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-10 nm, unidirectional layout style has become the mainstream for lower metal layers with tight pitches. Conventional redundant via (RV) insertion for yield improvement has become obsolete because unidirectional routing patterns forbid off-track routing, i.e., wire bending, for the metal coverage of RVs. To enhance the yield, redundant local-loop insertion (RLLI) is a new way of inserting RVs 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.**

***Index Terms*—Linear programming (LP), redundant local loop (RLL), timing analysis, via density.**

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 (RV) [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 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. RV insertion (RVI) has been widely used in industry to improve the manufacturing yield of metal interconnections, where nonminimum width wire and wire bending are allowed. In 10 nm 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 RVs. As shown in Fig. 1(a), two RVs have been inserted for associated single vias (SVs). Metal-2 (M2) and metal-3 (M3) tracks are horizontal and vertical, respectively. The 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 RVs 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.

Bickford *et al.* [1] and Anderson *et al.* [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] 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,

Manuscript received July 22, 2016; revised October 9, 2016 and December 4, 2016; accepted December 28, 2016. Date of publication January 11, 2017; date of current version June 16, 2017. This work was supported in part by Semiconductor Research Corporation, in part by National Science Foundation, in part by SPIE BACUS Scholarship, and in part by the University Graduate Continuing Fellowship from the University of Texas at Austin. This paper was recommended by Associate Editor M. Ozdal.

X. Xu, Y. Lin, M. Li, J. Ou, and D. Z. Pan are with the Department of Electrical and Computer Engineering, University of Texas at Austin, Austin, TX 78731 USA (e-mail: xiaoqingxu.austin@utexas.edu).

B. Cline is with ARM Inc., Austin, TX 78735 USA.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCAD.2017.2651811

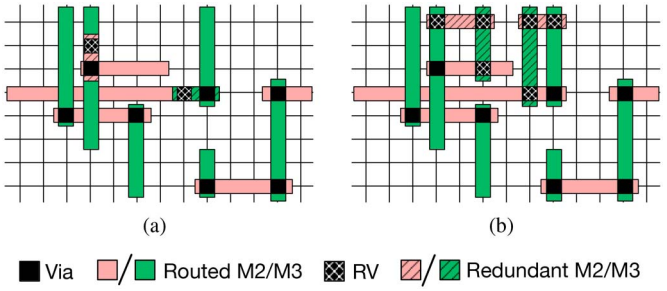


Fig. 1. (a) RVI with wire bending. (b) RLLI for unidirectional routing patterns.

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 (RV) candidates for neighboring SVs 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 (RLLC) may consist of multiple via and metal grids, which induces long-range conflict constraints among RLLCs. Meanwhile, RVs within one RLLC may cross multiple density windows. This means RVs within RLLCs 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 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 10 nm 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 SV. 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 controllable 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 SVs. 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.

1) We demonstrate bounding the timing impact due to RLLI using Elmore delay model analysis and SPICE simulations under the 10-nm predictive technology setup.
2) 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.
3) 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.
4) 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 this paper.

II. Preliminaries

A. Redundant Local Loop

For unidirectional routing patterns, RLLs are inserted to reduce the failure rate of within-loop SVs and one RLL could cover multiple SVs as shown in Fig. 2, where RLLs are inserted involving the M3, M2, and via-2 layers. RLL_1 consists of three redundant M3 grids, three redundant M2 grids, and one RV grid while RLL_2 includes five redundant M3 grids, three redundant M2 grids, and two RV grids. RLL_1 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

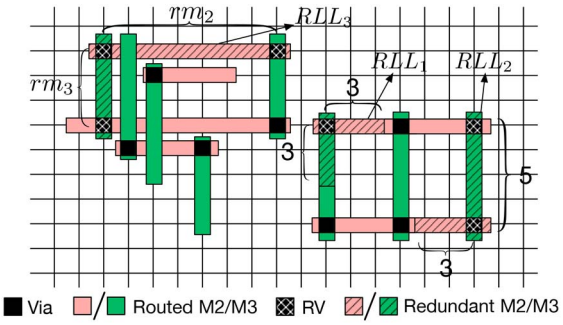


Fig. 2. RLL_1, RLL_2, and RLL_3 with configurations as $3 \times 3 \times 1$, $5 \times 3 \times 2$, and $rm_3 \times rm_2 \times 3$, respectively.

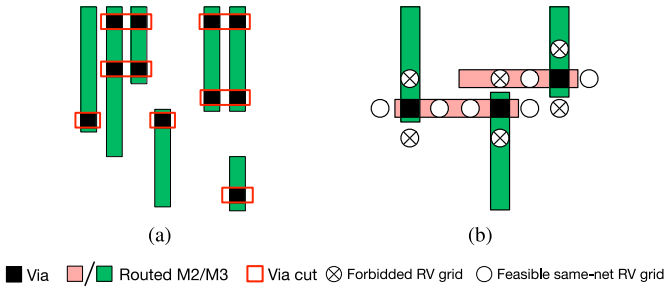


Fig. 3. (a) Via cuts for SAVs. (b) SAV design constraint.

