# ILPGRC: ILP-Based Global Routing Optimization With Cell Movements

Tiago Augusto Fontana, Erfan Aghaeekiasaraee, Renan Netto, Sheiny Fabre Almeida, Upma Gandhi, Laleh Behjat, *Senior Member, IEEE*, and José Luís Güntzel, *Senior Member, IEEE*

***Abstract*—The placement and routing steps directly impact the circuit performance, area, power consumption, and reliability. To handle the high complexity of modern circuits, these steps are tackled separately by applying a divide-and-conquer approach. Unfortunately, due to the continuous increase of design rules complexity, the convergence of solutions can suffer from misalignment, and the effects of an unsatisfactory placement will be noticed only during routing when the placement is considered fixed. In this work, we propose the ILPGRC, an integer linear programming (ILP)-based technique that simultaneously moves cells and routes nets to optimize Global Routing. ILPGRC enables the relocation of cells that can lead to routing issues without compromising the quality concerning the number of VIAs, wirelength, and design rule violations (DRVs). We also propose a partitioning strategy named Checkered paneling, which reduces the input size of the ILP model, making this approach scalable. The Checkered paneling strategy enables the execution of multiple ILP models in parallel, providing a speedup for large circuits. Additionally, we propose a GCell cluster-based approach to legalize the solution with minimum disturbance and displacement. We evaluated our technique for the ISPD 2018 and ISPD 2019 Contests circuits within a physical synthesis flow composed of state-of-the-art place and route academic tools. The results after the detailed routing show that ILPGRC can reduce, on average, the number of VIAs by 4.69% with less than 1% impact on wirelength. Additionally, ILPGRC reduces the number of DRVs in most cases with no open nets left.**

***Index Terms*—Integer linear programming (ILP), physical design, placement, routing, routing with cell movement.**

## I. Introduction

With the advancement of technology, the wire delay became more prominent than the circuit components delay [1], [2], [3]. Thereby, routing along with placement became the two most important steps determining a circuit's performance. Most of the recent routing works focus on wirelength reduction while ignoring the number of vertical interconnect accesses (VIAs) [4], [5], [6], [7], [8]. This increases the mismatch between global and detailed routing (DR).

Yield, reliability, product performance, and cost are the cornerstones of a successful IC manufacturing technology. In such a context, a large number of VIAs can significantly reduce the *reliability* of the circuit [1]. Additionally, the delay is proportional to wire resistance. Hence, reducing the number of VIAs can result in less wire resistance.

Placement and routing are usually treated as two separate problems. Therefore, the effects of a bad placement solution can be amplified during routing to the extent that the circuit placement is deemed unroutable and the placement needs to be redone. To mitigate such problems, some recent works have enabled changes in cells' placements during the routing [9], [10], [11], [12]. Such a procedure is currently referred to as *routing with cell movement* problem.

In this article, we propose an integer linear programming (ILP) model, named ILPGRC, that simultaneously moves cells and reroutes the nets. In order to reach scalability concerning routing using ILP models, a dynamic region-based partitioning, named Checkered Paneling, is proposed. The main contributions of this article are as follows.

1) Developing an ILP formulation that simultaneously moves cells and reroutes nets targeting routing optimization and DRVs reduction.
2) Designing a dynamic and hierarchical region-based partitioning strategy, named Checkered Paneling that reduces the input size of the ILP model and enables parallelization. The parallel execution using eight threads performs up to 6.4 times faster than the sequential version.
3) Designing a cluster-based approach to legalize the solution with minimum disturbance and displacement.
4) Evaluating the effectiveness of the proposed technique after the DR step in an academic design flow using state-of-the-art routers. We used on the benchmark circuits from ISPD 2018 [13] and ISPD 2019 [14] contests benchmarks. These circuits include DR data, thus allowing the evaluation of a number of metrics, such as off-track VIAs and wires, wrong way wires, metal shorts, min-areas, spacing rules, and open nets. This is in contrast to most of the related work, which limits

Manuscript received 9 November 2022; revised 9 May 2023 and 24 July 2023; accepted 2 August 2023. Date of publication 15 August 2023; date of current version 26 December 2023. This work was supported in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), Brazil (Finance Code 001); in part by the Brazilian Council for Scientific and Technological Development (CNPq) through PQ under Grant 316984/2021-3; and in part by the Natural Sciences and Engineering Council of Canada under Grant 10015685. This article was recommended by Associate Editor D. Z. Pan. *(Corresponding author: Tiago Augusto Fontana.)*

Tiago Augusto Fontana, Renan Netto, Sheiny Fabre Almeida, and José Luís Güntzel are with the Department of Computer Science and Statistics, Universidade Federal de Santa Catarina, Florianopólis 88040-900, Brazil (e-mail: tiagoaugustofontana@gmail.com).

Erfan Aghaeekiasaraee, Upma Gandhi, and Laleh Behjat are with the Department of Electrical and Software Engineering, University of Calgary, Calgary, AB T2N 1N4, Canada (e-mail: erfan.aghaeekiasarae@ucalgary.ca; upma.gandhi@ucalgary.ca; laleh@ucalgary.ca).

Digital Object Identifier 10.1109/TCAD.2023.3305579

© 2023 The Authors. This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License.
For more information, see https://creativecommons.org/licenses/by-nc-nd/4.0/

the improvement evaluation to the global routing (GR) step.

The proposed technique was able to reduce the number of VIAs by up to 11.74% while moving 6.44% of the cells. On average, ILPGRC reduces the number of VIAs by 4.69%, moving only 1.98% of the cells and degrading the wirelength by less than 1% as compared to the baseline flow.

The remaining of this article is organized as follows. In Section II, a brief background on placement and routing techniques is presented. In Section III, details of the GR problem formulation are given. In Section IV, the proposed mathematical formulation for the target problem and the proposed technique flow are presented. In Section V, the experimental evaluation is discussed. Finally, in Section VI, the main conclusion is drawn.

## II. Related Work

Traditionally, the routing engines assumed that the cells are fixed. As a result of the two recent ICCAD 2020 [15] and 2021 [16] computed-aided design (CAD) Contests, a number of academic works started to perform cell movements during the routing stage, such as Starfish [10], Zou et al. [12], Huang et al. [17], Zhu et al. [18], ATLAS [19], and CR&P [11], [20]. These two contests proposed "*routing with cell movement*" to point out that new routing engines need to be able to move some cells and reroute some nets to optimize the routing solution.

For the purpose of fusing routing with cell movement, Starfish [10] developed a partial rerouting technique. They created a multisource single-target AStar method that connects relocated cells to the topology trunk. The authors of Starfish use the ICCAD 2020 Contest benchmarks [15] to validate their technique. Huang et al. [17] used breadth-first search (BFS)-based approximation to reduce the optimal region for cell movement. The BFS search considers routing constraints such as layer direction, minimum layer, and overflow as the search heuristic. This work also uses AStar for partial rerouting after cell movement and ICCAD 2020 Contest benchmarks. The technique proposed by Zou et al. [12] first alleviates the congestion by rerouting the circuit using a congestion-aware 3-D GR. Then, it tries to move cells using a modified version of SRP [21]. In the end, it applies an edge-adjusting algorithm to reduce the wirelength. As in the two previous works, Zou et al. evaluated their technique using the ICCAD 2020 Contest benchmarks.

Zhu et al. [18] proposed a cell movement approach based on a lookup table considering routing directions and layer-based power consumption. The lookup table of wirelength is used to generate a gain map for each movable cell. Then, based on the gain map, alternately perform several rounds of cell movement and partial rip-up and rerouting. After each movement, they need to re-estimate and generate a new gain map for the subsequent cells, since this estimation probably becomes inaccurate. In the end, they moved some cells to their original positions to legalize the maximum number of moved cells imposed by the ICCAD Contest. Zang et al. [19] proposed a two-level layer-aware scheme named ATLAS. It first performs an Incremental 3-D GR to improve only the routing. Then performs a VIA-sharing cluster to group the cells. After this, iteratively move cells to the median and reroute the nets using an A*-based partial rerouting until no gain in routing is observed. Finally, a single-cell movement is performed to their original positions to satisfy the movement constraint. Both Zhu et al. and ATLAS were evaluated using the ICCAD 2020 and 2021 Contest benchmarks and slightly outperformed the first-place team of the ICCAD 2021 contest.

These five works have similar drawbacks that they only support the ICCAD 2020 and 2021 Contest benchmarks. Unfortunately, these benchmark suites are oversimplified since they rely only on GCells information to specify the locations of cells and net connectivity. There is no information about technology nodes, cell geometries, circuit rows, circuit sites, and routing tracks. It is impossible to say if a movement could be legalized and/or if the optimization in the GR step will result in some optimization after the DR. Therefore, by using this set of benchmarks, we cannot measure the real impact of movements in the physical design flow.

In [11] and its extension [20], CR&P and CRP 2.0 are proposed which use a cost function to identify critical areas and reduce the congestion by moving some cells. Then, an ILP-based legalizer is used to generate new legal locations for the candidate cells. CRP 2.0 also introduces caching technique to speedup the technique. Both CR&P and CRP 2.0 techniques were evaluated after the DR solution using the ISPD 2018 and 2019 Contest benchmarks. CR&P reduced the number of VIAs by 2.06% and the wirelength by 0.14% on average. CRP 2.0 reduced the number of VIAs by 3.59% and the wirelength by 0.09% on average. CR&P and CRP 2.0 first move a set of cells, leaving the rerouting of affected nets to the end. Consequently, the rerouting process may not be possible for some (or all) of those new cell locations, demanding more work to revert these changes.

To overcome the aforementioned issues, the present work moves the cells and reroutes the nets simultaneously. Also, the initial solution is always considered as a candidate. Therefore, the proposed technique never leads to results worse than the initial ones. The evaluation was conducted after the DR step for the ISPD 2018 and ISPD 2019 Contests circuits using the official contest evaluator binary. The biggest circuit of ISPD 2019 benchmarks has 900K cells and 895K nets using a 32-nm technology node.

## III. Problem Formulation

This section presents a mathematical formulation for the routing with cell movement problem. For convenience, Table I presents the description of all symbols used in this work. Given an initial solution where cells have been placed, and the GR is done, the routing with cell movement problem consists of moving a set of cells to minimize some routing metrics, such as the number of VIAs and/or wirelength, by finding the regions that connect all net pins. In the GR problem, the area of the circuit is partitioned into regions called GCells ($\mathcal{G}$), where the 3-D routing space can be modeled as a 3-D graph of GCells. In this formulation, the length of a net is measured based on the number of GCells it spans. A cell movement consists of reassigning the location of a cell from one GCell

