Redundant Local-Loop Insertion for Unidirectional Routing

Xiaoqing Xu Student Member, IEEE, Yibo Lin, Meng Li Student Member, IEEE, Jiaojiao Ou, Brian Cline Member, IEEE, and David Z. Pan Fellow, IEEE

Abstract—As the semiconductor manufacturing technology continues to scale down to sub-10nm, unidirectional layout style has become the mainstream for lower metal layers with tight pitches. Conventional redundant via insertion for yield improvement has become obsolete because unidirectional routing patterns forbid off-track routing, i.e., wire bending, for the metal coverage of redundant vias. To enhance the yield, redundant local-loop insertion (RLLI) is a new way of inserting redundant vias due to its compatibility with the unidirectional layout style. This paper proposes the first global optimization engine for RLLI considering advanced manufacturing constraints. Our key contributions include bounded timing impact analysis and evaluation for the local-loop structure, net-based local-loop candidate generation and pruning, an integer linear programming (ILP) formulation and scalable iterative relaxation/linear programming solving (IRLS) with incremental search scheme. Experimental results demonstrate that with bounded timing impact (within 1%), the ILP formulation obtains highest insertion rate while the IRLS with incremental search scheme achieves scalable solutions with competitive solution qualities.

Keywords—Redundant Local Loop, Linear Programming, Via Density, Timing Analysis

I. INTRODUCTION

Due to continued scaling of semiconductor technology, the manufacturing process is becoming more and more sensitive to process variations and random failures. In particular, via and wiring failures are major causes for the yield loss of the integrated circuit (IC) during the back-end-of-line (BEOL) process [1]. To reduce potential via and wiring failures at the post-routing stage, redundant via [2], [3] and redundant wire [4], [5] insertions have been proposed for manufacturing yield improvement. In advanced technology nodes, metal connection stack consists of multiple metal layers with various metal width and spacing [6]. For upper metal layers, the wiring width and spacing are relaxed to provide low resistance and timing “short-cuts” for long nets. For lower metal layers, the wiring, the geometric scaling leads to high metal density, which is enabled by complex design-for-manufacturing constraints, such as multiple patterning lithography and unidirectional layout style. Redundant via insertion (RVI) has been widely used in industry to improve the manufacturing yield of metal interconnections, where non-minimum width wire and wire bending are allowed. In 10nm and beyond, RVI is still feasible for upper metal layers with relaxed metal pitches. However, for lower metal layers with tight pitches, unidirectional routing has become the mainstream routine, which complies with the underlying multiple patterning constraints [7], [8]. Unidirectional routing style makes conventional RVI (double via) obsolete because unidirectional routing patterns forbid off-track wiring, i.e., wire bending, for the metal coverage of redundant vias. As shown in Fig. 1(a), two redundant vias have been inserted for associated single vias. Metal-2 (M2) and metal-3 (M3) tracks are horizontal and vertical, respectively. The redundant via (RV) on the M2 track introduces M3 wire bending and vice versa, which are strictly forbidden under the restrictive unidirectional routing style. Therefore, as demonstrated in Fig. 1(b), redundant local-loop insertion (RLLI), as a supplementary scheme for lower metal layers, can simultaneously insert redundant vias and redundant wires for yield improvement of unidirectional routing [1]. A redundant local loop (RLL) introduces via and wiring redundancy, i.e., redundant M2/M3 patterns, to guarantee all redundant metal patterns are on-track, which adheres to the unidirectional routing style.

[1] and [9] first introduced the local loop concept back in 2006. An ad-hoc local-loop insertion routine was proposed at the post-routing stage. A recent work [10] studied the yield and timing impact of local loops of different sizes. It confirmed the yield enhancement and demonstrated very small or negligible timing impact using local loops with the empirical timing simulations. However, comprehensive delay model analysis and timing simulations for various RLL configurations were not provided [10]. Furthermore, the continued technology scaling has imposed advanced manufacturing constraints on via layers. Among them, self-aligned via (SAV) [11], [12]

Fig. 1: (a) RVI with wire bending, (b) RLLI for unidirectional routing patterns.
and via density constraints [13] are particularly important for manufacturing via patterns in advanced technology nodes. In addition, our predictive timing simulations show that significant timing degradations could be introduced with specific kinds of RLLs. This means that those kinds of RLLs should be strictly forbidden to guarantee bounded and negligible timing impact for RLLI. It should be noted that for full-chip local-loop insertion, these advanced manufacturing constraints on via patterns and bounded timing impact from RLLI have to be considered, but the prior work [1], [9], [10] focused on proving and validating the local-loop concept, and did not address the advanced manufacturing constraints and timing impact aforementioned.

Moreover, all prior approaches for RLLI are greedy in nature [1], [9], [10] and no systematic algorithms were proposed for local-loop insertion in a full-chip manner. In contrast, traditional RVI has been extensively studied with various advanced optimization schemes, including the maximum independent set problem formulation, solved with fast heuristics [2], [14], [15], and 0-1 integer linear programming (ILP) formulation, solved with speed-up techniques [14], [16]–[18]. The conflict constraints for traditional RVI problem are purely local, which means only redundant-via candidates for neighboring single vias will introduce conflicts during RVI. This is the major reason for the high solution qualities obtained from fast heuristics and speed-up techniques for RVI. However, for RLLI, one RLL candidate may consist of multiple via and metal grids, which induces long-range conflict constraints among RLL candidates. Meanwhile, redundant vias within one RLL candidate may cross multiple density windows. This means redundant vias within RLL candidates need to be balanced under via density constraints across multiple density windows. Therefore, global optimization schemes are strongly needed to achieve better performance than the simple greedy scheme [1], [9], [10]. Moreover, the RLLI is not merely about maximizing insertion rate, because reducing the insertion cost is critical to considering distinct routing resource usages, timing and yield impact for different redundant local-loop candidates (RLLCs). It is important to combine the optimization of insertion rate and cost while accommodating conflict and density constraints, which makes ILP a competitive candidate for the global optimization scheme.

To the best of our knowledge, as technology moves toward unidirectional routing and vias continue to scale to extremely small geometries, which are difficult to yield and have high parasitic resistance, we expect increasing adoption of local loops in 10nm and beyond. In this paper, we propose the first global optimization engine for full-chip local-loop insertion, in consideration of advanced manufacturing constraints and bounded timing impact guided by SPICE simulations. Under bounded search space, we enumerate the RLLCs for each single via. Bounded timing impact is demonstrated with comprehensive Elmore delay model analysis and timing simulations. With routing grid model, we further analyze the conflict constraints among RLLCs and formulate the RLLI issue as a binary ILP problem, which simultaneously improves the insertion rate and reduces the overall cost of inserted RLLs. Although the number of RLLCs generated is considerable by limiting the search space, it is still much larger than that of the traditional RVI problem, which means the ILP formulation is not scalable to large designs with large numbers of single vias. However, theoretical analysis reveals that, due to the special constraint structures of ILP formulation, the linear programming (LP) relaxation leads to a solution that is intrinsically close to the ILP-integral solution with appropriate rounding schemes [19], [20]. Thus, we further propose the iterative relaxation and linear programming solving (IRLS) with incremental search scheme to achieve scalable solutions with affordable performance degradations. Our main contributions are summarized as follows.