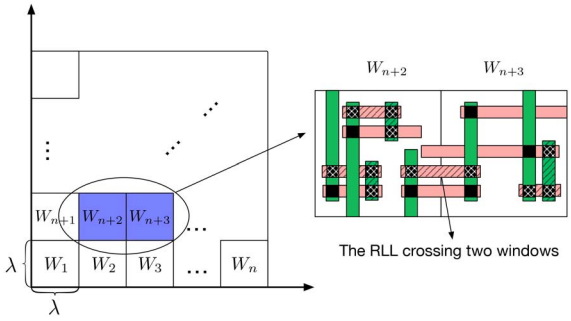


Fig. 4. Via density windows.

and the reasons are twofold. First, the RLLs with less number of RVs 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 $\mathrm{rm}_{x+1} \times \mathrm{rm}_x \times \mathrm{rv}_x$ 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 $\alpha \cdot \mathrm{rm}_{x+1} + \beta \cdot \mathrm{rm}_x + \gamma \cdot \mathrm{rv}_x$, where α, β, and γ 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 paper, we limit the local search space by bounding the number of redundant metal grids (i.e., rm_{x+1} or rm_x) by a predetermined parameter. Moreover, various RLLCs for an SV can be differentiated using the cost definition above, where α, β, and γ 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 10-nm node and beyond, the center-to-center spacing of lower via layers is in the sublithographic domain [7]. SAV patterning [11], [12] is a promising candidate for the sublithographic 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 sublithographic 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] and [12]. This paper considers basic SAV constraints in [12]. For an SV 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 RVs. 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.

Via density constraints are introduced primarily due to the chemical-mechanical-polishing and other manufacturing steps, which have varying outcomes depending on local layout densities [13]. If the number of RVs inserted within a density window is not well controlled, the violation of via density constraint could be detrimental to the yield of the design [14]. The windows for via density control are illustrated in Fig. 4. Specifically, a via layer is partitioned into a set of square regions. Each of the squares has a width of λ and the total number of vias, including SVs and RVs, within each window cannot exceed the preset upper bound. A possible scenario of the inserted RLLs is shown in detail for neighboring windows, i.e., W_{n+2} and W_{n+3}, in Fig. 4. With local search space of the RLLCs for an SV, an RLL may cross multiple density windows, such as W_{n+2} and W_{n+3}. This means that the optimal RLLI scheme should simultaneously consider all density windows to globally balance the via densities and maximize the number of RLLs inserted.

2) Metal-Pattern Constraints: Unidirectional metal patterns are manufacturing friendly in the sublithographic domain, which enables track-based coloring schemes for multiple patterning lithography. This paper addresses basic metal-pattern constraints including minimum metal tip-to-tip rule (one metal grid) for line-end control and minimum metal-length rule (two 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]. This paper 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 paper focuses on the RLLI with bounded timing impact for routed SVs 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 SV. Second, two RLLs inserted cannot occupy the same grid unless they belong to the same net. Third, the RLLI 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 RLLI 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 SVs 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 nontree 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 nontree 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]:

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

The close-form solution of Δdelay 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.

[1]The detailed principles of computation for Δdelay are given in [23].

TABLE I
Notations for RC Networks

C_d, C_l	the driving and loading capacitance
R_d, R_l	the driving and loading resistance
C_p, C_s	the capacitance on primary and secondary path
R_p, R_s	the resistance on primary and secondary path
C_{N1}, C_{N2}, C_{N3}	the effective capacitance on floating nodes

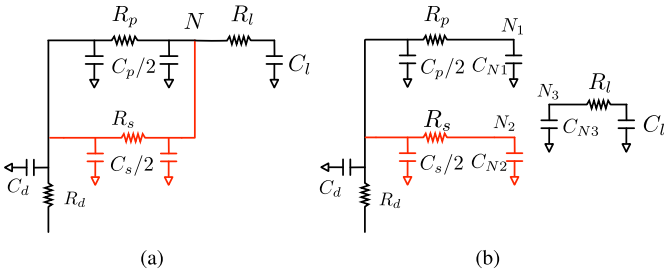


Fig. 5. Elmore delay model for one RLLC. (a) Simplified RC network. (b) Split RC network.

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 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 an SV. The region for valid RLLCs of an SV is subjected to a distance constraint, i.e., the horizontal and vertical distance from the SV to the farthest corner of the RLLC is bounded by a preset parameter. Thus, all valid RLLCs for

[2]Our RLLI scheme is independent of the timing engine as long as an LUT of forbidden RLLC configurations is given.

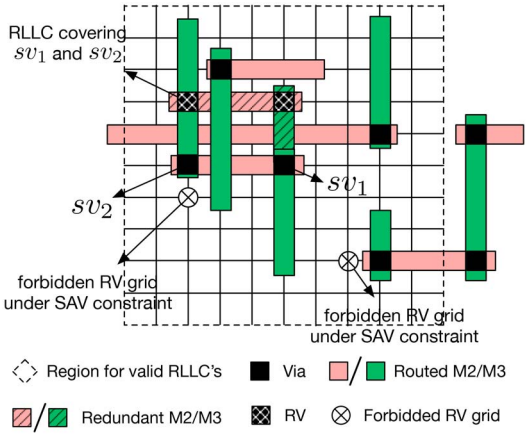


Fig. 6. RLLC generation for one SV.

an SV are bounded by a rectangular region. Fig. 6 illustrates an example of RLLC generation for an SV in a net with rm_3 and rm_2 bounded by 5. For sv_1, the region for valid RLLCs is shown with the rectangle with dashed lines. During the RLLC generation for sv_1, we scan the rectangular region and skip forbidden RV grids and metal grids occupied by a different net for valid RLLCs of sv_1. For example, due to the SAV constraints, forbidden RV grids are shown in Fig. 6 and RLLCs occupying those via grids are invalid. One valid RLLC for sv_1 is further shown in Fig. 6, which covers two SVs, i.e., sv_1 and sv_2. It shall be noted that, during the sequential RLLC generation, this RLLC will be generated twice for sv_1 and sv_2. In general, the sequential RLLC generation leads to duplicates, i.e., equivalent RLLCs, for SVs belonging to the same net. We define the equivalence of two RLLCs as follows.