TABLE I
SYMBOLS USED IN THIS WORK

| Symbol | Mean | Symbol | Mean | Symbol | Mean |
|---|---|---|---|---|---|
| $b_i$ | Pin blockage | $max(g_i)$ | Max corner of GCell $g_i$ | $via_{ij}$ | Number of VIAs of candidate $j$ of net $i$ |
| $\mathcal{C}$ | Set of Cells | $min(g_i)$ | Min corner of GCell $g_i$ | $X_{left}$ | Left limit of circuit rows |
| $C(n_i)$ | Set of cells connected to $n_i$ | $\mathcal{N}$ | Set of Nets | $X_{right}$ | Right limit of circuit rows |
| $c_i$ | Cell i | $\mathcal{N}(c_i)$ | Nets connected to $c_i$ | $x(c_i)$ | Coordinate X of cell $c_i$ |
| $cost_{i,j}$ | Cost of $n_i$ using candidate $j$ | $N(m_{ij})$ | Set of nets of $m_{i,j}$ | $W_{site}$ | Width of circuit site |
| $D(g_i^k)$ | Demand of GCell $g_i^k$ | $n_i$ | Net $i$ | $w(c_i)$ | Width of cell $c_i$ |
| $\Delta(g_m, g_n)$ | Distance between $g_m$ and $g_n$ | $\mathcal{P}$ | Set of Pins | $wl_{ij}$ | Wirelength of candidate $j$ of net $i$ |
| $\mathcal{G}$ | Set of GCells | $\mathcal{P}(n_i)$ | Pins belonging to net $n_i$ | $Y_{bottom}$ | Lower limit of circuit rows |
| $\mathcal{G}(n_i)$ | GCells crossed by net $n_i$ | $p_i$ | Pin $i$ | $Y_{top}$ | Upper limit of circuit rows |
| $g_i^k$ | GCell $i$ of metal layer $k$ | $R(m_{i,j})$ | Set of routing candidates of $m_{i,j}$ | $y(c_i)$ | Coordinate Y of cell $c_i$ |
| $H_{row}$ | Height of circuit row | $R(g_k)$ | Set of net candidates routed through $g_k$ | $\alpha_k$ | Wirelength weight of metal layer $k$ |
| $h(c_i)$ | Height of cell $c_i$ | $r_{i,j}$ | Routing candidate | $\beta_k$ | VIA weight of metal layer $k$ |
| $L(c_i)$ | Set of possible locations of $c_i$ | $S(g_i^k)$ | Supply of GCell $g_i^k$ | $\omega$ | VIA factor |
| $loc(p)$ | Location of Pin $p$ | $\mathcal{V}$ | Set of VIAs | | |
| $m_{i,j}$ | Position Candidate | $\mathcal{V}(n_i)$ | VIAs belonging to net $n_i$ | | |

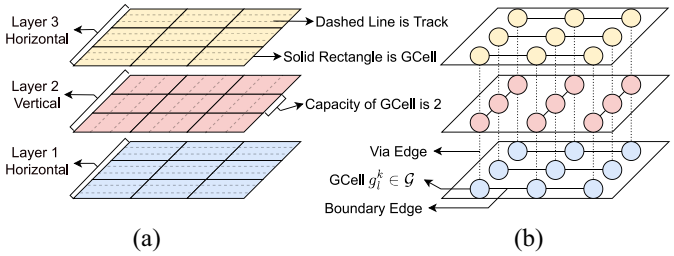


Fig. 1. (a) 3-D routing space. (b) Respective graph model.

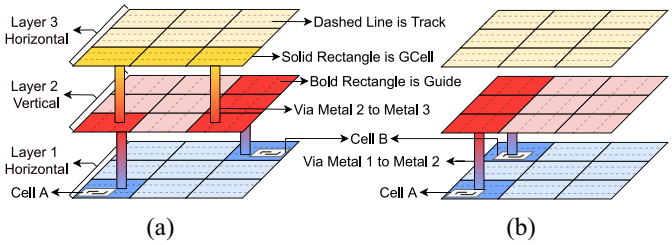


Fig. 2. Example of routing optimization through cell movement. (a) Initial cells position and routing. (b) Routing solution after moving call B to the top left GCell.

to another. The features of each cell, including pin location and blockages, are defined in the standard cell library.

Fig. 1 shows a representation of the 3-D routing space on the left side (a) and the respective graph model on the right side (b). Fig. 1(a) depicts how each metal layer ($k$) could be subdivided into regions called GCells ($g_i^k \in \mathcal{G}$). The GCells in Fig. 1(a) are demarcated with the solid rectangles. In the graph, each GCell is represented by a node. The dashed lines in Fig. 1(a) represent the tracks for each metal layer. The number of tracks crossing each GCell defines the capacity of each GCell. In this example, the capacity of each GCell is 2, and this value is associated with each graph node shown in Fig. 1(b). The neighborhood of two GCells will be represented by an edge in the graph [Fig. 1(b)]. This neighborhood could be: in the same metal layer (solid lines) or between lower/upper adjacent metal layers (dashed lines). Note that, for each metal layer, only the neighborhood in the preferred routing direction is connected by edges. For example, only horizontal connections are made within layers 1 and 3.

Fig. 2 illustrates an example of routing optimization through cell movement for a 2-pin net. Fig. 2(a) presents an initial solution for a single two-pin net. Assuming that for this net, the minimum routing layer is metal 2,[1] the initial routing solution is composed of four VIAs and nine GCells. Fig. 2(b) presents the same net after moving Cell B from the top right to the top left GCell in Layer 1, thus making it possible to optimize the routing. This final routing solution is composed of two VIAs and five GCells. This simple example shows that moving cells during the routing steps may further improve the solution without compromising the design constraints.

[1]The minimum routing layer constraint, for a specific net, establishes the layer that the routing must be on or above.

It is necessary to keep track of placement constraints during the routing to ensure that at the end of routing, the placement will be legalized. Therefore, the routing with cell movement problem can be defined as follows.

Input: A netlist with legalized placement and GR solution.

Problem: Move a set of cells to optimize the number of VIAs and wirelength, keeping a legalized placement and all nets connected.

The objective of this problem could be expressed by (1), while (2)–(7) are the placement constraints, and (8)–(11) are the routing constraints

$$\min \sum_{n_i \in \mathcal{N}} |\mathcal{V}(n_i)| + |\mathcal{G}(n_i)| \tag{1}$$

$$\text{s.t.: } X_{\text{left}} \leq x(c_i) \leq X_{\text{right}} - w(c_i) \quad \forall\, c_i \in \mathcal{C} \tag{2}$$

$$Y_{\text{bottom}} \leq y(c_i) \leq Y_{\text{top}} - h(c_i) \quad \forall\, c_i \in \mathcal{C} \tag{3}$$

$$x(c_i) = n \times W_{\text{site}}, \quad n \in \mathbb{N} \wedge \forall\, c_i \in \mathcal{C} \tag{4}$$

$$y(c_i) = m \times H_{\text{row}}, \quad m \in \mathbb{N} \wedge \forall\, c_i \in \mathcal{C} \tag{5}$$

$$(x(c_i) + w(c_i) < x(c_j)) \vee (x(c_j) + w(c_j) < x(c_i)) \quad \forall\, c_i, c_j \in \mathcal{C} \ \wedge\ y(c_i) = y(c_j) \ \wedge\ \forall\, i \neq j \tag{6}$$

$$(y(c_i) + h(c_i) < y(c_j)) \vee (y(c_j) + h(c_j) < y(c_i)) \quad \forall\, c_i, c_j \in \mathcal{C} \ \wedge\ x(c_i) = x(c_j) \ \wedge\ \forall\, i \neq j \tag{7}$$

$$\exists\, \mathcal{G}(n_i) \subset \mathcal{G} \quad \forall\, n_i \in \mathcal{N}(c_i) \tag{8}$$

$$\exists g_j \in \mathcal{G}(n_i) | \ \min(g_j) \leq \text{loc}(p_i) \leq \max(g_j), \quad \forall\, p_i \in \mathcal{P}(n_i) \wedge n_i \in \mathcal{N}(c_i) \tag{9}$$

$$\exists\, g_n \in \mathcal{G}(n_i) \mid \Delta(g_m, g_n) = 1 \quad \forall\, g_n, g_m \in \mathcal{G}(n_i) \wedge n_i \in \mathcal{N}(c_i) \tag{10}$$

$$D(g_l^k) \leq S(g_l^k) \,\forall\, g_j^k \in \mathcal{G}(n_i) \wedge n_i \in \mathcal{N}(c_i). \tag{11}$$

Equation (1) states that the objective of the problem is to minimize, for all nets, the total number of VIAs ($|\mathcal{V}|$) and the total wirelength ($|\mathcal{G}|$). As mentioned before, the length of a net is measured based on the number of GCells it spans. Equations (2) and (3) ensure that each Cell $c_i \in \mathcal{C}$ is placed within the circuit rows region while (4) and (5) state that each Cell $c_i$ is aligned with circuit sites and rows, respectively. The nonoverlap for each pair of cells $c_i, c_j \in \mathcal{C}$ is guaranteed by (6) if $c_i$ and $c_j$ are placed in the same row or by (7) if $c_i$ and $c_j$ are placed in different rows.

Equation (8) guarantees that for every net $n_i$ connected to Cell $c_i$, there is a subset of GCells allowing $n_i$ to be fully routed, and thus, no open nets can exist. Equation (9) establishes that all pins $p_i$ of a given net $n_i$ can be assigned to a GCell $g_j$. Equation (10) states that for every GCell $g_n$ belonging to $\mathcal{G}(n_i)$ there will be an adjacent GCell $g_m$. Equation (11) ensures that the demand for each GCell $g_l$ located in layer $k$ will not exceed the maximum capacity of that GCell. This takes into account the overflow concerns of each GCell.

With the aforementioned constraints, we can ensure that all the cells are legalized and that the circuit is fully routed. Therefore, no open nets are allowed.

**Algorithm 1:** RUN_ILPGRC(lef, def, guide)

**Input:** *lef* = Technology file,
*def* = Design placement,
*guide* = Global Routing solution
**Output:** *def* = Design with new placement,
*guide* = New Global Routing solution

```
1  load_circuit(lef, def, guide);
2  GUIDE_PREPROCESSING();                    // subsection IV-A
3  nlev ← CHECKERED_PANELING();              // subsection IV-B
4  foreach level ∈ nlev do
5      M ← {};                               // Map of movements
6      R ← {};                               // Map of routing guides
7      foreach color ∈ {black, white} do
8          panels ← get_panels(color, level);
           // run parallel
9          foreach panel ∈ panels do
10             RUN_ILP(panel);               // subsection IV-C
11             l ← LEGALIZE(panel);          // subsection IV-D
12             if l = TRUE then
13                 save_movements_and_routing(M, R);
14             end
15         end
16     end
17     update_database(M, R);
18 end
19 write_output_def();
20 write_output_guide();
```

## IV. Proposed Technique

This section presents the proposed technique for the target problem introduced in the previous section. The proposed technique comprises four main steps: 1) guides preprocessing; 2) checkered panels construction; 3) ILP model construction and solving; and 4) cluster-based panel legalization. Each of these steps will be discussed in detail in Sections IV-A–IV-D. The main contribution of this work is the ILP model that simultaneously moves cells and reroutes all nets, which is presented in Section IV-C.

The overall flow of the proposed technique is presented by Algorithm 1. The inputs to the algorithm are the technology file (.lef), the design file (.def), and the initial GR solution file (.guide) of the given layout. Library Exchange Format (LEF) and Design Exchange Format (DEF) are two industrial files to describe a design. LEF file defines the elements of an IC process technology and the associated library of cell models. DEF file defines the elements of an IC design relevant to physical layout, including the netlist and design constraints.

After loading the files that describe the circuit, the initial GR solution is preprocessed in line 2. More details of this preprocessing step will be presented in Section IV-A. Next, in line 3, the checkered panels are created, and the number of levels is stored in variable *nlev*. The motivation and details of this partitioning technique are given in Section IV-B. Each subpart of the problem is called a panel, and each panel is associated with a level and one color.

Afterward, all panels with the same *level* and *color* are processed in parallel (lines 9–15). For each of these panels, the proposed ILP model for moving cells and rerouting nets is created and solved in line 10. Section IV-C presents the details about this step. Then, in line 11, the panel is legalized using Abacus [22]. The details of the panel legalization process are presented in Section IV-D. Later, if the panel is considered legalized, all the movements and new routing solutions are stored in variables $\mathcal{M}$ and $\mathcal{R}$ (line 13). Once all panels of the same *level* and *color* were processed, all movements and routing solutions are applied to the database (line 17). In the end, the new DEF and Guide files are generated as output of the algorithm (lines 19 and 20).

### A. Guides Preprocessing

The first step of our proposed technique consists in preprocessing the input GR solution, so as to map the original guide rectangles to the adopted GCell structure. Fig. 3(a) presents an example of an original routing guide for a three-pin net: white squares represent GCells, gray rectangles represent cells, and the colored rectangles represent the guide(s). In this example, cells are placed in GCells A, G, and L. It also puts in evidence that the original guides may be entirely within a single row/column of GCells (red and green rectangles), or may span multiple rows/columns of GCells (purple and yellow rectangles). The latter case could cause cycles in the GCell graph, degrading or making unfeasible the solution of a path search algorithm. Therefore, mapping the original guides that span more than one row/column of GCells is crucial since it avoids those cycles.