- We demonstrate bounding the timing impact due to RLLI using Elmore delay model analysis and SPICE simulations under the 10nm predictive technology setup.
- We propose the first global optimization engine for the RLLI at the post-routing stage considering advanced manufacturing constraints and bounded timing impact from RLLI.
- With the net-based RLLC generation and pruning, we formulate the RLLI issue as a binary ILP problem, which simultaneously improves the insertion rate and reduces the overall cost of inserted RLLs.
- Taking advantage of the special constraint structures of ILP formulation, we propose the IRLS with incremental search scheme to obtain scalable solutions with negligible solution quality degradations.

The rest of this paper is organized as follows: Section II briefly introduces relevant background information and defines the RLLI problem. Section III analyzes the timing impact and discusses the net-based RLLC generation and pruning for the RLLI problem, which is solved with the binary ILP formulation and the scalable IRLS with incremental search scheme. Section IV compares the experimental results under different RLLI schemes and demonstrates the effectiveness of our proposed approaches. Section V concludes the paper.

II. PRELIMINARIES

A. Redundant Local Loop

For unidirectional routing patterns, RLLs are inserted to reduce the failure rate of within-loop single vias (SVs) and one RLL could cover multiple SVs as shown in Fig. 2, where RLLs are inserted involving the M3, M2 and Via-2 (V2) layers. RLL1 consists of 3 redundant M3 grids, 3 redundant M2 grids and 1 redundant-via grid while RLL2 includes 5 redundant M3 grids, 3 redundant M2 grids and 2 redundant-via grids. RLL3 has 3, instead of 5, redundant M3 grids due to existing M3 routing patterns, which are constrained by upper-level metal connections within a routing solution. In practice, we need to differentiate the cost of RLLs with different configurations and the reasons are twofold. First, the RLLs with less number of redundant vias are preferred since a larger number of vias leads to larger timing impact of a local-loop structure [10], which will be demonstrated with comprehensive SPICE simulations in Section IV-A. Second, RLLs with different configurations may lead to distinctive yield impacts, which highly depends...
on the technological setup and analysis [1], [10]. Therefore, we define the configuration and cost of an RLL as follows.

**Definition 1 (Redundant Local Loop)** An RLL with the configuration as \( rm_{x+1} \times rm_x \times rv_x \) is defined as a loop structure with \( rm_{x+1} \) redundant grids on the \( x+1 \) metal layer, \( rm_x \) redundant grids on the \( x \) metal layer and \( rv_x \) redundant grids on the \( x \) via layer. Its cost is defined as \( \alpha \cdot rm_{x+1} + \beta \cdot rm_x + \gamma \cdot rv_x \), where \( \alpha \), \( \beta \) and \( \gamma \) are user-defined parameters.

Since the RLLI problem targets at inserting local-loop structures for SVs, we can enumerate valid RLLCs for an SV within a limited local search space [10]. In this work, we limit the local search space by bounding the number of redundant metal grids (i.e., \( rm_{x+1} \) or \( rm_x \)) by a pre-determined parameter. Moreover, various RLLCs for an SV can be differentiated using the cost definition above, where \( \alpha \), \( \beta \) and \( \gamma \) are set to make the cost inversely proportional to the yield and timing improvement for an RLLC.

**B. Advanced Manufacturing Constraints**

1) **Via-Pattern Constraints**: In 10nm node and beyond, the center-to-center spacing of lower via layers is in the sub-lithographic domain [7]. SAV patterning [11], [12] is a promising candidate for the sub-lithographic printing of via patterns. As shown in Fig. 3(a), the neighboring vias in the horizontal direction are merged into via cuts to enable SAV patterning for vias within the same via cut. The SAV patterning takes advantage of the line spacers from manufacturing procedure and selectively etch via trenches defined by the via cuts, which enables sub-lithographic printing of via patterns within the same via cut. The patterning of via cuts is technology dependent and correlated to the M2/M3 patterning on the lower/upper metal layer. The details on the manufacturing procedures are introduced in [11], [12]. This work considers basic SAV constraints in [12]. For a single via in a routed net, the neighboring via grids along the upper metal layer direction, such as vertical M3, are not available for RVs and the neighboring via grids along the lower metal direction, such as horizontal M2, are only available for same-net redundant vias. An example is shown in Fig. 3(b). Our RLLI schemes only depend on the routing grid model and can be easily adapted to incorporate more complicated SAV constraints.

2) **Metal-Pattern Constraints**: Unidirectional metal patterns are manufacturing friendly in the sub-lithographic domain, which enables track-based coloring schemes for multiple patterning lithography. This work addresses basic metal-pattern constraints including minimum metal tip-to-tip rule (1 metal grid) for line-end control and minimum metal-length rule (2 metal grids) to avoid short metal defects. In advanced technology nodes, complex metal pattern rules are technology dependent. For example, self-aligned multiple patterning introduces complex line-end rules, which can be explicitly considered by avoiding prohibited line-end positions during the RLLC generation [21], [22]. Our work primarily focuses on the global optimization of the RLLI and provides a design-rule-checker interface, which enables the incorporation
of technology-specific metal and via pattern constraints (see Section III-B).

C. Problem Definition

Given a design with unidirectional routing for each net on a routing grid model, this work focuses on the RLLI with bounded timing impact for routed single vias at the post-routing stage. Apart from the advanced manufacturing constraints aforementioned, we need to explicitly consider the following constraints to obtain legal RLLIs. First, at most one RLL should be inserted for one single via. Second, two RLLs inserted cannot occupy the same grid unless they belong to the same net. Third, the RLL scheme should minimize the total cost of RLLs inserted since the cost is defined to be inversely proportional to the timing and yield improvement of a specific RLLC. Thus, we define the RLL problem as follows.

Problem 1 (Redundant Local-Loop Insertion) Given the unidirectional routing design and a set of density windows, the RLLI problem is to insert RLLs to cover as many routed single vias as possible while reducing the total cost of the inserted RLLs and accommodating advanced manufacturing constraints and bounded timing impact.

III. REDUNDANT LOCAL-LOOP INSERTION

A. Timing Impact Analysis

As shown in Fig. 2, an RLLC consists of redundant metal wires and vias, which generates a loop structure in routing solution. This makes timing analysis more complicated than that of the original routing tree. With the simplified RC network, our Elmore delay model provides a closed-form solution of the delay impact with signal delay computation in non-tree RC networks [23], which extends the empirical timing analysis in [10]. Fig. 5(a) shows the simplified RC network for an RLLC, where notations are given in Table I.

The primary path (from routing tree) and secondary path (from RLLC) from driver to load are denoted by black and red lines, respectively. This RC network can model any RLL defined in Definition 1 if the related resistance and capacitance are provided. The Elmore delay computation in non-tree topology [23] is based on the split RC network shown in Fig. 5(b). The key idea is that node $N$ can be split into three independent nodes, i.e. $N_1$, $N_2$ and $N_3$, and the delay of the three nodes are the same. Further delay analysis and computation yield the following closed-form solution of the delay impact from driver to load due to RLLI$^1$.

$$\Delta \text{delay} = C_s \cdot R_d + \frac{C_s \cdot R_p \cdot R_s - C_p \cdot R_p^2 - 2 \cdot C_1 \cdot R_s^2}{2 \cdot (R_p + R_s)}$$

The close-form solution of $\Delta \text{delay}$ reveals that the delay impact from RLLI could be quite different depending on specific resistance and capacitance parameters. In general, timing degradations could be arbitrarily small even negative (timing improvements) if the resistance ($R_s$) and capacitance ($C_s$) on the secondary path are much smaller compared to those ($R_p$ and $C_p$) on the primary path. If secondary-path resistance and capacitance are relatively large, associated RLLCs should not be inserted due to prohibitively large timing degradations.