Definition 2 (Equivalence of RLLCs): An RLLC is defined to be equivalent to another RLLC when they share the same metal and via grids associated with the same net.

We propose net-based RLLC generation and pruning technique to remove duplicates and achieve a compact set of RLLCs for each SV. The details of the RLLC generation and pruning scheme are shown in Algorithm 1. The RLLCs are generated in a net-by-net manner, which is explained from lines 3 to 18 of Algorithm 1. For each net, we define RLLC_{net} in line 4 to store a compact set of RLLCs belonging to that net. Then, the SVs within the net are traversed from lines 5 to 9. We enumerate the RLLCs for SV v as the set {rllc} in line 6. In line 7, we further remove illegal RLLC configurations based on the timing analysis in Section III-A and the design rule checker (DRC). Notable design rules included here are the minimum metal tip-to-tip rule (one metal grid), minimum metal-length rule (two metal grids) and SAV design rules in Section II-B. In advanced technology nodes, complex metal and via pattern rules are technology dependent, which can be easily incorporated into our optimization engine by inventing technology-specific DRC. The enumerated set {rllc} will be further combined into RLLC_{net} in line 8. In particular, since RLLC_{net} is a set data structure, it will automatically remove

Algorithm 1 RLLC Generation and Duplicate Removal

Input: Routed patterns for nets ({*nets*}), a lookup table (*LUT*) of forbidden RLLCs from bounding timing analysis and a design rule checker (*DRC*);

Output: A set of RLLC vectors indexed by single vias ($\{RLLC_{v_i}\}$) and a compact vector of RLLCs (*RLLC*);

1: Define *RLLC* as a compact vector of RLLCs for all SVs;
2: Define $\{RLLC_{v_i}\}$ as the set of RLLC vectors indexed by single vias;
3: **for** each *net* in {*nets*} **do**;
4: Define $RLLC_{net}$ as the set of RLLCs for *net*;
5: **for** each single via v_i in *net* **do**;
6: Enumerate RLLCs for v_i as set {*rllc*};
7: Remove illegal RLLCs in {*rllc*} with bounded timing (*LUT*) and design rule checker (*DRC*);
8: $RLLC_{net} = RLLC_{net} \cup \{rllc\}$;
9: **end for**
10: **for** each *llc* in $RLLC_{net}$ **do**;
11: Index *llc* with the size of *RLLC*;
12: Push *llc* to *RLLC*;
13: **end for**
14: **for** each single via v_i in *net* **do**;
15: Define $RLLC_{v_i}$ as the RLLC vector for v_i;
16: Select $RLLC_{v_i}$ from $RLLC_{net}$;
17: **end for**
18: **end for**
19: **for** $RLLC_{v_i}$ indexed by SV v_i **do**;
20: Prune redundant candidates in $RLLC_{v_i}$;
21: **end for**
22: Return $\{RLLC_{v_i}\}$ and *RLLC*;

duplicates according to the definition of the RLLC equivalence in Definition 2. In lines 10–13, we index the RLLCs generated for current net based on the size of compact RLLC vector RLLC, which are further merged into RLLC. We need a compact vector of RLLCs because only one binary variable is generated for an RLLC covering multiple SVs in mathematical formulations. From lines 14 to 17, we obtain a vector to index the RLLCs for each SV in the net. The RLLC vector for each SV (RLLC_{v_i}) 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 SV.

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 SVs per net times the maximum number of RLLCs per SV. Thus, the set (RLLC_{net}) construction time (from lines 4 to 9 in Algorithm 1) is $O(M \cdot R \cdot \log(M \cdot R))$. In lines 10–13 and 14–17, merging RLLC_{net} to RLLC and RLLC selection for each SV both take $O(M \cdot R)$ time. The enumeration loop in lines 3–18 takes $O(N \cdot M \cdot R \cdot \log(M \cdot R))$ time. In line 20, we sort the RLLCs

TABLE II
NOTATIONS FOR RLLI

N	the number of nets
M	the maximum number of SVs for one net
R	the maximum number of RLLCs for each SV
L	the total number of RLLCs
SV	the pre-defined set of SVs
sv_j	j^{th} SV
llc_i	i^{th} RLLC
c_i	the cost for llc_i
n_i	the number of single vias that llc_i covers
x_i	the binary variable for llc_i
X_j	the variable set for the RLLCs covering sv_j
G_k	the k^{th} variable set for RLLCs occupying the same grid
SA_k	the k^{th} variable set for RLLCs occupying conflicting SAV grids
X, G, SA	sets of variable sets for $\{X_i\}$, $\{G_k\}$ and $\{SA_k\}$, respectively
W_k	the k^{th} density window
v_{ik}	the number of RVs of llc_i in W_k
DB_k	via density upper bound for W_k
W	the set of density windows $\{W_k\}$

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 SVs is bounded by $O(N \cdot M)$. The pruning technique in lines 19–21 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 SVs and each SV has approximately the same number of RLLCs, i.e., $M \simeq |\text{SV}|/N$ and $R \simeq L/|\text{SV}|$. Then, the time complexity of Algorithm 1 reduces to $O(N \cdot |\text{SV}|/N \cdot L/|\text{SV}|) \cdot \log(|\text{SV}|/N \cdot L/|\text{SV}|)$, i.e., $O(L \cdot \log(L/N))$.