The guide preprocessing step begins by mapping the original guide to a graph [Fig. 3(b)] where the nodes represent GCells intersected by a guide in a given metal layer, and the edges represent the neighborhood of those GCells. GCell neighborhood may be within the same metal layer (solid edges) and, thus, must respect the preferred direction in that metal layer, or may be the overlapping between GCells in two different layers (dashed edges), therefore representing a VIA. The graph nodes representing the GCells that contain the net

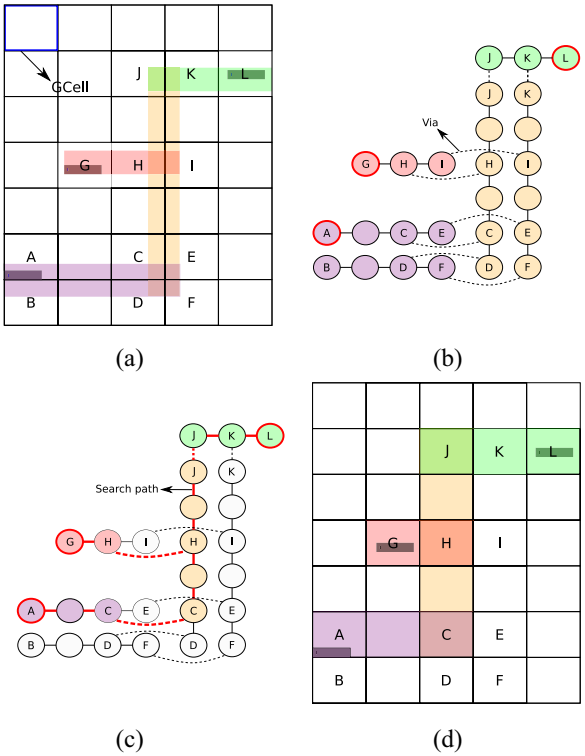


Fig. 3. Guide preprocessing steps. (a) Original guide. (b) Guide mapped to a graph. (c) Path search result. (d) Preprocessed guide.

cells are marked as terminal nodes and appear outlined in red in Fig. 3(b). Once the graph has been built, the preprocessing step performs a BFS, starting from any terminal node and progressing toward all other terminal nodes. Such a procedure determines the minimum path needed to keep this net connected. Fig. 3(c) shows the result for this BFS search on the example considered, where the minimum path is identified by the sequence of the colored nodes. Hence, in this example, the final guide structure has 13 GCells and 3 VIAs, as shown in Fig. 3(d).

### B. Checkered Panels

GR using math solvers can be time-consuming and very slow, especially for circuits with a large number of nets and/or containing nets that cross the whole circuit area. This may be worsened when routing is combined with cell movements because each cell movement generates a set of routing candidates, and therefore, the number of variables inside the ILP model is potentially huge. To circumvent these problems, we propose the so-called Checkered Paneling strategy. Basically, this strategy partitions the circuit area and, for each partition (panel), identifies all nets that are entirely inside the partition. Then, the routing with cell movement problem is solved separately for each panel considering only the identified nets. The partitioning procedure is repeated, assuming progressively larger partitions, so as to take into account the nets that span wider regions. In summary, the Checkered Paneling strategy divides the problem into a number of subproblems of smaller sizes that can be solved in parallel using an ILP solver in acceptable runtimes. As an extra benefit, the Checkered Paneling helps balance the workload between the threads since

**Algorithm 2:** CHECKERED_PANELING($gcw$, $gch$)

**Input:** $gcw$ = Number of GCells to panel width,
$gch$ = Number of GCells to panel height
**Output:** Number of levels

```
1  proc_nets ← ∅;                    // Set of processed nets
2  gcells ← max(num_gcells_width, num_gcells_height);
3  nlevels ← ⌈log2(gcells)⌉ + 1;
4  for level ← 1 to nlevels do
5      panel_w ← gcw * 2^(level−1);
6      panel_h ← gch * 2^(level−1);
7      panels ← make_regions(panel_w, panel_h);
8      foreach p ∈ panels do
9          nets ← search_inside(region(p))\proc_nets;
10         set_nets(p, nets);
11         proc_nets ← proc_nets ∪ nets;
12     end
13 end
14 return nlevels;
```

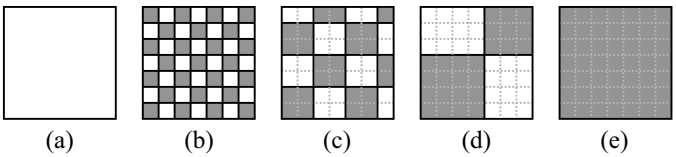


Fig. 4. Example of checkered panels for a given layout. (a) Layout. (b) Level 1. (c) Level 2. (d) Level 3. (e) Level 4.

the nets that are entirely within the panels of a given level present similar wirelength.

Fig. 4 presents how the Checkered Paneling procedure hierarchically partitions the circuit layout into panels. Two input variables define the panel width ($gcw$) and height ($gch$), expressed in the number of GCells, in the first level. The panel size in every subsequent level is defined as being twice the panel size in the preceding level. The levels end when the whole layout is covered by a single panel [Fig. 4(e)]. Within a given level, each panel is associated with a color (either black or white). Panels belonging to the same level and color will be executed in parallel. It is important to note that a panel will not have right or left neighbors of the same color. Such characteristic is to ensure that a cell that crosses the panel edge will not be mapped simultaneously to two threads at the same time. Consequently, our method is deterministic between sequential and parallel executions, i.e., sequential and parallel executions will produce the same result.

Algorithm 2 presents a pseudocode for creating the checkered panels and making the association between panels and nets. The algorithm receives two variables as input: the panel width ($gcw$) and height ($gch$), expressed in the number of GCells. The output is the number of levels. First, in lines 1 and 2, the set of processed nets is initialized, and the maximum number of GCells in both dimensions of the circuit layout is calculated. Then, the number of levels is calculated in line 3. As the panel size in a given level is at most twice the size of the panel in the preceding level, the number of levels will be 1 plus the ceiling of logarithmic in base 2 of the maximum number of GCells. Next, for each level starting in 1, the panel width ($panel\_w$) and height ($panel\_h$) are calculated in lines 5 and 6, respectively. Note that, in level 1, the panel size will equal the input variables $gcw$ and $gch$. After this, function $make\_regions(panel\_w, panel\_h)$ in line 7

partitions the circuit layout creating all the panels for this specific level. Subsequently, for each panel $p \in panels$, the procedure will identify and associate to the panel all nets that will be processed. To that purpose, the function *search_inside* in line 9 returns all the nets whose bounding boxes are inside the panel borders. Function *region*($p$) determines the borders for a given panel $p$. This search is implemented using the spatial structure RTree [23]. Note that, in line 9, all nets that are already associated with any lower-level panel (nets that are in *proc_nets* set) are removed before making the association with the current panel. Function *set_nets*($p$, *nets*) in line 10 generates the mapping between all nets and panel $p$. After this, *proc_nets* receives the nets for this panel in line 11. Finally, line 14 returns the number of levels.

### C. ILP Model

This section presents the ILP model to solve the routing with cell movement problem presented in Section III. To solve this problem, we need to model all nets and circuit resources. For each net $n_i$, we generate a set of possible routing candidates. Each routing candidate $j$ of net $n_i$ has its respective binary variable $r_{ij}$ to indicate which $j$, among all candidates of $n_i$, must be selected. Then, the ILP model aims to select the best set of candidates that optimizes the circuit routing. In order to do that, the objective function of the ILP formulation minimizes the weighted sum of each variable in (12), where $\text{cost}_{i,j}$ denotes the cost of $n_i$ when it is routed using candidate $j$

$$\min \sum_{i=1}^{n} \sum_{j=1}^{m} \text{cost}_{i,j} \times r_{ij}. \tag{12}$$

Equation (13) presents the cost $\text{cost}_{i,j}$ for a net $i$ and candidate $j$ that is a weighted sum of the wirelength $wl_{ij}$ and the number of VIAs $\text{via}_{ij}$ in each metal layer $k$. It is important to note that, as this model is applied to the GR step, the wirelength of a net is measured in the number of GCells the net spans. In contrast, the number of VIAs is the number of intersections between segments of the same net that are in adjacent metal layers. $\alpha_k$ and $\beta_k$ are constant values (weights) for each metal layer $k$, which is used to penalize the lower metal layers, thus distributing the net segments through the upper layers so as to alleviate the congestion. Short nets are less penalized because the sum of wirelength ($wl_{ij}$) and the number of VIAs ($\text{via}_{ij}$) are smaller. Therefore, short nets are assigned to the lower layers. On the other hand, for the long nets, the sum of wirelength and VIAs is bigger, and thus, they are assigned to the upper layers. In this work, we use $\beta_k = \omega \times (\alpha_k + \alpha_{k+1})$, where $\omega$ represents the VIA factor and is equal to 1 in all experiments. We experimentally tested different values for $\omega$, 0.001, 0.1, 1, 10, 100, and the best average VIA reduction was with $\omega = 1$. All constant values used in this work are presented in Section V-A2 in Table II

$$\text{cost}_{i,j} = \sum_{k=1}^{n} \alpha_k \times wl_{ij}^{k} + \sum_{k=1}^{n-1} \beta_k \times \text{via}_{ij}^{k}. \tag{13}$$

Concerning cell movement, for each cell $c_i$, we generate a set of possible locations $L(c_i) = \{l_1(c_i), l_2(c_i), \ldots, l_n(c_i)\}$. For each location, there is a binary variable $m_{i,j}$ that indicates if cell $c_i$ should be placed on that location $l_j(c_i)$. We also keep track of each cell initial location by using variable $l_0(c_i)$.

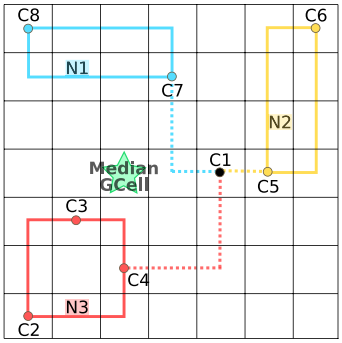


Fig. 5. Illustration showing how the proposed technique determines a candidate location for cell C1 by calculating the Median GCell. Each hue in this image represents the network associated to a cell (except the candidate cell), each cell is represented by colored dots. The green star indicates the median GCell.

When a cell is moved, its nets must be rerouted. Therefore, for each candidate location $m_{i,j}$, we define a set of nets $N(m_{i,j})$ with their respective routing candidates $R(m_{i,j})$. In addition, for each net $n_i$, we define the set of cells connected to this net as $C(n_i)$. Given that, the objective function remains the weighted sum of the net candidates, except that now we need to consider the nets resulting from the cell movements.

Fig. 5 shows an example of how we generate the candidate locations to move a cell C1 which is connected to seven other cells through nets N1, N2, and N3. We calculate the bounding box of each net (represented in the figure by the blue, yellow, and red rectangles) and select as the candidate location the median GCell with respect to all the bounding box coordinates. In the example of Fig. 5, the median GCell of C1 is identified by the green star. This strategy was inspired by the work presented in [24].

The decision to adopt the median GCell as the target position for cell movement was taken after we evaluated nine different movement candidates: the median point of the connected nets, the four neighbors of the median point, and the four neighbors of the initial location of the cell. According to our experiments, among all candidate movements, moving to the median point of each cell has the highest chance of reducing the wirelength of nets connected to the cell. We also performed an experiment with five movement candidates for each cell at the same time: the median GCell, and the four neighboring GCells of the median (N, S, W, and E). Such experiment did not help to improve the quality of the results. This is due to the fact that the median GCell is the optimum point for a movement considering all nets connected to a cell. Therefore, the ILP model will prefer to move the cells for this median point instead of the neighboring GCells. By adding four times more movement candidates the runtime increased by only 2.03 times, on average.