The closed-form solution (Elmore delay) is computed using simplified RC network for primary and secondary paths, where via and metal resistance/capacitance on the primary/secondary path are combined for one pi model as shown in Fig. 5(a).

For accurate timing evaluations, Elmore delay is conservative compared to SPICE simulations. Moreover, the via/metal resistance and capacitance shall be modeled separately for better accuracy, which generates a much more complex RC network. Therefore, comprehensive SPICE simulations shall be performed with complex RC networks aforementioned to determine actual timing impacts. As will be discussed in Section IV-A, the RLLI could introduce a wide range of timing impact, i.e. delay increase or decrease, depending on specific RLL configurations. To enable practical adoption of RLLI, we bound the resulting timing impact due to RLLI by forbidding RLLCs with timing degradations greater than a preset timing impact amount. Therefore, the timing impact analysis and simulation yield a look-up table (LUT) of forbidden RLLC configurations under the preset timing impact bound, which will be used for RLLC generation and pruning$^2$.

B. RLLC Generation and Pruning

We first discuss the RLLC generation for each single via (SV). As mentioned in Section II, $rm_{x+1}$ and $rm_x$ are bounded by a preset parameter for limiting the search space of RLLCs for a single via. The region for valid RLLCs is bounded by a preset parameter for limiting the search space of RLLC configurations. To enable practical adoption of RLLI, we bound the resulting timing impact due to RLLI by forbidding RLLCs with timing degradations greater than a preset timing impact amount. Therefore, the timing impact analysis and simulation yield a look-up table (LUT) of forbidden RLLC configurations under the preset timing impact bound, which will be used for RLLC generation and pruning$^2$.

---

$^1$The detailed principles of computation for $\Delta \text{delay}$ are given in [23].

$^2$Our RLL scheme is independent of the timing engine as long as an LUT of forbidden RLLC configurations is given.
We need a compact vector of RLLCs because only one binary variable is generated for an RLLC covering multiple single vias in mathematical formulations. From line 14 to line 17, we obtain a vector to index the RLLCs for each single via in the net. The RLLC vector for each single via \( RLLC_{vi} \) will only be selected from the compact set of RLLCs, i.e. \( RLLC_{net} \), computed for the net. This guarantees no duplicate among the RLLCs generated for each single via.

We analyze the complexity of Algorithm 1. Related notations are defined in Table II. The net-based RLLC generation and pruning form a loop over the total number of nets \( N \) in a design, which means the complexity is linear to \( N \). Within each loop, all RLLCs enumerated are stored in a set \( RLLC_{net} \). The maximum size of \( RLLC_{net} \) is bounded by \( O(M \cdot R) \), i.e. maximum number of single vias per net times the maximum number of RLLCs per single via. Thus, the set \( RLLC_{net} \) construction time (from line 4 to line 9 in Algorithm 1) is \( O(M \cdot R \cdot \log(M \cdot R)) \). In lines 10 to 13 and lines 14 to 17, merging \( RLLC_{net} \) to \( RLLC \) and RLLC selection for each single via both take \( O(M \cdot R) \) time. The enumeration loop in lines 3 to 18 takes \( O(N \cdot M \cdot R \cdot \log(M \cdot R)) \) time. In line 20, we sort the RLLCs for each SV based on cost and a linear-time scan can prune out redundant candidates, which altogether takes \( O(R \cdot \log(R)) \). The total number of single vias
TABLE II: Notations for RLLI

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>the number of nets</td>
</tr>
<tr>
<td>M</td>
<td>the maximum number of SVs for one net</td>
</tr>
<tr>
<td>R</td>
<td>the maximum number of RLLCs for each SV</td>
</tr>
<tr>
<td>L</td>
<td>the total number of RLLCs</td>
</tr>
<tr>
<td>SV</td>
<td>the pre-defined set of SVs</td>
</tr>
<tr>
<td>ilc_i</td>
<td>the i-th SV</td>
</tr>
<tr>
<td>c_i</td>
<td>the cost of ilc_i</td>
</tr>
<tr>
<td>n_i</td>
<td>the number of single vias that ilc_i covers</td>
</tr>
<tr>
<td>x_i</td>
<td>the binary variable for ilc_i</td>
</tr>
<tr>
<td>X_i</td>
<td>the variable set for the RLLCs covering SV_i</td>
</tr>
<tr>
<td>G_i</td>
<td>the</td>
</tr>
<tr>
<td>W_k</td>
<td>the k-th density window</td>
</tr>
<tr>
<td>v_i</td>
<td>the number of RVs of ilc_i in W_k</td>
</tr>
<tr>
<td>D_B_k</td>
<td>via density upper bound for W_k</td>
</tr>
<tr>
<td>W</td>
<td>the set of density windows (W_k)</td>
</tr>
</tbody>
</table>

is bounded by $O(N \cdot M)$. The pruning technique in lines 19 to 21 takes $O(N \cdot M \cdot R \cdot \log(R))$ time. Therefore, the time complexity of Algorithm 1 is $O(N \cdot M \cdot R \cdot \log(M \cdot R))$. We further assume each net has approximately the same number of single vias and each single via has approximately the same number of RLLCs, i.e. $M \approx |SV|/N$ and $R \approx L/|SV|$. Then, the time complexity of Algorithm 1 reduces to $O(N \cdot |SV|/N \cdot L/|SV|) \cdot \log(|SV|/N \cdot L/|SV|)$, i.e. $O(L \cdot \log(L/|N|))$.

The last step in Algorithm 1 from line 19 to line 21 is to further prune RLLCs without loss of optimality. For $RLLC_{v_i}$ associated with a single via $v_i$, if there exists an isolated RLLC with a cost of $c_i$, we can potentially prune some RLLCs in $RLLC_{v_i}$. Those RLLCs shall have a larger cost than $c_i$ and occupy the same via density windows as the isolated RLLC with a cost of $c_i$. An isolated RLLC means no other RLLCs conflict with it on the routing grid except for those RLLCs belonging to the same single via. Suppose any pruned RLLC is eventually selected in an optimal RLLI solution, we can always replace it with the isolated one for a lower cost and meanwhile the window density distributions remain the same. This contradicts with the optimality of the given RLLI solution. Thus, the pruning technique guarantees optimality. As discussed in the following section, each RLLC has an associated binary variable. Then, the RLLC pruning techniques in Algorithm 1 help to reduce the number of variables and constraints for our ILP formulation.