The last step in Algorithm 1 from lines 19 to 21 is to further prune RLLCs without loss of optimality. For RLLC_{v_i} associated with an SV 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 SV. 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.[3]

C. ILP Formulation

With the RLLCs generated for each SV, Problem 1 becomes a general assignment problem to cover as many SVs 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 ith RLLC llc_i is selected to cover associated SVs. In particular, only one binary variable is generated for an RLLC covering more than one SV. Then, Problem 1

$$\max \ \text{CB} \cdot \sum_{x_i} n_i \cdot x_i - \sum_{x_i} c_i \cdot x_i \tag{1}$$

$$\text{s.t.} \ \sum_{x_i \in X_j} x_i \leq 1 \quad \forall X_j \in X \tag{2}$$

$$\sum_{x_i \in A} x_i \leq 1 \quad \forall A \in G \cup SA \tag{3}$$

$$\sum_{\text{llc}_i \in W_k} v_{ik} \cdot x_i \leq \text{DB}_k \quad \forall W_k \in W \tag{4}$$

$$x_i \in \{0, 1\} \quad \forall x_i \in X_j \in X \tag{5}$$

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 $(\text{CB} \cdot \sum_{x_i} n_i \cdot x_i)$ 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 SVs. The selection of that particular RLLC can cover n_i SVs. The term $(-\sum_{x_i} c_i \cdot x_i)$ is the second objective, which reduces the overall cost of inserted RLLs. The two terms are balanced using the parameter CB, where $\text{CB} > \max_i c_i$. 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.

1) At most one RLLC is assigned to each SV. 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 (2).
2) 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 (3).[4]
3) 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 llc_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 (4).

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 (4), which generates much better solution qualities than a simple greedy scheme.

[3]This pruning technique could be restrictive under long-range conflict constraints and via density constraints, which may limit the effectiveness for practical test cases.

[4]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 (5) 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 (2) and conflicting constraints in (3) 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 (2) and (3) 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 and strong invalid inequality in the following equations:

$$x_i + x_j \leq 1 \quad \forall i,\ j \in [1, k] \tag{6}$$

$$\sum_{i=1}^{k} x_i \leq 1. \tag{7}$$

The selection constraints in (2) and conflict constraints in (3) are generated for each SV 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 (7), play as cutting planes to bound the LP solution to close to integral [19]. The via density constraints in (4) 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.

Nonintegral solutions generally exist from LP-relaxed instance even with strong valid inequalities (2), (3) and capacity constraints (4) for a constrained multiple knapsack problem. Thus, randomized rounding schemes are adopted in [19] and [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 noninteger 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 SVs 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.

Algorithm 2 IRLS With Incremental Search

Input: An instance of the RLLI problem (*RLLI*) and the single via set (*SV*);
Output: The assignment (F) of RLL for each SV;

```
1: Define F as the assignment of RLL for each single via;
2: while True do;
3:     Find the optimal solution X_opt with LP(RLLI);
4:     Define A_0 = ∅ as the integral assignment with 0's;
5:     Define A_1 = ∅ as the integral assignment with 1's;
6:     for each x_i = 1 in X_opt do;
7:         Add x_i = 1 to A_1;
8:     end for
9:     if A_1 = ∅ then
10:        Break;
11:    end if
12:    for each conflict constraint C in (2) and (3) do;
13:        Define X_C as the variable set in C;
14:        if X_C ∩ A_1 ≠ ∅ then;
15:            Add X_C − A_1 to A_0 with assignment 0;
16:            Remove C and X_C from RLLI;
17:        end if
18:    end for
19:    for each window density constraint C in (4) do;
20:        Update DB_k with assignment A_1 ∪ A_0;
21:    end for
22:    Update the objective (1) with assignment A_1 ∪ A_0;
23:    F = F ∪ A_1 ∪ A_0;
24: end while
25: for each single via v_i in SV do;
26:    if no RLLC assignment to v_i in F then;
27:        Obtain the set of RLLCs (RLLC_vi) for v_i;
28:        Sort RLLC_vi in the increasing order of cost;
29:        Traverse RLLC_vi for the first legal candidate compactable with F and add it to F;
30:    end if
31: end for
32: Return F;
```

The details of the IRLS scheme for Problem 1 are shown in Algorithm 2. The main loop for IRLS is from lines 2 to 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 lines 6 to 8, we obtain the integral assignment A_1 from X_{opt}, where only variables with solutions as 1 are collected into A_1. From lines 12 to 18, we iterate through the conflict constraints for constraints (2) and (3). 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 lines 19 to 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 lines 9–11. An incremental search step is explained in lines 25–31. During the incremental search, for each SV 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 nonintegral values for all variables. The solution for each iteration of IRLS also depends on specific ILP 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.