Fig. 6 presents an example of the routing candidates generation for a two-pin net connecting Cell A (lower left corner) and Cell B (upper right corner). For simplicity, in this example, we consider that the design uses only three metal layers. For a two-pin net, we generate all possible combinations of L-shape patterns, i.e., the combination of all possible layers for

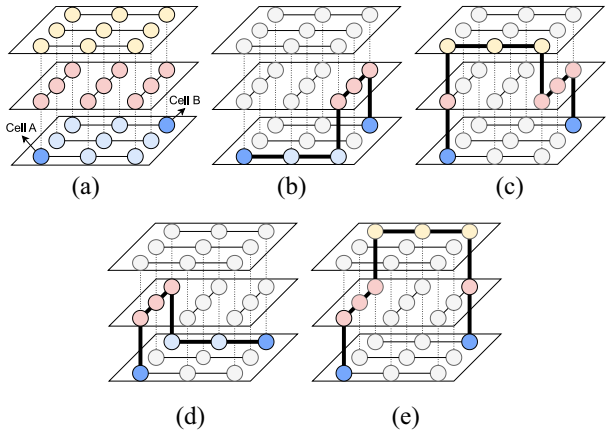


Fig. 6. Routing candidates for a 2-pin net connecting Cell A and Cell B. (a) Initial cell locations. (b) Lower L-shape in metal 1 and 2. (c) Lower L-shape in metal 2 and 3. (d) Upper L-shape in metal 1 and 2. (e) Upper L-shape in metal 2 and 3.

each of the L-pattern. Fig. 6(b) and (c) brings the two possible candidates for the lower L-shape, and Fig. 6(d) and (e) shows the two possible candidates for the upper L-shape. For a multipin net, we first generate the Steiner tree using the Flute algorithm [25]. Then, for each two-point segment, we use the same approach of two-pin nets described above. In the ILP model, we ensure that for a given net, one and only one routing candidate is selected for each of the two-point segments.

To properly route the circuit, we need to add a few constraints to the ILP model. The first one states that each net must be routed by a single routing candidate, which is captured by

$$\sum r_{ij} = 1 \ \forall j \ \wedge \ i = 1 \cdots n. \tag{14}$$

The second constraint ensures no overflow in the GCells. Each GCell $g_k$ has a supply $S(g_k)$ representing the number of available routing tracks in the GCell $g_k$. The number of nets routed through a GCell cannot exceed its supply. We define $R(g_k)$ as the set of net candidates routed through GCell $g_k$. Then, the second constraint is modeled as in (15), which must be defined for each GCell

$$\sum_{r_{ij} \in R(g_k)} r_{ij} \leq S(g_k), k = 1 \cdots K. \tag{15}$$

Next, we need to add a few constraints to ensure those movements do not invalidate circuit routing. The third constraint ensures that each cell is assigned to one candidate location, as expressed in

$$\sum m_{i,j} = 1 \ \forall j \ \wedge \ i = 1 \cdots n. \tag{16}$$

The next constraint ties the cell candidate locations to their respective nets. This constraint is specified in (17) and must be defined for each candidate location $m_{i,j}$ and for each net in $N(m_{i,j})$ associated with this location. For each of those nets, the sum of their candidate variables $r_{kl}$ must equal the value of the candidate location $m_{i,j}$. As a consequence, if the cell is moved to location $l_j(c_i)$, then $m_{i,j} = 1$ and the net must use one of the candidates associated with $m_{i,j}$. Otherwise, $m_{i,j} = 0$ and the net cannot use any of those candidates. Notice that we do not need to establish this constraint for the initial location $l_0(c_i)$. This is because there are already constraints that ensure that all nets are routed someway, so if the cell is not moved, one of the initial routing candidates will automatically be used

$$\sum_{l=1}^{m} r_{kl} = m_{i,j}, i = 1 \cdots n, j = 1 \cdots m, n_k \in N(m_{i,j}). \tag{17}$$

The next constraint certifies that we do not move two cells from the same net (18). This constraint must be specified for each net, and it is necessary to ensure we do not assign different routing candidates for the same net. For a given net, this is done by ensuring that the sum of the moved location variables of all cells connected to that net is one. Notice that this constraint only considers the location variables that do not correspond to the initial location ($i \geq 1$), so that if one cell in the net is moved, all the other ones should remain in their initial locations

$$\sum_{c_i \in C(n_k)} \sum_{j=1}^{m} m_{i,j} = 1, k = 1 \cdots n. \tag{18}$$

Finally, we need to update the GCell supply constraints to include cell blockages. Some cell pins impose blockages on metal layers, so the ILP formulation must consider this to avoid overflowing the GCells. In order to do that, we defined $Y(g_k)$ as the set of cell locations inside GCell $g_k$, and each blockage reduces the GCell supply in $b_i$. Then, (19) models this constraint by considering not only the nets inside each GCell but also the blockages

$$\sum_{r_{ij} \in R(g_k)} r_{ij} + \sum_{m_{i,j} \in Y(g_k)} b_i \times m_{i,j} \leq S(g_k), k = 1 \cdots K. \tag{19}$$

Algorithm 3 presents a pseudocode for creating and solving the described ILP model. This Algorithm receives as input a circuit panel. The first two lines will query from the database the nets and cells that are associated with the such panel. Then, function *create_initial_variables*, line 3, produces all the variables representing the initial state of placement and routing for this panel. Therefore, the initial solution is also considered as a valid solution which allows taking into account the situation when the technique is not able to find a better solution for a given net or cell. In other words, this ensures that the ILP model is always feasible. It is interesting to observe that in the worst-case scenario, the initial solution may be kept for the entire panel or event for the entire circuit. The function *create_nets_candidates*(*nets*), in line 4, creates different routing solutions for the initial placement. In this work, we generate different pattern routing solutions using different metal layers for each net.

Next, the loop between lines 5 and 13 generates different candidate positions and routing solutions for each cell in the current panel. First, the cell original location is stored in *optig_p* in line 6, and its median position is calculated by function median_position(*cell*) in line 7. The ILP variable representing this cell in the median position is generated by function create_variable(*cell*, *median_p*) in line 8. Then, the cell is placed in the median position (line 9), its connected nets are stored in variable *nets_c* in line 10, and the routing candidates for this position are generated by the function *create_nets_candidates*(*nets_c*) in line 11. Finally, the cell is restored to the initial location in line 12.

**Algorithm 3:** RUN_ILP(Panel)

```
   Input: panel = Panel to create and solve the ILP
1  nets ← get_nets(panel);
2  cells ← get_cells(panel);
3  create_initial_variables(cells, nets);
4  create_nets_candidates(nets);
5  foreach cell ∈ cells do
6  |   orig_p ← position(cell);
7  |   median_p ← median_position(cell);
8  |   create_variable(cell, median_p);
9  |   place(cell, median_p);
10 |   nets_c ← get_nets(cell);
11 |   create_nets_candidates(nets_c);
12 |   place(cell, orig_p);
13 end
14 add_constraints();
15 add_objective();
16 result ← solve_model();
17 if result = Optimal then
18 |   apply_movements_and_routing();
19 end
```

After creating all these candidates for positions and routing, the previous constraints presented in (14)–(19) are added into the ILP model in line 14. Then, the ILP objective, (12), is added in line 15. The ILP model is solved using CPlex [26], and the result is stored in variable *result* (line 16). Finally, if the result from ILP is *Optimal*, the movements and routing solutions are applied to the design in line 18.

### D. Panel Legalization

The ILP model does not consider legal positions when moving cells. This simplification in choosing the positions is fundamental to keeping a small number of variables inside the ILP model. If the ILP model would consider legal positions for each move, these positions should be seen as fixed for the subsequent moves, which would reduce the solution space. In addition, considering legalized positions, the number of variables would increase considerably. This is because each move would have to be combined with all other moves, and for each of those placement combinations, we would have new solutions and candidates for routing. Therefore a legalization step is required after solving the ILP model. This legalization step is executed by function "LEGALIZE(*panel*)" presented in line 11 of Algorithm 1.

This step should legalize the moved cells moving as few already legalized cells as possible in order not to disturb the solution. If we call a legalization algorithm for the whole panel region, the number of affected cells could be higher. Therefore, we propose a GCell clustered-base approach to legalize each panel. Fig. 7 displays an example of a GCell cluster for a panel. For each row of GCell, we cluster neighbor GCells that are unlegalized. In this example, we have nine clusters (C1–C9). A cluster is based on GCells that have unlegalized cells inside. If a GCell has unlegalized cells and all neighbors are legalized, this cluster will be only one GCell (case of C1, C2, C3, C6, and C7). Otherwise, the neighbor unlegalized GCells are merged (case of C4, C5, C8, and C9). Then, picking cluster 5 (C5) as the target cluster (red bold rectangle),

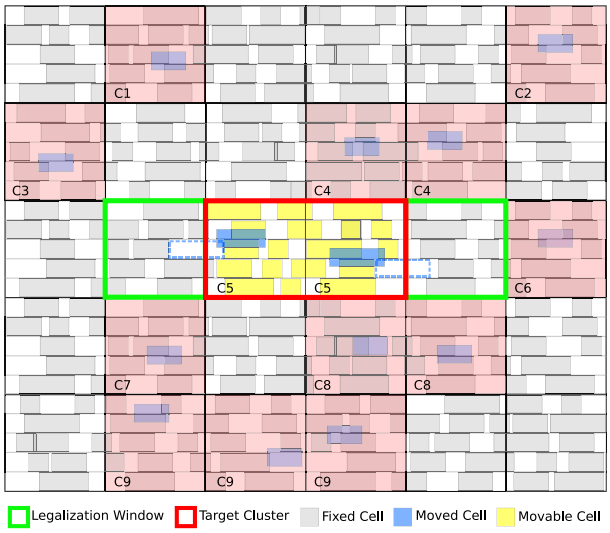


Fig. 7. Example of a GCells clustered-base approach to legalize each panel.

we create a legalization window expanding one GCell on each side of the cluster. The green rectangle presents the legalization window in Fig. 7. Only the cells that are totally inside the cluster area (colored in yellow) are considered movable for the legalization process.

Observe that the GCells belonging to the expansion area (left and right of the cluster) are always legalized because of our previous cluster step. Another importance of this expansion is that it enables us to legalize the cells using free spaces available in the neighboring legalized GCell. This helps us to reduce the displacement of legalized cells (cells that the ILP model has not moved). The blue dashed rectangle marks these free areas in Fig. 7. The next step is to call the Abacus [22] algorithm to legalize each legalization window. Finally, if in the legalization process, some pin access of any cell changes between GCells, we reconnect this pin to the previous GCell using the L-shape pattern route algorithm. In the end, the whole panel will be considered legalized if, and only if, all the clusters can be legalized. Otherwise, all the cells and nets belonging to this panel are restored to the state before the ILP call.

## V. Experimental Results

In order to assess the effectiveness of the proposed technique, a series of experiments were conducted. The obtained results are reported and discussed in this section, which is organized as follows. First, Section V-A presents the experimental setup and software infrastructure. Then, Section V-B explains the methodology adopted to investigate the efficacy of the ILPGRC technique, as well as the chosen benchmarks and tools. Finally, Sections V-C and V-D present the results and discussions.