C. ILP Formulation

With the RLLCs generated for each single via, Problem 1 becomes a general assignment problem to cover as many single vias as possible while reducing the total cost of inserted RLLs. Related notations are shown in Table II. We assume each RLLC has binary variable $x_i$ that denotes whether the $i$-th RLLC $ilc_i$ is selected to cover associated single vias. In particular, only one binary variable is generated for an RLLC covering more than one single via. Then, Problem 1

\[
\text{max } CB \cdot \sum_{i=1}^{n} a_i \cdot x_i - \sum_{i=1}^{n} c_i \cdot x_i \\
\text{s.t. } \sum_{i=1}^{n} x_i \leq 1 \quad \forall X_j \in X \quad (C1) \\
\sum_{i=1}^{n} x_i \leq 1 \quad \forall A \in G \cup SA \quad (C2) \\
\sum_{i=1}^{n} v_{ik} \cdot x_i \leq DB_k \quad \forall W_k \in W \quad (C3) \\
x_i \in \{0, 1\} \quad \forall x_i \in X_j \in X \quad (C4)
\]

can be formulated as a binary ILP problem. The objective of our ILP formulation consists of the weighted summation of the associated RLLC binary variable $x_i$, which should be maximized to improve the insertion rate and reduce the cost of inserted RLLCs. The objective (1) consists of two terms. The term $(CB \cdot \sum_{i=1}^{n} a_i \cdot x_i)$ is the first objective, which improves the insertion rate. The parameter $n_i$ is added before $x_i$ to consider that one RLLC may cover multiple single vias. The selection of that particular RLLC can cover $n_i$ single vias. The term $(-\sum_{i=1}^{n} c_i \cdot x_i)$ is the second objective, which reduces the overall cost of inserted RLLs. The two terms are balanced using the parameter $CB$, where $CB > \max c_i$. Therefore, our formulation can simultaneously improve the insertion rate and reduce the overall cost of inserted RLLs.

To ensure legal assignment results for the RLLI problem, we consider three sets of constraints listed as follows.

- At most one RLLC is assigned to each single via. Thus, the summation of all $x_i \in X_j$ is bounded by one to guarantee that at most one binary variable will be 1 as listed in constraint (C1).
- Conflict constraints are primarily related to two reasons. First, one single grid can only be occupied by one RLLC. Second, conflicting via grids cannot be occupied simultaneously under $SAV$ constraints. Related constraints are represented by constraint (C2).
- Via density constraints are applied to each density window. Since one RLLC may have multiple vias and cross multiple density windows, the value of coefficient $v_{ik}$ denotes the number of RVs from $ilc_i$ in window $W_k$. The total number of vias in a window should not exceed an upper bound $DB_k$ and the related constraints are listed in constraint (C3).

Our RLLI works on the given routing grid model. The metal and via grids store references to the RLLCs occupying those grids during the RLLC generation and pruning. Then, a simple grid traversal can set up all the constraints aforementioned. The objective of the ILP formulation combines two objectives for simultaneous insertion rate and RLL cost optimization. Moreover, the ILP formulation can globally balance the via densities for constraints (C3), which generates much better solution qualities than a simple greedy scheme.

\footnote{Similar conflicting constraints can be added once technology-dependent design rules are given for metal and via patterns.}
D. Special Constraint Structures

Although optimal solutions can be obtained with the ILP formulation for Problem 1, the exponential time complexity of the ILP formulation makes it unscalable to large problem sizes. It shall be noted that a typical ILP solving approach for Problem 1 involves two steps [19], [20]. The first step is to relax the binary constraints in constraint \((C_4)\) into the linear constraints \((0 \leq x_i \leq 1)\) and solve the LP-relaxed problem instance. The second step is to retrieve the binary solution via branch and bound scheme with the bounding information provided by LP-relaxed solution, which can be prohibitively time-consuming for a complex ILP problem.

Fortunately, the special constraint structures in Problem 1 lead to an LP-relaxation solution that is intrinsically close to a binary solution. Specifically, the selection constraints in \((C_1)\) and conflicting constraints in \((C_2)\) are strong valid inequalities [19] and can reduce the feasible region (polytope) of the LP-relaxed problem instance, which makes the LP-relaxed solution close to a binary solution. Strong invalid inequality means that each constraint in \((C_1)\) and \((C_2)\) is tight. Supposing \(k\) RLLCs conflict with each other, only one variable within \(\{x_i, \forall i \in [1, k]\}\) can be assigned as 1, which can be formulated as two kinds of linear constraints, including loose inequality in \((C_5)\) and strong invalid inequality in \((C_6)\).

\[
x_i + x_j \leq 1 \quad \forall i, j \in [1, k] \tag{C5}
\]

\[
\sum_{i=1}^{k} x_i \leq 1 \tag{C6}
\]

The selection constraints in \((C_1)\) and conflict constraints in \((C_2)\) are generated for each single via and each routing grid, which makes them strong in nature. For the geometric solution space of an LP-relaxed instance, the strong valid inequalities, such as \((C_6)\), play as cutting planes to bound the LP solution to close to integral [19]. The via density constraints in \((C_3)\) turn out to be the same as capacity constraints for the ILP formulation of a constrained multiple knapsack problem [20]. It has been shown that LP-based relaxation effectively delivers integral solutions to the ILP formulation of the constrained multiple knapsack problem [20]. Therefore, due to special constraint structures in Problem 1, LP relaxation provides valuable opportunities to obtain close-to-integral solution.

Non-integral solutions generally exist from LP-relaxed instance even with strong valid inequalities \(((C_1)\) and \((C_2)\)) and capacity constraints \(((C_3)\)) for a constrained multiple knapsack problem. Thus, randomized rounding schemes are adopted in [19], [20], where final solutions can be bounded to optimal solutions with a certain probability. However, randomized rounding scheme generates uncertainty in the solution, which is not preferred for the RLLI problem. Instead, we propose a deterministic greedy scheme after solving LP-relaxed instances, which empirically delivers close-to-optimal solutions.

E. IRLS with Incremental Search

To take advantage of the special constraint structures aforementioned, we propose an IRLS with incremental search scheme to iteratively solve the LP-relaxed problem instance followed by an incremental search step for a scalable solution. Within each iteration, we determine the integral assignment, ignore non-integer results from the optimal LP solution and update the problem instance itself. The iteration stops when no integral assignment can be achieved with the LP relaxation for the problem instance. For those single vias without RLLCs assigned from the iterative LP solving procedure, an incremental search step is adopted, i.e. greedy rounding for IRLS, to improve the result quality.

The details of the IRLS scheme for Problem 1 are shown in Algorithm 2. The main loop for IRLS is from line 2 to line 24. Within each iteration, the LP relaxation of the RLLI problem is solved and optimal solutions are stored in \(X_{opt}\) at line 3. From line 6 to line 8, we obtain the integral assignment \(A_1\) from \(X_{opt}\), where only variables with solutions as 1 are collected into \(A_1\). From line 12 to line 18, we iterate through the conflict constraints for constraints \((C_1)\) and \((C_2)\). We
define the variable set in constraint $C$ as $X_c$ in line 13. In line 15, if one of the variables in $X_c$ has been assigned as 1 in $A_1$, we can deduce that all other binary variables should be assigned 0 (add to $A_0$) since each conflict constraint requires that only one binary variable can be assigned to 1. In line 16, we update the problem instance $RLLI$ by removing related conflict constraints and variables because associated binary variables have become constants with assigned 0/1 values. From line 19 to line 21, the density upper bound $DB_k$ is updated for each density window considering the integral assignment $A_1 \cup A_0$. Each RLLC with a binary variable as 1 in integral assignment $A_1$ consumes the density budgets in related density windows. This means the density upper bound of the related density windows shall be updated for the next iteration of integral assignment. In line 22, the objective is further updated with the integral assignment $A_1 \cup A_0$ since variables in $A_1 \cup A_0$ become constants for the next iteration. With the update on the problem instance $RLLI$, the next iteration of LP relaxation performs another round of integral assignment on top of the previous assignment results. We stop the iteration when no integral assignment can be obtained from the $X_{opt}$ as shown in line 9 to line 11. An incremental search step is explained in lines 25 to 31. During the incremental search, for each single via without RLL inserted (line 26), we traverse associated RLLCs in the increasing order of cost (line 28) until a legal RLLC can be inserted or no legal RLLC is obtained (line 29). This is equivalent to a greedy rounding step if no integral solution is achieved for a specific variable after IRLS procedure.

