where $p$ with pattern routing. Therefore, the ID/S algorithm candidate patterns in its corresponding routing is the total number of SSMS with potential net segments in 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 $G_t(V_t, E_t)$. 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 $G_t$ noncritical and hence those net segments can be simply deleted.

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 $G_t(V_t, E_t)$ 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 $G_t(V_t, E_t)$. 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 $G_t$ 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 $R_i$)
  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 $N_i$, 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 $N_i$ 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 $N_i$. 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 $N_i$ 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 $N_j$ 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 $N_i$. 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

<table>
<thead>
<tr>
<th>Circuits</th>
<th># of nets</th>
<th># of regions</th>
<th># of pins</th>
<th>Region capacity</th>
</tr>
</thead>
<tbody>
<tr>
<td>prim1.1</td>
<td>1266</td>
<td>8 × 16</td>
<td>2738</td>
<td>H:14 V:18</td>
</tr>
<tr>
<td>prim1.2</td>
<td>1266</td>
<td>16 × 16</td>
<td>2738</td>
<td>H:12 V:12</td>
</tr>
<tr>
<td>prim1.3</td>
<td>1266</td>
<td>128 × 16</td>
<td>2738</td>
<td>H:12 V:3</td>
</tr>
<tr>
<td>prim2.1</td>
<td>3817</td>
<td>8 × 16</td>
<td>10971</td>
<td>H:32 V:42</td>
</tr>
<tr>
<td>prim2.2</td>
<td>3817</td>
<td>32 × 32</td>
<td>10971</td>
<td>H:18 V:14</td>
</tr>
<tr>
<td>prim2.3</td>
<td>3817</td>
<td>128 × 28</td>
<td>10971</td>
<td>H:16 V:8</td>
</tr>
<tr>
<td>ibm07</td>
<td>13056</td>
<td>64 × 64</td>
<td>44266</td>
<td>H:20 V:14</td>
</tr>
<tr>
<td>ibm08</td>
<td>26104</td>
<td>80 × 64</td>
<td>75710</td>
<td>H:38 V:28</td>
</tr>
<tr>
<td>ibm04</td>
<td>31328</td>
<td>96 × 64</td>
<td>89591</td>
<td>H:30 V:30</td>
</tr>
</tbody>
</table>

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 $[(3, S, 4, 2, S, 1, 5)]$ of an edge means that the final net order from left to right in that edge are: $N_3$, Shield, $N_4$, Shield, $N_1$, $N_2$.

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 $LSK$ 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
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 prim1.3 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 prim1.3 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

<table>
<thead>
<tr>
<th>Test Circuit</th>
<th>Phase I</th>
<th>Phase II</th>
<th>Phase III</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shield (%)</td>
<td>OvSeg</td>
<td>OvReg</td>
<td>Shield (%)</td>
</tr>
<tr>
<td>prim1.1 GR+SINO</td>
<td>336.7</td>
<td>98.0</td>
<td>36.0</td>
</tr>
<tr>
<td>GSINO</td>
<td>339.6</td>
<td>51.3 (-47.7)</td>
<td>28 (-22.2)</td>
</tr>
<tr>
<td>prim1.2 GR+SINO</td>
<td>517.6</td>
<td>171.5</td>
<td>84</td>
</tr>
<tr>
<td>GSINO</td>
<td>507.6</td>
<td>101.6 (-40.8)</td>
<td>53 (-36.9)</td>
</tr>
<tr>
<td>prim1.3 GR+SINO</td>
<td>2345.5</td>
<td>92.0</td>
<td>417</td>
</tr>
<tr>
<td>GSINO</td>
<td>2381.8</td>
<td>483.6 (-50.8)</td>
<td>308 (-26.1)</td>
</tr>
<tr>
<td>prim2.1 GR+SINO</td>
<td>666.6</td>
<td>192.5</td>
<td>42</td>
</tr>
<tr>
<td>GSINO</td>
<td>645.5</td>
<td>49.2 (-74.4)</td>
<td>16 (-61.9)</td>
</tr>
<tr>
<td>prim2.2 GR+SINO</td>
<td>3108.8</td>
<td>503.4</td>
<td>127</td>
</tr>
<tr>
<td>GSINO</td>
<td>3078.6</td>
<td>428.8 (-14.8)</td>
<td>215 (-5.3)</td>
</tr>
<tr>
<td>prim2.3 GR+SINO</td>
<td>7952.1</td>
<td>2362.8</td>
<td>787</td>
</tr>
<tr>
<td>GSINO</td>
<td>7810.9</td>
<td>1647.1 (-30.3)</td>
<td>630 (-19.9)</td>
</tr>
</tbody>
</table>

TABLE V

COMPARISON OF SHEILD INSERTION OVERHEAD BETWEEN GR, GR+SINO AND GSINO

<table>
<thead>
<tr>
<th>Test Circuit</th>
<th>GR</th>
<th>GR+SINO</th>
<th>GSINO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overhead (%)</td>
<td>OvSeg</td>
<td>OvReg</td>
<td>OvSeg</td>
</tr>
<tr>
<td>prim1.1 GR+SINO</td>
<td>22</td>
<td>12</td>
<td>178 (709.0%)</td>
</tr>
<tr>
<td>GSINO</td>
<td>8.4</td>
<td>1397.0</td>
<td>1614.2</td>
</tr>
<tr>
<td>prim1.2 GR+SINO</td>
<td>16.8</td>
<td>1623.7</td>
<td>1721.2</td>
</tr>
<tr>
<td>GSINO</td>
<td>19.7</td>
<td>1601.7</td>
<td>1586.7</td>
</tr>
<tr>
<td>prim1.3 GR+SINO</td>
<td>124.5</td>
<td>6957.3</td>
<td>6611.2</td>
</tr>
<tr>
<td>GSINO</td>
<td>118.8</td>
<td>6671.7</td>
<td>6014.4</td>
</tr>
<tr>
<td>prim2.1 GR+SINO</td>
<td>98.2</td>
<td>8700.0</td>
<td>18470.4</td>
</tr>
<tr>
<td>GSINO</td>
<td>104.4</td>
<td>8794.2</td>
<td>17134.7</td>
</tr>
<tr>
<td>prim2.2 GR+SINO</td>
<td>930.4</td>
<td>12176.0</td>
<td>16344.0</td>
</tr>
<tr>
<td>GSINO</td>
<td>889.2</td>
<td>11979.0</td>
<td>15740.7</td>
</tr>
<tr>
<td>prim2.3 GR+SINO</td>
<td>2095.7</td>
<td>26552.9</td>
<td>29651.0</td>
</tr>
<tr>
<td>GSINO</td>
<td>1777.1</td>
<td>25597.1</td>
<td>27975.6</td>
</tr>
</tbody>
</table>

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. 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 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 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 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 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 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 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 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
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

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.