### A. Experimental Setup

The developed algorithms were implemented in C++ using the open-source library Ophidian [27]. The boost library [28] was used for the data structure, including graphs, trees, and geometric operations. We also used the CPLEX [26] 12.8 library to solve the ILP model. The experiments were

TABLE II
CONSTANT VALUES USED IN THE ILP MODEL

| Metal | Const. | Value | VIA | Const. | Value |
|---|---|---|---|---|---|
| Metal 1 | $\alpha_1$ | 10K | VIA 1 | $\beta_1$ | 20K |
| Metal 2 | $\alpha_2$ | 10K | VIA 2 | $\beta_2$ | 11K |
| Metal 3 | $\alpha_3$ | 1K | VIA 3 | $\beta_3$ | 2K |
| Metal 4 | $\alpha_4$ | 1K | VIA 4 | $\beta_4$ | 1.1K |
| Metal 5 | $\alpha_5$ | 100 | VIA 5 | $\beta_5$ | 200 |
| Metal 6 | $\alpha_6$ | 100 | VIA 6 | $\beta_6$ | 110 |
| Metal 7 | $\alpha_7$ | 10 | VIA 7 | $\beta_7$ | 20 |
| Metal 8 | $\alpha_8$ | 10 | VIA 8 | $\beta_8$ | 11 |
| Metal 9 | $\alpha_9$ | 1 | | | |

executed on a Linux cluster with 48 cores, 2x Intel Xeon Gold 6240R CPU @ 2.40 GHz (Cascade Lake, 2021) and 185-GB RAM.

In addition, this work uses GCells and Panels sizes, as well as the constant values described in the sequel.

*1) GCells and Panels Default Sizes:* We conducted experiments with different GCell sizes: 4k by 5k dbu (2 rows × 50 sites), 4k by 10k dbu (2 rows × 100 sites), and 10k by 10k dbu (5 rows × 100 sites). These GCell sizes did not impact the quality of the final results. We consider the GCells size of 10k by 10k dbu (5 rows and 50 sites) in this work to increase the chances of legalizing the movements without impacting the already placed cells in the same region. With a larger GCell, there are more empty spaces in the GCells meaning less displacement for the placed cells.

1) CUGR GCell size is 3K by 3K dbu.
2) ILPGRC GCell size is 10K by 10K dbu.
3) ILPGRC Checkered panels in level 1 are 5 by 5 GCells.

The only exceptions for these values are circuits ispd19_test4 and ispd19_test5, which use a 65-nm technology node. For these two circuits:

1) CUGR GCell size is 2K by 2K dbu;
2) ILPGRC GCell size is 8K by 8K dbu;
3) ILPGRC Checkered panels in level 1 are 5 by 5 GCells.

The use of the above-mentioned values solves the following issue: if the default values are used to route circuits ispd19_test4 and ispd19_test5, TritonRoute incorrectly reports that the guides are not connected (no pin access found), and ends the execution.

*2) Constant Values Used in the ILP Model:* The constant values used in (13) of the ILP model are presented in Table II. These values penalize the lower metal layers and keep the net nearest to the provided initial solution. Note that the values for VIAs are greater than those used in routing wires. Therefore the ILP model will prioritize solutions that use fewer VIAs, which is the focus of this work. All these values were obtained empirically through extensive experimentation.

### B. Experimental Methodology

To assess the efficacy of ILPGRC and also to comprehend the real impact of moving cells during the global routing (GR) step we relied on the evaluation flow shown in Fig. 8, which includes the detailed routing (DR) step. First, we use CUGR [4] to generate the GR solution for the ISPD Contest 2018 [13] and ISPD Contest 2019 [14] initial placements. Then, the proposed technique (ILPGRC) is applied to optimize the circuits by moving cells and rerouting nets. Subsequently, TritonRoute [7] is used to perform the DR step.


Fig. 8. Evaluation flow using the ISPD Contest 2018 and ISPD Contest 2019 benchmarks.

Finally, the official ISPD Contest 2018 evaluator generates the report that allows us to investigate the quality of the solution. Section V-B2 details the evaluation process. We also used the flow in Fig. 8 to generate baseline results, but in this case, skipping the application of the proposed technique (ILPGRC). The reasons for choosing such benchmarks suite are exposed in Section V-B1. The selection process of CUGR and TritonRoute as the GR and DR tools is justified in Section V-B3.

*1) Benchmarks:* We have chosen the ISPD Contest 2018 [13], and ISPD Contest 2019 [14] benchmarks because they are the most recent benchmarks that include technology files and advanced design constraints, thus allowing the generation of DR solutions. We did not use the ICCAD CAD Contest 2020 [15] or ICCAD CAD Contest 2021 [16] benchmarks because they are overly simplified and do not have technology information. For example, ICCAD 2020 and ICCAD 2021 benchmarks consider that the cells are placed in the center of GCells. Since there is no row or site information, it is not possible to produce DR solutions because the placements cannot be legalized.

On the contrary, the ISPD Contest 2018 [13] and ISPD Contest 2019 [14] benchmarks have LEF and DEF files, and the circuits use one among three technology nodes: 65, 45, and 32 nm. Additionally, these benchmark suites also provide technology information for the circuits, such as Standard cells, Macros, and IO Cells, and the circuits are fully placed and legalized. Finally, cells have complex pin shapes, such as L, Z, and others. Nonetheless, there is no power and timing information in these benchmarks. The main characteristics of these circuits are presented in Table III. In this table, the first column brings the circuit names. For simplicity, we shortened the circuit names by removing the "ispd" and "test" words. For example, circuit "ispd18_test1" has been renamed to "18_t1." Columns 2–8 in Table III display the technology node, number of cells, number of nets, number of I/O pins, number of macro blockages, placement density, and number of routing layers.

Note that benchmark 18_t10 is the only circuit with 100% of density. This is because in this circuit there are no empty spaces due to the use of filler cells. Filler cells are cells with no logical functionality but have the VDD/VSS metal lines matching the rest of the standard cells, ensuring that all power nets are connected. Due to the lack of empty spaces, the ILP model produces a solution for circuit 18_t10 that is exactly the same as the input placement. Therefore, we created a modified version of circuit 18_t10, called 18_t10_nf, by removing the filler cells. This way, the density of 18_t10_nf is 92%. Another interesting point is that all the circuits have nine metal layers for routing, except circuits 19_t4 and 19_t5, which have only five metal layers.

*2) ISPD Evaluator:* The quality of the routing solution result was measured using the official contest ISPD 2018 evaluator. This evaluator receives as input the DEF, Guide,

TABLE III
MAIN CHARACTERISTICS OF ISPD 18 AND ISPD 19 BENCHMARKS. FROM THE LEFT TO RIGHT COLUMNS BRING THE CIRCUIT NAMES, TECHNOLOGY NODE, NUMBER OF CELLS, NUMBER OF NETS, NUMBER OF I/O PINS, NUMBER OF MACRO BLOCKAGES, PLACEMENT DENSITY, AND NUMBER OF ROUTING LAYERS

| ISPD Circuits | Node (nm) | Cells (K) | Nets (K) | I/O Pin | Macros | Density (%) | Layers |
|---|---|---|---|---|---|---|---|
| 18_t1 | 45 | 9 | 3 | 0 | 0 | 85 | 9 |
| 18_t2 | 45 | 36 | 37 | 1,211 | 0 | 57 | 9 |
| 18_t3 | 45 | 36 | 37 | 1,211 | 4 | 65 | 9 |
| 18_t4 | 32 | 72 | 72 | 1,211 | 4 | 89 | 9 |
| 18_t5 | 32 | 72 | 72 | 1,211 | 8 | 92 | 9 |
| 18_t6 | 32 | 108 | 108 | 1,211 | 0 | 99 | 9 |
| 18_t7 | 32 | 180 | 180 | 1,211 | 16 | 90 | 9 |
| 18_t8 | 32 | 192 | 180 | 1,211 | 16 | 90 | 9 |
| 18_t9 | 32 | 193 | 179 | 1,211 | 0 | 91 | 9 |
| 18_t10 | 32 | 290 | 182 | 1,211 | 0 | 100 | 9 |
| 18_t10_nf | 32 | 290 | 182 | 1,211 | 0 | 92 | 9 |
| 19_t1 | 32 | 9 | 3 | 0 | 0 | 83 | 9 |
| 19_t2 | 32 | 72 | 72 | 1,211 | 4 | 72 | 9 |
| 19_t3 | 32 | 8 | 9 | 57 | 4 | 84 | 9 |
| 19_t6 | 32 | 180 | 180 | 1,211 | 16 | 75 | 9 |
| 19_t7 | 32 | 360 | 359 | 2,216 | 16 | 96 | 9 |
| 19_t8 | 32 | 540 | 538 | 3,221 | 16 | 79 | 9 |
| 19_t9 | 32 | 899 | 895 | 3,221 | 16 | 84 | 9 |
| 19_t10 | 32 | 899 | 895 | 3,221 | 16 | 88 | 9 |
| 19_t4 | 65 | 146 | 152 | 4,802 | 7 | 21 | 5 |
| 19_t5 | 65 | 29 | 29 | 360 | 6 | 9 | 5 |

and LEF files and executes the following tasks: 1) invokes Cadence Innovus [29] to perform design rule and connectivity checking; 2) generates DRV and connectivity reports; 3) starts an evaluation program to perform guide and track obedience checking and read the Innovus reports; and 4) generates the report table as output.

The metrics used to evaluate the quality of results in this work were obtained by using the official ISPD 2018 evaluator. They are as follows.

1) *Number of VIAs:* The total number of VIAs after the DR.
2) *Wirelength:* The total length of wires after the DR.
3) *Off-Track VIAs:* The number of VIAs whose center is misaligned with the locations of the tracks.
4) *Off-Track Wirelength:* The length of wires placed out of the track locations.
5) *Wrong Way wirelength:* The length of wires routed in the nonpreferred direction of the layer.
6) *Shorts:* Either a VIA or wire metal overlaps with another object like VIA, wire metal, blockages, or pin shapes.
7) *Min Area:* Specifies the minimum metal area required for polygons on each layer.[2]
8) *Spacing:* Specifies the required spacing between two objects. This metric encompasses the different types of violations which are parallel-run length, end-of-line (EOL), and cut spacing.
9) *Open Nets:* If any pin in a net is disconnected, then the net will be considered as an open net.

*3) Global and Detailed Routing Tools:* It is hard to determine which combination of academic tools will lead to the best results considering the physical flow. Although some academic tools are well-known for being state-of-the-art, we cannot guess how those tools will interact with each other. To investigate such interaction, in our recent work [30] we evaluated 12 different flows built by combining three placements (original placement by contest, DreamPlace [31] and EhPlacer [32]), two GR engines (CUGR [4] and FastRoute [33]), and two DR tools (DRCU [5] and TritonRoute [7]). These flows were evaluated by using the ISPD Contest 2018 [13] and ISPD Contest 2019 [14] benchmarks. Among all these flows, the best final results were obtained by the flow with Contest placement + CUGR + TritonRoute, whereas the second-best flow was DreamPlace + CUGR + TritonRoute. Therefore, in this work, we use the best flow to serve as our baseline. Additionally, we also compare our results with the second-best flow.

[2]Min Area counts the number of occurrences in which a metal shape area is less than the specified value for a layer. Therefore, it may occur multiple times for a single net.

### C. Quality Evaluation of ILPGRC

This section presents the experimental results. Table IV brings the results obtained by using the ILPGRC technique and the baseline results (i.e., without using ILPGRC), employing the evaluation flow depicted in Fig. 8. For each circuit listed in the leftmost column, the table brings the technology node, the number and the percentage of movements, the total number of VIAs, the number of off-track VIAs, the total wirelength, the off-track wirelength, the wrong way wirelength, the number of metal short violations, the number of min area violations, the number of metal spacing violations, and the number of open nets. To make the comparison easy, the columns labeled as Basel bring the absolute number of VIAs and wirelength obtained by the baseline flow, whereas the columns labeled as ILPGRC Imp % show the percentage of improvement achieved when the ILPGRC technique is applied. Thus, a positive value in any of these columns means that ILPGRC has improved the given metric with respect to the baseline result.