In particular, the problem scale, including the number of variables and constraints, of the later iteration will be much smaller than that of the former iteration and the LP problem instance will be updated based on fixed values (0/1). Then the solution space, i.e., polytope, of the LP instance for the later iteration will be different (smaller dimension) from the former iteration. One vertex of the polytope is one feasible solution of the LP instance. A typical LP solver adopts the simplex algorithm to seek an optimal solution by moving from one vertex to another vertex of the polytope. The optimal solution of the former iteration is a vertex of the polytope for the corresponding LP instance. It is within the solution space but not necessarily a vertex of the polytope for the later LP instance. By moving among vertices of the polytope for the updated LP instance, the IRLS scheme keeps updating the solution of the RLLI. In theory, it is possible that the first iteration of IRLS produces non-integral values for all variables. The solution for each iteration of IRLS also depends on specific LP solver and underlying solving schemes. But in practice, the corner case aforementioned rarely happens in real experiments because LP relaxation provides a close-to-integral solution. Moreover, an incremental search step is performed after the IRLS solving. This means our IRLS with incremental search scheme can deliver insertion results at least as good as the greedy scheme. Meanwhile, the LP relaxation leads to polynomial time complexity within each iteration, which makes the runtime of the IRLS with incremental search scheme much more scalable than the optimal ILP approach.

IV. EXPERIMENTAL RESULTS

We have implemented the RLLI algorithms in C++ and all experiments are performed on a Linux machine with a 2.9GHz Intel(R) Core and 192 GB memory. Gurobi [24] is adopted as our ILP/LP solver. For RLL cost, the parameters are set as $\alpha = \beta = 1.0$ and $\gamma = 5.0$. The cost upper bound for weight computation is set as $CB = 100.0$. For limiting the search space of RLLCs for an SV, the upper bound of $r_{ix_{k+1}}$ and $r_{ix_k}$ is set as 20. For windows of via density control, the width is set as $\lambda = 20$ routing grids and the via density upper bound within a window is set as 40 for results in Table IV and V. The benchmark statistics are listed in Table III, where modules from OpenSparc T1 are synthesized with Design Compiler [25] and placed using Cadence SOC Encounter [26] with the standard cell utilization rate set as 0.7. The unidirectional routing results are generated using a state-of-the-art unidirectional router [27] and mapped onto a routing grid model. The routing density of the original unidirectional routing results is relatively low. Extra blockages are created on the M2 and M3 layer to represent dense routing cases in advanced technology nodes. Specifically, we take the sparse cases, i.e., the original unidirectional routing results, and add metal blockages every 1 out of 3 tracks (everywhere the routing grid is not occupied), which attempts to increase the routing utilization by 33% over the sparse cases. Our experiments end up with sparse and dense routing cases in Table IV and Table V, respectively. Our framework is based on a grid structure and supports partitioning of large benchmarks to make the runtime and memory more practical, e.g., test case “spare” is partitioned into 9 parts evenly in physical dimension. We further run our algorithm on each part separately for affordable memory and runtime. Our RLLI optimization framework takes the unidirectional routing patterns on the routing grid model as the input for the post-routing RLLI.

A. RLLC with Bounded Timing Impact

Our timing simulations build on the 10nm predictive technology setup, where wire resistance and capacitance are set by the ITRS roadmap [28] and via resistance is provided by our industrial collaborator. The driver is set as the INV_X1 from the NanGate 15nm library [29] scaled to 10nm dimensions with PTM models [30]. We assume the M2 and M3 layers share the same resistance/capacitance and the input routing solution has no detour. Under bounded local search space, we only consider RLL configurations shown in Fig. 7(a), where each RLLC is in the rectangular shape and covers at most 2 single vias. A rectangular RLLC covers exactly 4 vias. If an RLLC covers more than 2 single vias, that means the input routing solution has detour because the router should have selected the path containing redundant vias (less than 2) for short wirelength and a smaller number of single vias. For SPICE simulations of fan-out-4 delay impact in Fig. 7(b), we
Fig. 7: (a) A complete set of RLLC configurations assuming no detour in a routing solution, (b) timing impact evaluation.

As shown in Fig. 7(b), a wide range of delay impact, from -1.5% to +99%, is introduced for different configurations of RLLCs and the delay bound is set as 1%. The general observation from timing analysis is that more redundant vias in an RLLC will introduce larger secondary-path resistance and capacitance, which further leads to larger timing impact. Specifically, for one RLL configuration such as Case5, the timing degradations increase as the number of redundant metal grids increases as shown in Fig. 7(b). Moreover, for the same number of \( rm_2 \) and \( rm_3 \), the secondary-path resistance and capacitance increase monotonically from Case1 to Case5, which makes timing degradations monotonically increase as well. For Case4 and Case5, the timing degradations can be prohibitively large (9%). The delay impact bound is set as 1% in our experiments during RLLC generation, which means we strictly forbid the RLLCs with more than 1% delay impact. Although our timing simulations are preliminary, more accurate timing analysis can be easily introduced to the RLLI framework as long as an LUT of forbidden RLLC configurations is provided.

B. Comparisons on Different RLLI Schemes

In Table IV and V, we compare four RLLI schemes, including the greedy scheme, the ILP scheme, the IRLS without (w/o) and with (w/) incremental search scheme, in terms of solution qualities for the RLLI problem. The greedy scheme is adapted from [9], [10] with explicit considerations of SAV and via density constraints. The greedy scheme is the same as the incremental search in Algorithm 2 without the initial assignment results from IRLS. In Table IV and V, the insertion rate, denoted as “I.R.”, is defined as the number of SVs with an RLL inserted over the total number of SVs in the design. The “RLL#” is the total number of RLLs inserted. To quantify the redundant-via (RV) usage for each RLL inserted, we average the number RVs from the inserted RLLs over the total number of inserted RLLs, i.e. RV number per RLL, denoted as “R.p.R.”. “I” denotes the runtime. In general, better RLLI schemes should lead to larger “RLL#” and higher “I.R.” for yield improvement, less “R.p.R” for less resistance and smaller timing impact as shown in Fig. 7. Although the greedy scheme runs very fast due to linear time complexity, it has several drawbacks in terms of solution qualities for both sparse and dense routing cases. First, the “I.R.” is not consistent and highly depends on the test cases. For “ecc”, the “I.R.” is more than 98% (sparse) and 82% (dense) while for “alu”, the “I.R.” is less than 81% (sparse) and 61% (dense). Second, the “R.p.R” is higher than the IRLS w/ incremental search scheme by 13.1% (sparse) and 9.0% (dense) on average. This means the IRLS w/ incremental search can effectively select those RLLCs covering multiple SVs, which in general induces less timing impacts as shown in Fig. 7. Third, for the benchmark “top” the greedy scheme has around 10% less “I.R.” and ≥ 10% higher “R.p.R” compared to the IRLS w/ incremental search scheme for both sparse and dense routing cases.