For the IRLS with incremental search scheme, each iteration of LP solving simultaneously considers all conflict constraints and via density constraints. With an incremental search step, it gives much better solution qualities than the simple 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 2.9 GHz Intel 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 rm_{x+1} and rm_x 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 Tables 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 one out of three 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 Tables IV and 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 "sparc" is partitioned into nine 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 10-nm 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 15-nm library [29] scaled to 10-nm 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 two SVs. A rectangular RLLC covers exactly four vias. If an RLLC covers more than two SVs, that means the input routing solution has detour because the router should have selected the path containing RVs (less than 2) for short wirelength and a smaller number of SVs. For SPICE simulations of fan-out-4 delay impact in Fig. 7(b), we adopt the pi model for each wire segment and via leading to more accurate RC network for

TABLE III
BENCHMARK STATISTICS

Ckt	ecc	efc	ctl	alu	div	top	sparc
SV#	4013	4619	5873	6683	12878	48847	360239
Grid Size	446 x 436	421 x 406	503 x 496	408 x 406	646 x 636	1179 x 1176	3969 x 3966
RLLC # per single via (sparse)	47.3	39.0	43.7	32.6	36.0	35.2	44.2
RLLC # per single via (dense)	11.2	9.6	11.0	8.2	8.9	8.6	10.7
RLLC generation time(s) (sparse)	4.5	4.9	6.5	6.3	13.0	49.4	443.2
RLLC generation time(s) (dense)	3.1	3.4	4.3	4.6	9.1	34.8	290.6

TABLE IV
RESULT COMPARISONS ON SPARSE ROUTING WITH DIFFERENT RLLI SCHEMES

Sparse	Greedy				0-1 ILP				IRLS w/o incremental search				IRLS w/ incremental search			
Ckt	I.R.(%)	RLL#	R.p.R	T(s)	I.R.(%)	RLL#	R.p.R	T(s)	I.R.(%)	RLL#	R.p.R	T(s)	I.R.(%)	RLL#	R.p.R	T(s)
ecc	98.26	2542	2.45	4.6	99.85	2119	2.11	207.5	88.21	1874	2.11	53.5	99.75	2136	2.13	53.5
efc	92.35	2799	2.45	4.9	99.06	2465	2.11	500.9	82.37	2056	2.12	61.3	98.00	2476	2.14	61.3
ctl	95.23	3543	2.42	6.5	99.66	3055	2.08	355.7	84.62	2598	2.09	81.1	99.05	3059	2.10	81.1
alu	80.40	3232	2.34	6.4	N/A	N/A	N/A	$> 10^5$	66.68	2271	2.04	82.5	91.50	3221	2.10	82.5
div	88.12	7103	2.40	13.0	98.11	6612	2.09	9285.4	78.42	5258	2.08	166.1	96.12	6579	2.12	166.2
top	83.00	24705	2.36	49.8	N/A	N/A	N/A	$> 10^5$	72.53	17902	2.02	844.2	93.08	23680	2.08	844.4
sparc	94.64	216143	2.42	445.4	N/A	N/A	N/A	$> 10^5$	82.74	155836	2.09	7218.7	98.17	188394	2.12	7219.7
Avg.	0.935	1.133	1.131	0.062					0.822	0.818	0.977	1.000	1.000	1.000	1.000	1.000

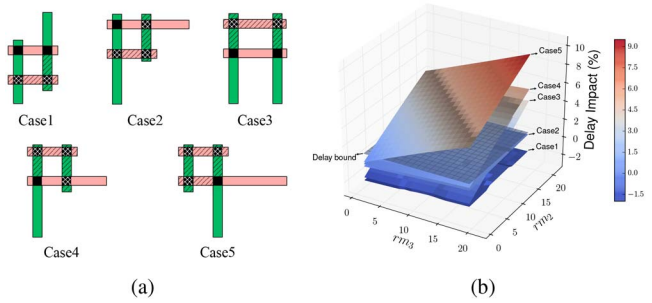


Fig. 7. (a) Complete set of RLLC configurations assuming no detour in a routing solution. (b) Timing impact evaluation.

delay simulations than that in Fig. 5. The primary path consists of M2 and M3 wire segment both in 4-um length [10].

As shown in Fig. 7(b), a wide range of delay impact, from -1.5% to $+9\%$, is introduced for different configurations of RLLCs and the delay bound is set as 1%. The general observation from timing analysis is that more RVs 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 case 5, 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 cases 1 to 5, which makes timing degradations monotonically increase as well. For cases 4 and 5, 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.

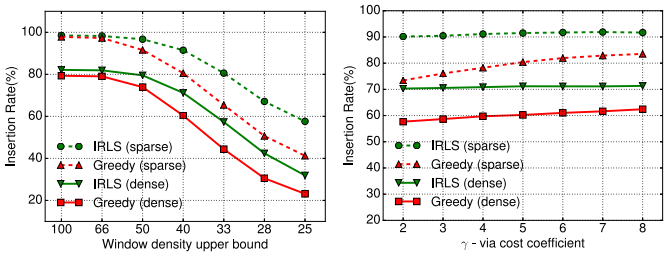
B. Comparisons on Different RLLI Schemes

In Tables 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] and [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 Tables 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 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." "T" 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 $\geq$ 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 Tables 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 nonintegral 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.