The results in Table IV show that ILPGRC performed the best in the total number of VIAs for all circuits. The average reduction in the total number of VIAs is 4.69%, which was reached by moving only 1.98% of cells. This improvement comes from the path cost considered in (13) and from the multiple routing candidates generated by function "*create_nets_candidates*" (lines 4 and 11 in Algorithm 3). This function generates multiple routing paths for each net. Each of these paths is routed through different metal layers, and as a consequence, they have different costs. Therefore, the ILP model tries to minimize the whole routing cost (12) by selecting, whenever it is possible, the routing paths that use the lower metal layers. It is worth noting that applying ILPGRC has enabled TritonRoute to reduce the off-track VIAs by 5.61% on average.

The ILPGRC technique achieves free DRV solutions in 17 out of 20 benchmarks, outperforming the reference flow by one benchmark (19_t9). It eliminates open nets and min area violations, with only a few shorts and spacing violations remaining. As a result, ILPGRC offers an improved solution after DR, requiring minimal intervention from CAD engineers.

Moving our attention to the wirelength metric, we can observe significant improvements in the off-track and wrong-way wirelength when the ILPGRC was applied. This is highly correlated with the fact that the ILPGRC technique does not allow a net to be routed outside the tracks or in the nonpreferred direction. Finally, although the ILPGRC technique was not able to lead TritonRoute to reduce wirelength, it is important to note that the wirelength was increased by

TABLE IV
EXPERIMENTAL RESULTS FOR ISPD 2018 AND ISPD 2019 BENCHMARKS EVALUATED AFTER THE DR SOLUTION

| Circuits ISPD | Node (nm) | Movements | | Vias | | | | Wirelength (mm) | | | | | | Design Rule Violations (DRVs) | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | Total | | Off Track | | Total | | Off Track | | Wrong Way | | Shorts | | Min Area | | Spacing | | Open Nets | |
| | | # | % | Basel. (K) | ILPGRC Imp. % | Basel. | ILPGRC Imp. % | Basel. | ILPGRC Imp. % | Basel. | ILPGRC Imp. % | Basel. | ILPGRC Imp. % | Basel. | ILPGRC | Basel. | ILPGRC | Basel. | ILPGRC | Basel. | ILPGRC |
| 18_t1 | 45 | 642 | 7.23 | 36 | **1.92** | 171 | **9.36** | 172 | -0.46 | 0.35 | **72.76** | 3.37 | **8.14** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 18_t2 | | 922 | 2.57 | 360 | **1.93** | 2,164 | **6.24** | 3,134 | -0.58 | 3.99 | **63.83** | 31.81 | **13.32** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 18_t3 | | 1,042 | 2.90 | 363 | **1.54** | 2,197 | **6.87** | 3,505 | -0.48 | 4.83 | **57.20** | 32.36 | **14.08** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 18_t4 | 32 | 2,264 | 3.14 | 725 | **2.68** | 17,336 | -0.72 | 5,240 | -0.39 | 9.98 | **51.22** | 27.84 | **24.31** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 18_t5 | | 1,011 | 1.41 | 860 | **8.53** | 14,946 | -0.58 | 5,481 | -0.39 | 3.82 | **76.68** | 17.13 | -2.64 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 18_t6 | | 108 | 0.10 | 1,278 | **8.20** | 22,899 | -1.31 | 7,100 | -0.38 | 4.19 | **63.62** | 23.79 | -6.62 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 18_t7 | | 1,544 | 0.86 | 2,096 | **6.59** | 28,490 | **0.13** | 12,918 | -0.30 | 7.07 | **60.20** | 37.40 | **1.36** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 18_t8 | | 1,399 | 0.73 | 2,147 | **6.20** | 28,508 | **0.13** | 13,033 | -0.25 | 7.95 | **59.27** | 39.06 | **2.86** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 18_t9 | | 1,227 | 0.64 | 2,150 | **7.08** | 28,478 | **0.09** | 10,808 | -0.44 | 7.49 | **66.47** | 37.70 | **3.89** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 18_t10_nf | | 1,267 | 0.44 | 2,308 | **7.83** | 32,184 | **1.54** | 13,542 | -0.12 | 12.18 | **65.37** | 51.80 | **14.30** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 19_t1 | | 99 | 1.11 | 38 | **4.54** | 3,965 | **0.53** | 126 | -0.70 | 0.44 | **17.03** | 1.81 | **5.11** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 19_t2 | | 985 | 1.37 | 798 | **1.62** | 132,593 | **0.57** | 4,940 | -0.36 | 14.28 | **12.55** | 32.04 | **16.96** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 19_t3 | | 343 | 4.14 | 63 | **1.66** | 6,083 | **2.98** | 164 | -0.42 | 0.48 | **17.06** | 3.17 | **13.11** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 19_t6 | | 2,041 | 1.13 | 1,972 | **2.19** | 46,333 | **4.29** | 13,117 | -0.05 | 12.20 | **28.89** | 57.30 | **18.16** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 19_t7 | | 2,263 | 0.63 | 3,789 | **2.91** | 255,196 | **1.11** | 24,198 | -0.05 | 34.29 | **42.32** | 169.90 | **19.47** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 19_t8 | | 5,755 | 1.07 | 6,176 | **5.19** | 382,323 | **0.83** | 37,073 | -0.48 | 44.78 | **38.48** | 174.34 | -2.67 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 19_t9 | | 11,135 | 1.24 | 10,265 | **5.88** | 637,914 | **0.79** | 56,021 | -0.48 | 77.37 | **35.66** | 295.01 | -2.74 | 0 | 0 | 3 | **0** | 2 | **0** | 0 | 0 |
| 19_t10 | | 10,951 | 1.22 | 9,574 | **2.61** | 637,558 | **0.92** | 55,325 | -0.30 | 85.11 | **39.45** | 423.47 | **14.00** | 1 | 39 | 0 | 0 | 15 | 46 | 0 | 0 |
| 19_t4* | 65 | 9,424 | 6.44 | 1,050 | **11.74** | 3,874 | **71.17** | 5,428 | -6.40 | 31.11 | **80.47** | 259.16 | **81.00** | 170K | **3** | 0 | 0 | 90K | **5** | 1 | **0** |
| 19_t5* | | 1,881 | 6.50 | 151 | **0.12** | 155 | **10.97** | 927 | -3.28 | 1.30 | **41.52** | 15.65 | **47.74** | 3K | **7** | 0 | 0 | 31 | **0** | 0 | 0 |
| Average | | | 1.98 | | **4.69** | | **5.61** | | -0.83 | | **48.28** | | **14.47** | | | | | | | | |
| Avg. 18 | | | 2.00 | | **5.25** | | **2.17** | | -0.38 | | **63.66** | | **7.30** | | | | | | | | |
| Avg. 19 | | | 2.48 | | **3.85** | | **9.42** | | -1.25 | | **35.34** | | **21.01** | | | | | | | | |

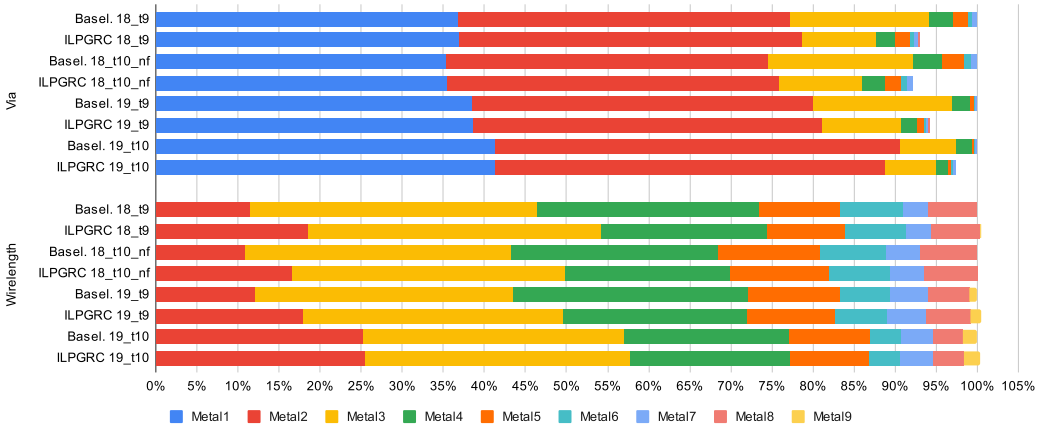


Fig. 9. VIAs and wirelength distribution after DR for circuits ispd18_test9, ispd18_test10_nf, ispd19_test9, and ispd19_test10.

only 0.83% on average, and in most cases (15 out of 20 benchmarks), the degradation in wirelength is less than 0.5%.

The counterpart of reducing the total number of VIAs can be an increase in the wirelength in the lower layers. The VIA and Wirelength distributions for the baseline and ILPGRC for circuits 18_t9, 18_t10_nf, 19_t0, and 19_t10 are presented in Fig. 9. We extract these values from the DEF file generated by TritonRoute using a commercial tool. The bars shorter than 100% mean that the ILPGRC could improve this metric for the specific circuit. In contrast, bars longer than 100% mean that ILPGRC degrades this metric for the given circuit.

Considering the VIA distribution, we can observe that the percentage of VIAs in Metal 1 is very similar for both ILPGRC and baseline. On the other hand, the difference in the number of VIAs comes from the upper layers. Considering the wirelength distribution, we can observe that both approaches result in the same length of metal in all layers. Therefore, we can conclude that ILPGRC is able to reduce the number of VIAs without impacting the circuit congestion.

We conducted an additional experiment in which the (complete) ILPGRC technique was applied five times before the DR step. This experiment aims to evaluate the impact of multiple iterations since each net permits only one cell movement in the ILP model. By running ILPGRC 5 times, it is possible to slightly reduce the number of VIAs for some circuits, but the average (3.97%) is worst than a single iteration (4.69%). Running ILPGRC 5 times reduces the impact on the wirelength (−0.19 on average). However, the total runtimes of the five iterations are much longer (5.56 times on average) than those of the single execution of the ILPGRC. Therefore, we consider that the iterative version is not worth it. In future work, we plan to make a convergence condition to determine if another iteration should be executed.

To the best of our knowledge, CRP [11] and its respective extension CRP 2.0 [20] are the only available works able to do routing with cell movement using realistic benchmarks with technology node information. We acquired the data of CRP 2.0 from their authors. This way, we could evaluate the quality of DR solutions in CRP 2.0 and ILPGRC.

Table V presents how ILPGRC performs using the same initial routing and baseline of CRP 2.0 technique [20]. The values in this table are the percentages of improvement in relation to the baseline. So, the higher values are better, and the best value for each circuit and metric is highlighted in bold.