The incremental search step is critical to result quality of Algorithm 2. In Table IV and V, more than 16% “I.R.” can be obtained with the incremental search on top of IRLS results with negligible runtime impact. This is because non-integral solutions may still exist from LP-relaxed instances, which means RLLCs are not assigned to some SVs after IRLS procedure. As RLLC generation and most of the assignment task has been performed after IRLS procedure, the incremental search takes linear time complexity, which is negligible compared to the runtime of RLLC generation and IRLS.

The ILP scheme maximizes a weighted sum to simultaneously improve the insertion rate and reduce the overall insertion cost. The best solution qualities are reported for the first three and four test cases for sparse and dense routing, respectively. However, the exponential time complexity of the ILP scheme leads to unaffordable runtime for large test cases. In particular, for the sparse routing case, ILP cannot generate a solution for “alu” under \( 10^5 \) seconds although “alu” has similar problem scale as “ctl”. The “I.R.” for “alu” is 14.83% and 7.55% less than that of “ctl” for greedy and IRLS w/ incremental search scheme, respectively. This means that the constraints for “alu” are very hard to resolve than those for “ctl”, which forbids ILP scheme reaching an optimal solution within \( 10^5 \) seconds. Actually, IRLS w/ incremental search scheme achieves competitive solution qualities as the “I.R.” is less than the ILP scheme by only 1% for the first three test cases. We achieve significant speed-up from the IRLS w/ incremental search scheme as the runtime is much more scalable and affordable across various test cases compared to the ILP approach. In addition, compared with the greedy scheme, the IRLS w/ incremental search scheme improves the “I.R.” by 17.8% and 16.3% with 18.2% and 15.3% increase in “RLL#” on average for sparse and dense routing, respectively.

In practice, the problem scale of later iterations of the IRLS procedure reduces dramatically, which makes it more scalable than the ILP scheme among test cases with different sizes shown in Table IV and V. Therefore, we argue that the IRLS w/ incremental search is the best scheme among the four schemes aforementioned since it has much better solution qualities than the greedy scheme and significant speed-up compared to the ILP approach.

As mentioned in Section II, via density constraints are critical for the RLLI problem because one RLL may cross...
This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2017.2651811, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems.

### TABLE III: Benchmark statistics

<table>
<thead>
<tr>
<th>Ckt</th>
<th>ecc</th>
<th>efc</th>
<th>ctl</th>
<th>alu</th>
<th>div</th>
<th>top</th>
<th>spec</th>
</tr>
</thead>
<tbody>
<tr>
<td>Grid Size</td>
<td>446 x 436</td>
<td>421 x 406</td>
<td>503 x 496</td>
<td>408 x 406</td>
<td>646 x 636</td>
<td>1179 x 1176</td>
<td>3969 x 3966</td>
</tr>
<tr>
<td>RLLC # per single via (sparse)</td>
<td>47.3</td>
<td>39.0</td>
<td>43.7</td>
<td>32.6</td>
<td>36.0</td>
<td>35.2</td>
<td>44.2</td>
</tr>
<tr>
<td>RLLC # per single via (dense)</td>
<td>11.2</td>
<td>9.6</td>
<td>11.0</td>
<td>8.2</td>
<td>8.9</td>
<td>8.6</td>
<td>10.7</td>
</tr>
<tr>
<td>RLLC generation time(s) (sparse)</td>
<td>4.5</td>
<td>4.9</td>
<td>6.5</td>
<td>6.3</td>
<td>13.0</td>
<td>63.5</td>
<td>144.2</td>
</tr>
<tr>
<td>RLLC generation time(s) (dense)</td>
<td>3.1</td>
<td>3.4</td>
<td>4.3</td>
<td>4.6</td>
<td>9.1</td>
<td>34.8</td>
<td>290.6</td>
</tr>
</tbody>
</table>

### TABLE IV: Result comparisons on sparse routing with different RLLI schemes

<table>
<thead>
<tr>
<th>Sparse</th>
<th>Greedy</th>
<th>0-1 ILP</th>
<th>IRLS w/o incremental search</th>
<th>IRLS w/ incremental search</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ckt</td>
<td>R.p.R</td>
<td>T(s)</td>
<td>R.p.R</td>
<td>T(s)</td>
</tr>
<tr>
<td>ecc</td>
<td>98.26</td>
<td>2542</td>
<td>4.6</td>
<td>207.5</td>
</tr>
<tr>
<td>efc</td>
<td>92.35</td>
<td>2799</td>
<td>4.9</td>
<td>355.7</td>
</tr>
<tr>
<td>ctl</td>
<td>95.23</td>
<td>3543</td>
<td>6.5</td>
<td>355.7</td>
</tr>
<tr>
<td>alu</td>
<td>80.40</td>
<td>3232</td>
<td>6.4</td>
<td>355.7</td>
</tr>
<tr>
<td>div</td>
<td>88.12</td>
<td>7103</td>
<td>14.0</td>
<td>1823.5</td>
</tr>
<tr>
<td>top</td>
<td>83.00</td>
<td>24705</td>
<td>49.8</td>
<td>1823.5</td>
</tr>
<tr>
<td>sparc</td>
<td>94.64</td>
<td>216143</td>
<td>445.4</td>
<td>1823.5</td>
</tr>
</tbody>
</table>

### TABLE V: Result comparisons on dense routing with different RLLI schemes

<table>
<thead>
<tr>
<th>Dense</th>
<th>Greedy</th>
<th>0-1 ILP</th>
<th>IRLS w/o incremental search</th>
<th>IRLS w/ incremental search</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ckt</td>
<td>R.p.R</td>
<td>T(s)</td>
<td>R.p.R</td>
<td>T(s)</td>
</tr>
<tr>
<td>ecc</td>
<td>82.28</td>
<td>2226</td>
<td>3.2</td>
<td>293.3</td>
</tr>
<tr>
<td>efc</td>
<td>73.91</td>
<td>2330</td>
<td>3.4</td>
<td>355.7</td>
</tr>
<tr>
<td>ctl</td>
<td>79.70</td>
<td>3119</td>
<td>4.3</td>
<td>355.7</td>
</tr>
<tr>
<td>alu</td>
<td>60.23</td>
<td>2527</td>
<td>4.7</td>
<td>355.7</td>
</tr>
<tr>
<td>div</td>
<td>69.27</td>
<td>5890</td>
<td>9.2</td>
<td>355.7</td>
</tr>
<tr>
<td>top</td>
<td>63.50</td>
<td>19800</td>
<td>35.1</td>
<td>355.7</td>
</tr>
<tr>
<td>sparc</td>
<td>77.34</td>
<td>187495</td>
<td>294.5</td>
<td>355.7</td>
</tr>
</tbody>
</table>

Fig. 8: Parameter analysis with benchmark “alu”, (a) degradations of insertion rates as the window density upper bound decreases with γ set as 5, (b) change of insertion rate when varying γ with window density upper bound set as 40.

multiple density windows. This not only introduces complicated conflicting constraints but also brings the opportunity to globally balance the via densities with efficient RLLI schemes. The strength of the IRLS w/ incremental search scheme is further demonstrated by the trade-off between the insertion rate and the window density upper bound in Fig. 8(a) for both the sparse and dense routing cases. The insertion rates are approximately the same for the greedy and IRLS w/ incremental search scheme when the upper bound is at 100 due to loose via density constraints. However, when the upper bound gradually decreases, the insertion rate rapidly decreases for the greedy scheme while the decrease for the IRLS scheme is much slower. Since the IRLS w/ incremental search scheme globally considers the via density constraints in each iteration, we observe better insertion rates when the upper bound is pushed to its lower limit.