TABLE V
RESULT COMPARISONS ON DENSE ROUTING WITH DIFFERENT RLLI SCHEMES

Dense	Greedy				0-1 ILP				IRLS w/o incremental search				IRLS w/ incremental search			
Ckt	I.R.(%)	RLL#	R.p.R	T(s)	I.R.(%)	RLL#	R.p.R	T(s)	I.R.(%)	RLL#	R.p.R	T(s)	I.R.(%)	RLL#	R.p.R	T(s)
ecc	82.28	2226	2.52	3.2	85.47	2010	2.29	29.3	76.70	1810	2.30	11.1	85.12	2015	2.30	11.1
efc	73.91	2330	2.51	3.4	82.69	2272	2.29	135.0	67.34	1859	2.30	13.1	81.11	2258	2.31	13.1
ctl	79.70	3119	2.50	4.3	85.71	2921	2.28	174.5	72.94	2514	2.30	17.5	84.62	2915	2.30	17.5
alu	60.23	2527	2.41	4.7	74.52	2729	2.18	11469.6	53.05	1947	2.18	18.5	71.18	2674	2.22	18.5
div	69.27	5890	2.49	9.2	N/A	N/A	N/A	$> 10^5$	64.81	4795	2.26	35.3	78.31	5868	2.28	35.3
top	63.50	19800	2.43	35.1	N/A	N/A	N/A	$> 10^5$	59.85	15941	2.17	150.3	74.15	20257	2.21	150.4
sparc	77.34	187495	2.51	294.5	N/A	N/A	N/A	$> 10^5$	71.40	150731	2.29	1389.8	82.64	175980	2.31	1390.5
Avg.	0.907	1.054	1.090	0.217					0.837	0.847	0.992	1.000	1.000	1.000	1.000	1.000



(a) (b)

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.

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 s 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 s. 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 Tables 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 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 tradeoff between the insertion rate and the window density upper bound in Fig. 8(a) for both 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 10-nm technology setup, the SV resistance will be larger than a piece of metal with minimum length (two 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 RVs, i.e., RLLCs covering more than one SV. 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

In this 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.

A. 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.

B. 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.

1) 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].
2) After the IRLS with incremental search scheme, we can determine the assignment of RLLC to each SV. 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.

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 one and two 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 SV 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 paper) during RLLI.

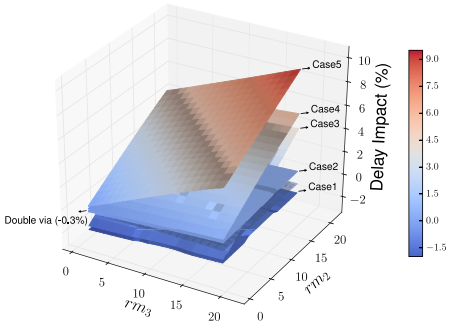


Fig. 9. RLLI versus DVI in terms of timing.

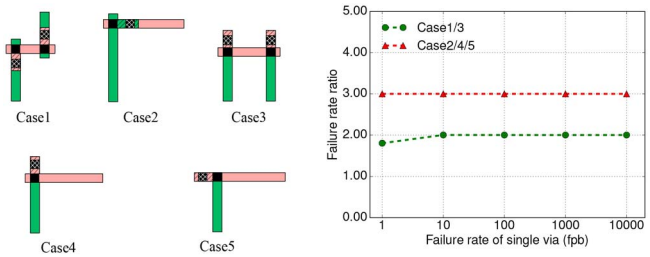


Fig. 10. (a) DVI in different cases and each case corresponds to a case of RLLC in Fig. 7(a). (b) Failure rate ratio, i.e., RLLI over DVI across various via failure rate.

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 an SV is p and via failures are independent of each other. With probability calculation, we obtain the failure probability of each double-via and RLLI 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 an RLL structure is one or two times larger than that of a double-via structure. Although RLL is not as robust as double via, RLL still provides better robustness than SV. Typically p is a very small value, which makes the values of $(1-(1-p)^2)^2$ (for case 1/3) and $p \cdot (1-(1-p)^3)$ (for case 2/4/5) much smaller than p itself. For instance, suppose $p = 1$fpb, then $(1-(1-p)^2)^2 \simeq 4 \cdot 10^{-9}$ fpb and $p \cdot (1-(1-p)^3) \simeq 3 \cdot 10^{-9}$ fpb. Thus, RLLI is a promising candidate for yield enhancement of unidirectional routing.

TABLE VI
COMPARISONS BETWEEN DVI AND RLLI

	Failure rate		Routing resource	
	DVI	RLLI	DVI	RLLI
Case 1	$1-(1-p^2)^2$	$(1-(1-p)^2)^2$	6	rm_2+rm_3+2
Case 2	p^2	$p\cdot(1-(1-p)^3)$	3	rm_2+rm_3+3
Case 3	$1-(1-p^2)^2$	$(1-(1-p)^2)^2$	6	$rm_2+2\cdot rm_3+2$
Case 4	p^2	$p\cdot(1-(1-p)^3)$	3	$rm_2+2\cdot rm_3+3$
Case 5	p^2	$p\cdot(1-(1-p)^3)$	3	$2\cdot rm_2+2\cdot rm_3+3$

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 (rm_2 and rm_3) 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 SV consumes less amount of routing resource. For instance, case 5 uses much more routing resource than case 1 for RLLI. In general, RLLI consumes 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 preselection 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 cannot be directly applied under via density constraints. Thus, it is not incorporated into our optimization engine.

ACKNOWLEDGMENT

The authors would like to thank Dr. L. Liebmann from GLOBALFOUNDRIES for his helpful discussions on SAV constraints.

REFERENCES

[1] J. Bickford *et al.*, "Yield improvement by local wiring redundancy," in *Proc. IEEE Int. Symp. Qual. Electron. Design (ISQED)*, San Jose, CA, USA, 2006, pp. 473–478.