We can see that, on average, ILPGRC is superior in reducing VIAs but slightly degrades wirelength, which suggests that our work is more aggressive in the VIA reduction. ILPGRC can produce a valid solution for all benchmarks, whereas CRP 2.0 fails in the 19_t4 benchmark. Also, ILPGRC only generates a few DRVs that can be easily solved by hand, while CRP 2.0 leaves 24k shorts in circuit 18_t10. It is worth noting that most

TABLE V
RESULTS COMPARING ILPGRC WITH CRP 2.0 [20]

| Circuits ISPD | Node (nm) | Vias (Imp. %) | | | | Wirelength (Imp. %) | | | | | | Design Rule Violations (DRVs) | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Total | | Off Track | | Total | | Off Track | | Wrong Way | | Shorts | | Min Area | | Spacing | | Open Nets | |
| | | ILPGRC | CRP 2.0 | ILPGRC | CRP 2.0 | ILPGRC | CRP 2.0 | ILPGRC | CRP 2.0 | ILPGRC | CRP 2.0 | ILPGRC | CRP 2.0 | ILPGRC | CRP 2.0 | ILPGRC | CRP 2.0 | ILPGRC | CRP 2.0 |
| 18_t1 | 45 | **4.16** | 0.21 | **1.95** | 0.65 | -1.03 | **-0.05** | **80.33** | -0.37 | **8.88** | -1.16 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 18_t2 | | **5.08** | 1.18 | **5.32** | -2.71 | -0.76 | **-0.2** | **76.55** | -5 | **17.71** | -7.23 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 18_t3 | | **4.71** | 1.21 | **7.00** | -0.84 | -0.48 | **-0.13** | **69.74** | -7.23 | **16.73** | -7.75 | 0 | 0 | 0 | 0 | 1 | **0** | 0 | 0 |
| 18_t4 | 32 | **2.18** | -0.05 | -0.55 | **0.21** | -0.28 | **0.01** | **46.76** | -1.64 | **25.73** | -1.89 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 18_t5 | | **6.83** | 0.78 | -0.77 | **-0.11** | -0.43 | **-0.01** | **70.05** | -3.61 | **-3.69** | -6.75 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 18_t6 | | **6.64** | 0.14 | -1.38 | **-0.14** | -0.43 | **0.04** | **71.14** | -0.78 | -4.89 | **-1.23** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 18_t7 | | **5.73** | 0.98 | **0.13** | -0.09 | -0.34 | **0.03** | **70.59** | -4.7 | **1.23** | -8.93 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 18_t8 | | **5.60** | 1.07 | **0.14** | -0.12 | -0.34 | **0.09** | **70.14** | -2.23 | **2.84** | -7.68 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 18_t9 | | **5.74** | 1.17 | **0.13** | -0.05 | -0.52 | **-0.03** | **74.57** | -1 | **2.82** | -7.51 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 18_t10 | | | **0.1** | | **-0.56** | | **0** | | **-3.44** | | **-2.61** | | 24K | | 0 | | 0 | | 0 |
| 18_t10_nf | | **6.38** | | **2.10** | | **-0.20** | | **72.92** | | **12.18** | | **1** | | 0 | | 0 | | 0 | |
| 19_t1 | | **8.87** | -2.45 | **1.72** | **1.32** | -0.64 | **0.22** | -19.38 | **17.53** | 0.23 | **14.18** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 19_t2 | | **2.41** | -0.59 | **0.62** | -0.32 | -0.16 | **0.01** | **23.32** | **14.79** | **19.21** | 15.7 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 19_t3 | | **2.17** | 0.79 | **2.93** | 0.38 | -0.07 | **0.34** | **26.01** | 11.46 | **9.30** | 7.32 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 19_t6 | | **6.36** | -1.41 | **5.09** | -1.39 | **0.06** | -0.22 | **36.78** | **34.77** | 18.42 | **20.75** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 19_t7 | | **3.42** | **16.14** | **1.11** | 0.46 | 0.05 | **0.58** | **42.65** | 6.84 | **19.47** | -6.18 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 19_t8 | | 8.76 | **11.61** | **1.36** | **0.92** | -0.52 | **0.76** | **-76.64** | **14.15** | -8.95 | **9.82** | 0 | 0 | 0 | 0 | **0** | 5 | 0 | 0 |
| 19_t9 | | 9.59 | **11.51** | **1.21** | **0.9** | -0.56 | **0.95** | **-62.75** | **12.56** | -8.73 | **8.19** | 0 | 0 | 0 | 0 | **0** | 8 | 0 | 0 |
| 19_t10 | | 2.93 | **18.29** | 0.93 | **1** | -0.19 | **0.84** | **39.83** | 9.06 | **13.77** | -10.76 | 53 | **0** | 0 | 0 | **54** | 73 | | 0 |
| 19_t4* | 65 | **7.90** | Fail | **-19.60** | Fail | **2.58** | Fail | **41.75** | Fail | **2.12** | Fail | **31** | Fail | **0** | Fail | **159** | Fail | **0** | Fail |
| 19_t5* | | 6.74 | **7.62** | 19.31 | **37.97** | **1.08** | **-1.49** | **59.86** | 34.51 | **23.91** | -10.47 | 74 | **0** | **0** | 3 | **49** | **2** | 0 | 0 |
| Average | | **5.61** | 3.59 | **1.44** | **1.97** | -0.16 | **0.09** | **40.71** | 6.61 | **8.41** | -0.22 | | | | | | | | |
| Avg. 18 | | **5.31** | 0.68 | **1.41** | -0.38 | -0.48 | **-0.02** | **70.28** | -3 | **7.95** | -5.27 | | | | | | | | |
| Avg. 19 | | **5.92** | **6.83** | **1.47** | **4.58** | 0.16 | **0.22** | **11.14** | **17.3** | **8.87** | 5.4 | | | | | | | | |

TABLE VI
EXPERIMENTAL RESULTS FOR ISPD 2018 AND ISPD 2019 BENCHMARKS COMPARED WITH DREAMPLACE (DRPL) + CUGR + TRITONROUTE. THE RESULTS WERE EVALUATED AFTER THE DR SOLUTION

| Circuits ISPD | Node (nm) | Movements | | VIAs | | | | Wirelength (mm) | | | | | | Design Rule Violations (DRVs) | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | Total | | Off Track | | Total | | Off Track | | Wrong Way | | Shorts | | Min Area | | Spacing | | Open Nets | |
| | | # | % | DrPl (K) | ILPGRC Imp. % | DrPl | ILPGRC Imp. % | DrPl | ILPGRC Imp. % | DrPl | ILPGRC Imp. % | DrPl | ILPGRC Imp. % | DrPl | ILPGRC | DrPl | ILPGRC | DrPl | ILPGRC | DrPl | ILPGRC |
| 18_t1 | 45 | 893 | 10.06 | 38 | **4.85** | 196 | **26.02** | 176 | -1.62 | 0.58 | **83.85** | 3.50 | **19.52** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 18_t2 | | 1,180 | 3.29 | 388 | **5.29** | 4,405 | **56.19** | 3,151 | -1.36 | 6.57 | **72.76** | 32.25 | **22.72** | 0 | 0 | 0 | 0 | 0 | 0 | 1 | **0** |
| 18_t3 | | 1,287 | 3.58 | 382 | **4.92** | 4,430 | **55.58** | 3,529 | -0.94 | 6.67 | **60.02** | 32.21 | **18.44** | 0 | 0 | 0 | 0 | 0 | 0 | 2 | **0** |
| 18_t4 | 32 | 3,194 | 4.43 | 722 | **1.29** | 17,363 | -1.70 | 5,313 | -0.27 | 10.49 | **53.74** | 31.47 | **36.64** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 18_t5 | | 4,976 | 6.92 | 889 | **12.49** | 19,637 | **20.97** | 5,580 | -0.51 | 4.03 | **78.59** | 18.17 | **3.79** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 18_t6 | | 720 | 0.67 | 1,382 | **13.06** | 28,522 | **18.59** | 7,752 | -0.56 | 4.89 | **64.69** | 24.67 | -1.28 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 18_t7 | | 3,001 | 1.67 | 2,214 | **10.97** | 28,557 | **0.29** | 13,055 | -0.52 | 8.03 | **62.60** | 37.85 | **3.62** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 18_t8 | | 2,899 | 1.51 | 2,264 | **10.71** | 28,595 | **0.35** | 13,137 | -0.39 | 8.92 | **63.05** | 38.90 | **6.01** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 18_t9 | | 2,716 | 1.41 | 2,265 | **11.05** | 28,862 | **1.32** | 10,884 | -0.66 | 8.01 | **65.29** | 37.61 | **6.31** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 18_t10_nf | | Fail | Fail | Fail | Fail | Fail | Fail | Fail | Fail | Fail | Fail | Fail | Fail | Fail | Fail | Fail | Fail | Fail | Fail | Fail | Fail |
| 19_t1 | | 147 | 1.66 | 39 | **4.18** | 3,990 | **2.21** | 132 | -0.93 | 0.42 | **16.12** | 1.97 | **11.50** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 19_t2 | | 1,397 | 1.94 | 791 | -1.63 | 133,119 | **0.43** | 4,992 | -0.64 | 14.89 | **12.06** | 37.34 | **22.77** | 0 | 0 | 0 | 0 | 3 | 3 | 0 | 0 |
| 19_t3 | | Fail | Fail | Fail | Fail | Fail | Fail | Fail | Fail | Fail | Fail | Fail | Fail | Fail | Fail | Fail | Fail | Fail | Fail | Fail | Fail |
| 19_t6 | | 3,091 | 1.72 | 1,948 | **0.81** | 54,757 | **18.26** | 13,136 | -0.48 | 12.22 | **30.34** | 68.90 | **27.30** | 0 | 0 | 6 | **0** | 16 | **0** | 0 | 0 |
| 19_t7 | | 5,236 | 1.46 | 3,726 | **1.28** | 264,367 | **4.35** | 24,381 | **0.00** | 40.34 | **50.82** | 230.98 | **39.24** | 0 | 0 | 0 | 0 | 10 | **0** | 0 | 0 |
| 19_t8 | | 9,191 | 1.70 | 6,242 | **4.77** | 403,575 | **4.33** | 37,377 | -0.72 | 46.99 | **38.90** | 203.45 | **5.59** | 0 | 0 | 0 | 0 | 14 | **0** | 0 | 0 |
| 19_t9 | | 16,815 | 1.87 | 10,413 | **5.62** | 671,099 | **4.21** | 56,561 | -0.64 | 80.70 | **36.99** | 337.54 | **5.06** | 0 | 0 | 0 | 0 | 24 | **0** | 0 | 0 |
| 19_t10 | | 20,185 | 2.24 | 9,428 | **0.99** | 663,260 | **3.91** | 56,004 | -0.31 | 96.94 | **46.42** | 572.25 | **32.28** | 0 | 613K | 0 | 0 | 66 | 146 | 0 | 0 |
| 19_t4* | 65 | Fail | Fail | Fail | Fail | Fail | Fail | Fail | Fail | Fail | Fail | Fail | Fail | Fail | Fail | Fail | Fail | Fail | Fail | Fail | Fail |
| 19_t5* | | 2,306 | 7.97 | 158 | **3.35** | 169 | -17.16 | 930 | **0.15** | 1.57 | **50.53** | 19.06 | **49.13** | 2,890K | **2,226K** | 0 | 0 | 1K | 1K | 0 | 0 |
| Average | | | 3.26 | | **5.53** | | **11.66** | | -0.61 | | **52.16** | | **18.16** | | | | | | | | |
| Avg. 18 | | | 3.46 | | **8.29** | | **19.73** | | -0.76 | | **67.18** | | **12.86** | | | | | | | | |
| Avg. 19 | | | 3.07 | | **2.42** | | **2.57** | | -0.45 | | **35.27** | | **24.11** | | | | | | | | |

of the gain from CRP 2.0 comes from the circuits 19_t5, 19_t6, 19_t7, 19_t8, 19_t9, and 19_t10, while for the other circuits, CRP 2.0 only achieves slight gains. Furthermore, ILPGRC is superior in off-track and Wrong Way Wirelengh.

We conducted another experiment to evaluate the efficacy of the ILPGRC technique under a different scenario. This experiment relied on the evaluation flow in Fig. 8, except that we used DreamPlace [31] to generate new placement solutions for the ISPD Contests 2018 and 2019 circuits. The flow DreamPlace + CUGR + TritonRoute resulted in the second-best placement to DR flow reported in [30]. Table VI presents the results for this experiment. The columns labeled as DrPl bring the absolute values for the new baseline (DreamPlace + CUGR + TritonRoute) whilst the columns labeled as ILPGRC Imp % show the percentage of improvement achieved when the ILPGRC technique is applied. The rest of the table is organized in a similar fashion as Table IV. The baseline flow fails for three circuits: 18_t10_nf, 19_t3, and 19_t4. In particular, 18_t10_nf was not legalized after the execution of DreamPlace, whereas 19_t3 and 19_t4 resulted in a time-out of 48 h when performing TritonRoute.