For RLL cost, we assume metal patterns with the same length on M2 and M3 share the same resistance and capacitance. Thus, α and β are normalized to 1.0. In 10nm technology setup, the single via resistance will be larger than a piece of metal with minimum length (2 metal pitches in our experiments). Thus, we set via cost coefficient (γ) larger than α and β. Fig. 8(b) further demonstrates the impact on insertion rate from different via cost coefficient. In general, a larger via cost coefficient leads to lower cost for RLLCs with less number of redundant vias, i.e. RLLCs covering more than one single via. As shown in Fig. 4, these RLLCs usually cross more than one density window, which globally balances via densities and generates higher insertion rate. When γ increases, the insertion rate from the IRLS w/ incremental search scheme increases more slowly than that for the greedy scheme. The IRLS w/ incremental search scheme is still more favorable than the greedy scheme due to consistently larger insertion rates. Moreover, for the IRLS w/ incremental scheme, the insertion rate saturates as γ increases beyond 5 as shown in Fig. 8(b).

We empirically set γ as 5, which approximately denotes the starting point of saturation for the IRLS w/ incremental scheme in our experiment setup.
V. CONCLUSION AND DISCUSSION

In the paper, we propose the first global optimization engine on RLLI considering advanced manufacturing constraints on via patterns and bounded timing impact from RLLI. Our RLLI framework is independent of timing simulation setup as long as an LUT of forbidden RLLC configurations is given by the timing engine under bounded timing impact. With the net-based RLLC generation and pruning, we further propose the binary ILP formulation and the IRLS with incremental search scheme to obtain scalable solutions with negligible performance degradations. Our experimental results demonstrate that the IRLS with incremental search scheme achieves more consistent solution qualities compared to the greedy scheme and more scalable runtime compared to the ILP approach.

Global Timing Impact: We focus on the bounded timing impact on RLLI, which only controls the local timing degradations and expects negligible global impacts after RLLI. Global timing impacts can be captured by more comprehensive bounded timing LUT. An example is that the RLLIs close to the driver side could be more preferable than those close to the load side due to smaller timing impact. In general, accurate and global timing impact should be evaluated with parasitic extractions and timing simulations, which can be incorporated into our RLLI engine under industrial infrastructures.

Guidelines to handle complex DRC: In advanced technology nodes, complex design rules beyond those mentioned in Section II-B may be introduced for lower metal layers due to complicated design-for-manufacturing strategies, such as off-track metal tip-to-tip rules [22], [31], [32]. Complex design rules come from underlying patterning schemes and are foundry dependent. Our global optimization engine can be adapted to handle complex DRC in the following two ways.

- Formulate complex design rules into linear constraints and merge them into the IRLS scheme. Similar studies have been done for complex metal tip constraints [22], [31], [32].
- After the IRLS with incremental search scheme, we can determine the assignment of RLLC to each single via. If complex DRC handling is needed, we can further explicitly consider DRC when inserting each RLLC into the routing grid based on the assignment results, where only DRC-legal RLLCs will be physically inserted.

VI. ACKNOWLEDGEMENTS

The authors would like to thank Dr. Lars Liebmann from GLOBALFOUNDRIES for his helpful discussions on self-aligned via constraints. This work is supported in part by SRC, NSF, SPIE BACUS scholarship and university graduate continuing fellowship from University of Texas at Austin.

REFERENCES


VII. APPENDIX

To elaborate the potential usages of RLLI in advanced technology nodes, we compare traditional double-via insertion (DVI) and RLLI in terms of timing impact, random failure rate [10], routing resource usage and problem complexity. We assume off-track metal coverage for DVI, the width and length of which are set as 1 and 2 metal grids, respectively.

A. Timing Impact

We have performed the timing impact analysis and simulation for RLLI in Section III-A and Section IV-A. We evaluate the fan-out-4 delay with DVI using the same technology setup in Section IV-A. Fig. 9 compares the timing impact between DVI and various cases of RLLCs. DVI for one single via improves the fan-out-4 delay by 0.3% under our technology setup. In general, RLLI generates more redundant resistance and capacitance to form a local-loop structure comparing to DVI. Thus, RLLI introduces more timing impact than DVI except Case 1. This becomes the major motivation to control the timing degradations (within 1% in this work) during RLLI.

Fig. 9: RLLI vs DVI in terms of timing.

![RLLI vs DVI in terms of timing.](image)

![Timing Impact](image)

B. Random Failure Rate

We compare the random failure rate between DVI and RLLI to elaborate practical usages of RLLI. To simplify the analysis, we consider via failure rate while assuming metal patterns are free of opens or shorts [10]. Fig. 10(a) illustrates different cases of DVI and each case corresponds to a case of RLLC in Fig. 7(a). We assume the random failure probability of a single via is \( p \) and via failures are independent of each other. With probability calculation, we obtain the failure probability of each double-via and redundant local-loop insertion case as shown in Table VI [10]. If we vary \( p \) in the unit of “failure per billion (fpb)” [10], we can compute the failure rate ratio between RLLI and DVI for different cases as shown in Fig. 10(b). In general, the failure rate of a redundant local loop structure is 1 or 2 times larger than that of a double-via structure. Although RLL is not as robust as double via, RLL still provides better robustness than single via. Typically \( p \) is a very small value, which makes the values of \( (1 - (1 - p)^2) \) (for Case 1/3) and \( p \cdot (1 - (1 - p)^2) \) (for Case 2/4/5) much smaller than \( p \) itself. For instance, suppose \( p = 1 \text{fpb}, \) then \( (1 - (1 - p)^2) \approx 4 \cdot 10^{-9} \text{fpb} \) and \( p \cdot (1 - (1 - p)^2) \approx 3 \cdot 10^{-9} \text{fpb} \). Thus, RLLI is a promising candidate for yield enhancement of unidirectional routing.

![Random Failure Rate](image)

Fig. 10: (a) DVI in different cases and each case corresponds to a case of RLLC in Fig. 7(a), (b) failure rate ratio, i.e. RLLI over DVI across various via failure rate.