[2] K.-Y. Lee and T.-C. Wang, "Post-routing redundant via insertion for yield/reliability improvement," in *Proc. IEEE/ACM Asia South Pac. Design Autom. Conf. (ASPDAC)*, Yokohama, Japan, 2006, pp. 303–308.

[3] F. Luo, Y. Jia, and W. W.-M. Dai, "Yield-preferred via insertion based on novel geotopological technology," in *Proc. IEEE/ACM Asia South Pac. Design Autom. Conf. (ASPDAC)*, 2006, pp. 730–735.

[4] A. B. Kahng, B. Liu, and I. I. Mandoiu, "Non-tree routing for reliability and yield improvement," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, San Jose, CA, USA, 2002, pp. 260–266.

[5] F.-Y. Chang, R.-S. Tsay, and W.-K. Mak, "How to consider shorts and guarantee yield rate improvement for redundant wire insertion," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, San Jose, CA, USA, 2009, pp. 33–38.

[6] M.-K. Hsu *et al.*, "Design and manufacturing process co-optimization in nano-technology," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, San Jose, CA, USA, 2014, pp. 574–581.

[7] L. Liebmann, A. Chu, and P. Gutwin, "The daunting complexity of scaling to 7nm without EUV: Pushing DTCO to the extreme," in *Proc. SPIE*, San Jose, CA, USA, 2015, Art. no. 942702.

[8] J. Ryckaert *et al.*, "DTCO at N7 and beyond: Patterning and electrical compromises and opportunities," in *Proc. SPIE*, San Jose, CA, USA, 2015, Art. no. 94270C.

[9] B. A. Anderson *et al.*, "Redundant micro-loop structure for use in an integrated circuit physical design process and method of forming the same," U.S. Patent 8 234 594, Jul. 31, 2012.

[10] W. Huang *et al.*, "Local loops for robust inter-layer routing at sub-20 nm nodes," in *Proc. SPIE*, 2012, Art. no. 83270D.

[11] J. C. Arnold, S. D. Burns, S. K. Kanakasabapathy, and Y. Yin, "Self aligning via patterning," U.S. Patent 8 298 943, Oct. 30, 2012.

[12] M. L. Rieger and V. Moroz, "Self-aligned via interconnect using relaxed patterning exposure," U.S. Patent 8 813 012, Aug. 19, 2014.

[13] A. B. Kahng, "Research directions for coevolution of rules and routers," in *Proc. ACM Int. Symp. Phys. Design (ISPD)*, Monterey, CA, USA, 2003, pp. 122–125.

[14] K.-Y. Lee, T.-C. Wang, and K.-Y. Chao, "Post-routing redundant via insertion and line end extension with via density consideration," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, San Jose, CA, USA, 2006, pp. 633–640.

[15] C.-K. Lei, P.-Y. Chiang, and Y.-M. Lee, "Post-routing redundant via insertion with wire spreading capability," in *Proc. IEEE/ACM Asia South Pac. Design Autom. Conf. (ASPDAC)*, Yokohama, Japan, 2009, pp. 468–473.

[16] K.-Y. Lee, C.-K. Koh, T.-C. Wang, and K.-Y. Chao, "Optimal post-routing redundant via insertion," in *Proc. ACM Int. Symp. Phys. Design (ISPD)*, Portland, OR, USA, 2008, pp. 111–117.

[17] K.-Y. Lee, S.-T. Lin, and T.-C. Wang, "Redundant via insertion with wire bending," in *Proc. ACM Int. Symp. Phys. Design (ISPD)*, San Diego, CA, USA, 2009, pp. 123–130.

[18] S.-T. Lin, K.-Y. Lee, T.-C. Wang, C.-K. Koh, and K.-Y. Chao, "Simultaneous redundant via insertion and line end extension for yield optimization," in *Proc. IEEE/ACM Asia South Pac. Design Autom. Conf. (ASPDAC)*, Yokohama, Japan, 2011, pp. 633–638.

[19] R. Kay and R. A. Rutenbar, "Wire packing: A strong formulation of crosstalk-aware chip-level track/layer assignment with an efficient integer programming solution," in *Proc. ACM Int. Symp. Phys. Design (ISPD)*, San Diego, CA, USA, 2000, pp. 61–68.

[20] G. Dahl and N. Foldnes, "LP based heuristics for the multiple knapsack problem with assignment restrictions," *Ann. Oper. Res.*, vol. 146, no. 1, pp. 91–104, 2006.

[21] G. Luk-Pat *et al.*, "Avoiding wafer-print artifacts in spacer is dielectric (SID) patterning," in *Proc. SPIE*, San Jose, CA, USA, 2013, Art. no. 868312.

[22] X. Xu, B. Cline, G. Yeric, B. Yu, and D. Z. Pan, "Self-aligned double patterning aware pin access and standard cell layout co-optimization," in *Proc. ACM Int. Symp. Phys. Design (ISPD)*, Petaluma, CA, USA, 2014, pp. 101–108.

[23] T.-M. Lin and C. A. Mead, "Signal delay in general RC networks," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 3, no. 4, pp. 331–349, Oct. 1984.