C. Routing Resource

We further compare routing resource usages between DVI and RLLI. For a specific case of DVI or RLLI, we quantify the routing resource as the total number of redundant metal \( (r_{m2} \text{ and } r_{m3}) \) and via grids. As shown in Table VI, the routing resource usage of RLLI varies from case to case, and RLLC covering more than one single via consumes less amount of routing resource. For instance, Case 5 uses much more routing resource than Case 1 for RLLI. In general, RLLI consumes

![Routing Resource](image)

TABLE VI: Comparisons between DVI and RLLI

<table>
<thead>
<tr>
<th>Case</th>
<th>DVI</th>
<th>RLLI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Case 1</td>
<td>(1 - (1 - p)^2)</td>
<td>(1 - (1 - p)^2) (p \cdot (1 - (1 - p)^2))</td>
</tr>
<tr>
<td>Case 2</td>
<td>(p^2)</td>
<td>(p \cdot (1 - (1 - p)))</td>
</tr>
<tr>
<td>Case 3</td>
<td>(1 - (1 - p)^2)</td>
<td>(1 - (1 - p)^2) (p \cdot (1 - (1 - p)))</td>
</tr>
<tr>
<td>Case 4</td>
<td>(p^2)</td>
<td>(p \cdot (1 - (1 - p)))</td>
</tr>
<tr>
<td>Case 5</td>
<td>(p^2)</td>
<td>(p \cdot (1 - (1 - p)))</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Failure rate</th>
<th>Routing resource</th>
</tr>
</thead>
<tbody>
<tr>
<td>DVI</td>
<td>RLLI</td>
</tr>
<tr>
<td>Case 1</td>
<td>(r_{m2} + r_{m3} + 2)</td>
</tr>
<tr>
<td>Case 2</td>
<td>(r_{m2} + r_{m3} + 3)</td>
</tr>
<tr>
<td>Case 3</td>
<td>(r_{m2} + r_{m3} + 3)</td>
</tr>
<tr>
<td>Case 4</td>
<td>(r_{m2} + 2 \cdot r_{m3} + 3)</td>
</tr>
<tr>
<td>Case 5</td>
<td>(2 \cdot r_{m2} + 2 \cdot r_{m3} + 3)</td>
</tr>
</tbody>
</table>
more routing resource than DVI due to the local-loop structure. RLLI and RVI are typically performed at the post-routing stage, routing resources (i.e. empty grids on metal and via layers) are given as input for RLLI/RVI engine to maximize the insertion rate. For metal layers where unidirectional routing is strictly enforced, RLLI is still a feasible candidate to improve manufacturing yield. Given limited routing resources for RLLI, it is important to perform global optimization and maximize insertion rate with bounding timing impact.

D. Problem Complexity

The RLLI problem is different from the conventional RVI problem although the ILP formulation looks similar. In Table III, we calculate the average number of RLLCs for each SV, denoted as “R.p.R”, to be 39 and 10 on average for sparse and dense routing cases, respectively, which are larger than the number of RV candidates available for each SV in the conventional RVI problem. Through our experiments, we find that the complexities of the RLLI problem make the fast techniques from the conventional RVI problem obsolete. Specifically, the pre-selection technique, i.e. selecting the conflict-free RLLC with minimum cost for an SV, breaks the optimality of the solution under tight via density constraints as an RLLC may cross multiple density windows as shown in Fig. 4. Independent component computation does not work well due to complicated conflicting constraints across several horizontal/vertical tracks. In our experiments, the largest independent component of each constructed conflict graph consists of around 99% and 85% of nodes within the entire graph for sparse and dense routing cases, respectively. Independent component computation is not effective for the RLLI problem and can not be directly applied under via density constraints. Thus, it is not incorporated into our optimization engine.

Xiaoqing Xu (S’15) received his B.S. degree in Microelectronics from Peking University, Beijing, China, in 2012. He is currently pursuing the Ph.D. degree in Electrical and Computer Engineering, University of Texas at Austin, under the supervision of Professor David Z. Pan. His research interests include robust standard cell design, design for manufacturability and physical design.

He received the Gliod Medal for ACM Design Automation Student Research Competition at IC-CAD 2016, the SRC best in session award in SRC TECHCON 2015, the SPIE BACUS Fellowship in 2016, the MCD Fellowship and University Graduate Continuing Fellowship from University of Texas at Austin in 2012 and 2016, respectively.

Yibo Lin received his B.S. degree in microelectronics from Shanghai Jiaotong University, Shanghai, China, in 2013. He is currently pursuing the Ph.D. degree at the Department of Electrical and Computer Engineering, University of Texas at Austin. His research interests include physical design and design for manufacturability.

He has received Franco Cerrina Memorial Best Student Paper Award at SPIE Advanced Lithography Conference 2016, and National Scholarship at Shanghai Jiaotong University in 2012. He has interned at IMEC, Cadence, and Oracle.

Meng Li (S’16) received his B.S. degree in Microelectronics from Peking University, Beijing, China in 2013. He is currently pursuing the Ph.D degree in Electrical and Computer Engineering, University of Texas at Austin, under the supervision of Prof. David Z. Pan. His research interests include hardware-oriented security, reliability and power grid simulation acceleration. He received Graduate Fellowship from UT Austin in 2013.

Jiaojiao Ou received her M.S. degree in Microelectronics from Peking University, Beijing, China, in 2013. She is currently pursuing the Ph.D. degree at the Department of Electrical and Computer Engineering, University of Texas, Austin, TX, USA. Her current research interests include design and design for manufacturability with emerging nanolithography.

Brian Cline (M’10) received the B.S. degree in electrical engineering from the University of Texas, Austin, in 2004, and the M.S. and Ph.D. degrees in electrical engineering from the University of Michigan, Ann Arbor, in 2006 and 2010, respectively. From 2006 to 2010, he was a Graduate Fellow with Semiconductor Research Corporation, Durham, NC. Currently, he is with the ARM Research Group, Austin, TX, as a Principal Research Engineer. His research interests include design technology co-optimization, low-power circuit design, variation-aware computer-aided design tool development, and very large scale integration design optimization for high-performance and low-power designs.

David Z. Pan (S’97–M’00–SM’06–F’14) received his B.S. degree from Peking University, and his M.S. and Ph.D. degrees from University of California, Los Angeles (UCLA). From 2000 to 2003, he was a Research Staff Member with IBM T. J. Watson Research Center. He is currently the Engineering Foundation Endowed Professor at the Department of Electrical and Computer Engineering, The University of Texas at Austin. He has published over 250 papers in refereed journals and conferences, and is the holder of 8 U.S. patents. His research interests include cross-layer nanometer IC design for manufacturability, reliability, security, new frontiers of physical design, and CAD for emerging technologies.

He has served as a Senior Associate Editor for ACM Transactions on Design Automation of Electronic Systems (TODAES), an Associate Editor for IEEE Design & Test, IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems (TCAD), IEEE Transactions on Very Large Scale Integration Systems (TVLSI), IEEE Transactions on Circuits and Systems PART I (TCAS-I), IEEE Transactions on Circuits and Systems PART II (TCAS-II), Science China Information Sciences (SCIS), Journal of Computer Science and Technology (JCST), etc. He has served as Program/General Chair of ISPD 2007/2008, TPC Chair for ASPDAC 2016, Vice Program Chair for ICCAD 2017, Tutorial Chair for DAC 2014, among others.

He has received a number of awards, including the SRC 2013 Technical Excellence Award, DAC Top 10 Author in Fifth Decade, DAC Prolific Author Award, ASPDAC Frequently Cited Author Award, 13 Best Paper Awards and several international CAD contest awards, Communications of the ACM Research Highlights (2014), ACM/SIGDA Outstanding New Faculty Award (2005), NSF CAREER Award (2007), SRC Inventor Recognition Award three times, IBM Faculty Award four times, UCLA Engineering Distinguished Young Alumnus Award (2009), and UT Austin RAISE Faculty Excellence Award (2014).