[24] Gurobi. (2014). *GUROBI*. [Online]. Available: http://www.gurobi.com/html/academic.html

[25] Synopsys. (2012). *Synopsys Design Compiler*. [Online]. Available: http://www.synopsys.com
[26] Cadence. (2012). *Cadence SOC Encounter*. [Online]. Available: http://www.cadence.com/
[27] X. Xu, B. Yu, J.-R. Gao, C.-L. Hsu, and D. Z. Pan, "PARR: Pin access planning and regular routing for self-aligned double patterning," in *Proc. ACM/IEEE Design Autom. Conf. (DAC)*, San Francisco, CA, USA, 2015, pp. 1–6.
[28] *ITRS*. Accessed on Nov. 20, 2014. [Online]. Available: http://www.itrs.net
[29] NanGate. (2014). *NanGate FreePDK15 Open Cell Library*. [Online]. Available: http://www.nangate.com/?page_id=2328
[30] *Predictive Technology Model Ver. 2.1*. Accessed on Feb. 10, 2016. [Online]. Available: http://ptm.asu.edu
[31] Y. Du, H. Zhang, M. D. F. Wong, and K.-Y. Chao, "Hybrid lithography optimization with e-beam and immersion processes for 16nm 1D gridded design," in *Proc. IEEE/ACM Asia South Pac. Design Autom. Conf. (ASPDAC)*, Sydney, NSW, Australia, 2012, pp. 707–712.
[32] Y. Ding, C. Chu, and W.-K. Mak, "Throughput optimization for SADP and e-beam based manufacturing of 1D layout," in *Proc. ACM/IEEE Design Autom. Conf. (DAC)*, San Francisco, CA, USA, 2014, pp. 1–6.

Xiaoqing Xu (S'15) received the 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 with the University of Texas at Austin, Austin, TX, USA, under the supervision of Prof. D. Z. Pan.

His current research interests include robust standard cell design, design for manufacturability, and physical design.

Mr. Xu was a recipient of the Gold Medal for ACM Design Automation Student Research Competition at ICCAD 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 the University of Texas at Austin, in 2012 and 2016, respectively.

Yibo Lin received the B.S. degree in microelectronics from Shanghai Jiaotong University, Shanghai, China, in 2013. He is currently pursuing the Ph.D. degree with the Department of Electrical and Computer Engineering, University of Texas at Austin, Austin, TX, USA.

He was an Intern with IMEC, Leuven, Belgium, Cadence, San Jose, CA, USA, and Oracle, Redwood City, CA, USA. His current research interests include physical design and design for manufacturability.

Mr. Lin was a recipient of the Franco Cerrina Memorial Best Student Paper Award at the SPIE Advanced Lithography Conference 2016, and the National Scholarship at Shanghai Jiaotong University in 2012.

Meng Li (S'16) received the 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 with the University of Texas at Austin (UT Austin), Austin, TX, USA, under the supervision of Prof. D. Z. Pan.

His current research interests include hardware-oriented security, reliability and power grid simulation acceleration.

Mr. Li was a recipient of the Graduate Fellowship from UT Austin, in 2013.

Jiaojiao Ou received the M.S. degree in microelectronics from Peking University, Beijing, China, in 2013. She is currently pursuing the Ph.D. degree with the Department of Electrical and Computer Engineering, University of Texas at Austin, Austin, TX, USA.

Her current research interests include physical design and design for manufacturability with emerging nanolithography.

Brian Cline (M'10) received the B.S. degree from the University of Texas at Austin, Austin, TX, USA, in 2004, and the M.S. and Ph.D. degrees from the University of Michigan, Ann Arbor, MI, USA, in 2006 and 2010, respectively, all in electrical engineering.

He was a Graduate Fellow with Semiconductor Research Corporation, Durham, NC, USA, from 2006 to 2010. He is currently a Principal Research Engineer with ARM Research Group, Austin. His current 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 the B.S. degree from Peking University, Beijing, China, and the M.S. and Ph.D. degrees from University of California at Los Angeles (UCLA), Los Angeles, CA, USA.

He was a Research Staff Member with IBM T. J. Watson Research Center, Yorktown Heights, NY, USA, from 2000 to 2003. He is currently the Engineering Foundation Endowed Professor with the Department of Electrical and Computer Engineering, University of Texas at Austin, Austin, TX, USA. He has published over 250 papers in refereed journals and conferences and holds eight U.S. patents. His current research interests include cross-layer nanometer IC design for manufacturability, reliability, security, new frontiers of physical design, and computer-aided design for emerging technologies.

Dr. Pan was a recipient of number of awards, including the SRC 2013 Technical Excellence Award, the Design Automation Conference (DAC) Top 10 Author in Fifth Decade, the DAC Prolific Author Award, the Asia and South Pacific DAC (ASPDAC) Frequently Cited Author Award, 13 Best Paper Awards, several International CAD Contest Awards, the Communications of the ACM Research Highlights in 2014, the ACM/SIGDA Outstanding New Faculty Award in 2005, the National Science Foundation CAREER Award in 2007, the SRC Inventor Recognition Award three times, the IBM Faculty Award four times, the UCLA Engineering Distinguished Young Alumnus Award in 2009, and the University of Texas at Austin RAISE Faculty Excellence Award in 2014. He has served as a Senior Associate Editor for *ACM Transactions on Design Automation of Electronic Systems*, an Associate Editor for the IEEE DESIGN AND TEST, the IEEE TRANSACTIONS ON COMPUTER AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS, the IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS, the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—I: REGULAR PAPERS, the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—II: EXPRESS BRIEFS, *Science China Information Sciences*, and the *Journal of Computer Science and Technology*. He has served as the Program/General Chair of ISPD 2007/2008, the TPC Chair for ASPDAC 2016, the Vice Program Chair for the 2017 International Conference on Computer Aided Design, the Tutorial Chair for DAC 2014, among others.