The results in Table VI show that applying ILPGRC resulted in VIA reduction for all circuits, except 19_t2. The average reduction is 5.53% and was achieved by moving on average 3.26% of cells. As in the previous experiment, applying ILPGRC has enabled TritonRoute to reduce the off-track VIAs, in this case, by 11.66% on average.

Regarding DRVs, the baseline flow generated violations for nine circuits: 18_t2, 18_t3, 19_t2, 19_t5, 19_t6, 19_t7, 19_t8, 19_t9, and 19_10. On the other hand, ILPGRC flow produced violations in only five circuits: 19_t2, 19_t5, 19_t6, and 19_10. The baseline produced open nets in circuits 18_t2 and 18_t3; no open nets are reported when ILPGRC was applied.

ILPGRC improves the wirelength in two circuits (19_t7 and 19_t5) while not increasing it by a maximum of 1.62%. On average, the wirelength is increased by only 0.61%. Additionally, ILPGRC resulted in significant improvements in the off-track and wrong-way wirelength in this experiment. This behavior shows that the ILPGRC technique leads TritonRoute to choose better routing paths for the nets.

### D. Runtime Analysis of ILPGRC

The Checkered Paneling strategy is essential to make the ILPGRC technique viable since it allows for reducing the problem size and breaks data dependency, thus enabling parallel execution. For example, for the largest circuit in our experiment (19_t10) containing 899 K cells and 895 K nets, we have less than ten ILP instances with more than 1 K cells and 5K nets. For all the other ILP instances (5584) the number

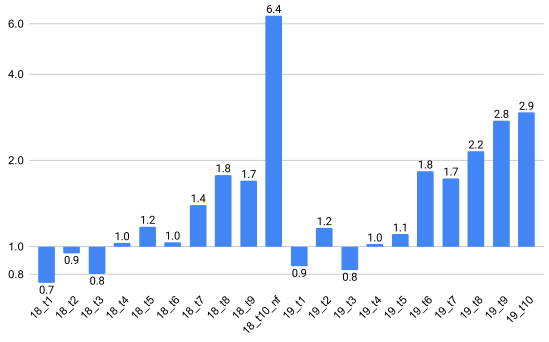


Fig. 10. Speedup when using parallel mode with eight threads (*y*-axis) for each circuit (*x*-axis).

TABLE VII
RUNTIME IN SECONDS OF THE ILPGRC IN SEQUENTIAL AND PARALLEL MODE. THE BEST RUNTIME FOR EACH CIRCUIT IS HIGHLIGHTED IN BOLD

| Circuits ISPD | Sequential | Parallel | Circuits ISPD | Sequential | Parallel |
|---|---|---|---|---|---|
| 18_t1 | **22** | 29 | 19_t1 | **24** | 28 |
| 18_t2 | **45** | 48 | 19_t2 | 127 | **109** |
| 18_t3 | **49** | 62 | 19_t3 | **12** | 14 |
| 18_t4 | 112 | **109** | 19_t4 | 157 | **154** |
| 18_t5 | 135 | **115** | 19_t5 | 30 | **27** |
| 18_t6 | 190 | **184** | 19_t6 | 614 | **335** |
| 18_t7 | 552 | **396** | 19_t7 | 1172 | **677** |
| 18_t8 | 673 | **378** | 19_t8 | 2175 | **1011** |
| 18_t9 | 510 | **300** | 19_t9 | 4928 | **1791** |
| 18_t10_nf | 2644 | **413** | 19_t10 | 4605 | **1568** |

of cells and nets is smaller. Therefore, the circuit partitioning makes it possible to build and solve the ILP model in an acceptable runtime. It is important to mention that, in contrast with an ILP model for GR which must consider only the net candidates, in ILPGRC the ILP model must consider all candidate nets and cells that are entirely inside the panel.

In ILPGRC only the Checkered panels within the same level and of the same color are run in parallel. This guarantees a deterministic solution for all levels, hence, making the results of the parallel version exactly the same as those of the sequential version. Fig. 10 displays the speedup achieved when the Checkered panels are run in parallel using eight threads, assuming the sequential execution as a baseline.

The speedup difference among the circuits is very noticeable in Fig. 10, ranging from less than 1× for the smaller circuits to more than 6× in the case of circuit 18_t10_nf. As shown in Table VII, for the smaller circuits, such as 18_t1, 18_t2, 18_t3, 19_t1, and 19_t3, the runtime of the parallel version is greater than that of the sequential version. This is because the overhead time to create and synchronize the threads supersedes the benefits of running ILP in parallel. On the contrary, for the larger circuits, with more than 290K cells and 182K nets (18_t10_nf, and 19_t7 to 19_t10), the time to create and synchronize the threads is less significant compared to the time to create and solve the ILP model.

## VI. CONCLUSION

In this article, we proposed ILPGRC, an ILP-based technique to optimize the GR by simultaneously moving cells and routing nets, maintaining the legality of the circuit, and avoiding DRVs. We also proposed a checkered paneling strategy, which reduces the input size of the ILP model, making this approach scalable. The Checkered paneling strategy also enables the execution of multiple ILP models in parallel, providing a speedup for large circuits. ILPGRC allows the router to move cells to optimize different routing objectives, such as the total number of VIAs, wirelength, and the number of DRVs. This is in contrast with the traditional approach that considers fixed placement during routing steps. This approach can assist other routing algorithms in working collaboratively with the placement algorithms, making both placement and routing more agile and efficient. Unlike other related work, we evaluated the impacts of the proposed technique after the DR. The experimental results show that ILPGRC reduces the number of VIAs by 4.69%, moving only 1.98% of the cells and degrading the wirelength less than 1%, on average. In addition, the detailed router reported no open nets after the proposed technique, while the number of DRVs was reduced in two out of three cases.

## REFERENCES

[1] A. B. Kahng, J. Lienig, I. L. Markov, and J. Hu, *VLSI Physical Design: From Graph Partitioning to Timing Closure*, 1st ed. Dordrecht, The Netherlands: Springer, 2011.
[2] S. Held, D. Müller, D. Rotter, R. Scheifele, V. Traub, and J. Vygen, "Global routing with timing constraints," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 37, no. 2, pp. 406–419, Feb. 2018.
[3] P. Tu, W.-K. Chow, and E. F. Young, "Timing driven routing tree construction," in *Proc. SLIP*, 2017, pp. 1–8.
[4] J. Liu, C. Pui, F. Wang, and E. Young, "CUGR: Detailed-Routability-driven 3D global routing with probabilistic resource model," in *Proc. DAC*, 2020, pp. 1–6.
[5] G. Chen, C. Pui, H. Li, and E. F. Y. Young, "Dr. CU: Detailed routing by sparse grid graph and minimum-area-captured path search," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 39, no. 9, pp. 1902–1915, Sep. 2020.
[6] A. B. Kahng, L. Wang, and B. Xu, "TritonRoute: An initial detailed router for advanced VLSI technologies," in *Proc. ICCAD*, Nov. 2018, pp. 1–8.
[7] A. B. Kahng, L. Wang, and B. Xu, "TritonRoute: The open-source detailed router," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 40, no. 3, pp. 547–559, Mar. 2021.
[8] A. B. Kahng, L. Wang, and B. Xu, "TritonRoute-WXL: The open-source router with integrated DRC engine," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 41, no. 4, pp. 1076–1089, Apr. 2022.
[9] T. A. Fontana et al., "ILP-based global routing optimization with cell movements," in *Proc. ISVLSI*, 2021, pp. 25–30.
[10] F. Wang, L. Liu, J. Chen, J. Liu, X. Zang, and M. D. Wong, "Starfish: An efficient P&R co-optimization engine with A*-based partial rerouting," in *Proc. ICCAD*, 2021, pp. 1–9.
[11] E. Aghaeekiasaraee et al., "CR&P: An efficient co-operation between routing and placement," in *Proc. DATE*, 2022, pp. 772–777.
[12] P. Zou, Z. Cai, Z. Lin, C. Ma, J. Yu, and J. Chen, "Incremental 3-D global routing considering cell movement and complex routing constraints," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 42, no. 6, pp. 2016–2029, Jun. 2023.
[13] S. Mantik, G. Posser, W.-K. Chow, Y. Ding, and W.-H. Liu, "ISPD 2018 initial detailed routing contest and benchmarks," in *Proc. ISPD* 2018, pp. 140–143.
[14] W.-H. Liu, S. Mantik, W.-K. Chow, Y. Ding, A. Farshidi, and G. Posser, "ISPD 2019 initial detailed routing contest and benchmark with advanced routing rules," in *Proc. ISPD*, 2019, pp. 147–151.
[15] K.-S. Hu, M.-J. Yang, T.-C. Yu, and G.-C. Chen, "ICCAD-2020 CAD contest in routing with cell movement," in *Proc. ICCAD*, 2020, pp. 1–4.
[16] K.-S. Hu, M.-J. Yang, and T.-C. Yu. "Problem B: Routing with cell movement advanced." 2021. [Online]. Available: http://iccad-contest.org/2021/Problems/Problam_B_20210220_1540.pdf
[17] Z. Huang et al., "Detailed placement and global routing co-optimization with complex constraints," *Electronics*, vol. 11, no. 1, p. 51, 2021.

[18] Z. Zhu, F. Shen, Y. Mei, Z. Huang, J. Chen, and J. Yang, "A robust global routing engine with high-accuracy cell movement under advanced constraints," in *Proc. ICCAD*, 2022, pp. 1–9.
[19] X. Zang, F. Wang, J. Liu, and M. D. Wong, "ATLAS: A two-level layer-aware scheme for routing with cell movement," in *Proc. ICCAD*, 2022, pp. 1–7.
[20] E. Aghaeekiasaraee et al., "CRP2.0: A fast and robust cooperation between routing and placement in advanced technology nodes," *ACM Trans. Design Autom. Electron. Syst.*, to be published.
[21] X. He, W.-K. Chow, and E. F. Young, "SRP: Simultaneous routing and placement for congestion refinement," in *Proc. ISPD*, 2013, pp. 108–113.
[22] P. Spindler, U. Schlichtmann, and F. M. Johannes, "Abacus: Fast legalization of standard cell circuits with minimal movement," in *Proc. ISPD*, 2008, pp. 47–53.
[23] Y. Manolopoulos, A. Nanopoulos, A. Papadopoulos, and Y. Theodoridis, *R-Trees: Theory and Applications*. London, U.K.: Springer, 2006.
[24] M. Pan, N. Viswanathan, and C. Chu, "An efficient and effective detailed placement algorithm," in *Proc. ICCAD*, 2005, pp. 48–55.
[25] C. Chu and Y. Wong, "FLUTE: Fast lookup table based rectilinear steiner minimal tree algorithm for VLSI design," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 27, no. 1, pp. 70–83, Jan. 2008.
[26] "CPLEX optimizer 12.8." IBM. 2018. [Online]. Available: www.ibm.com/analytics/data-science/prescriptive-analytics/cplex-optimizer
[27] "Ophidian: An open source library for physical design research and teaching." Embedded Computing Lab. 2022. [Online]. Available: https://gitlab.com/eclufsc/ophidian
[28] B. Schling, *The Boost C++ Libraries*, 1st ed. XML Press, Laguna Hills, CA, USA, 2011.
[29] "Innovus." Cadence. 2022. [Online]. Available: https://www.cadence.com/ko_KR/home/tools/digital-design-and-signoff/soc-implementation-and-floorplanning/innovus-implementation-system.html
[30] T. A. Fontana, R. Netto, S. F. Almeida, E. Aghaeekiasaraee, L. Behjat, and J. L. Güntzel, "Towards a reference place and route flow for academic research," *J. Integr. Circuits Syst.*, vol. 17, no. 3, pp. 1–12, 2022.
[31] Y. Lin, S. Dhar, W. Li, H. Ren, B. Khailany, and D. Z. Pan, "DREAMPlace: Deep learning toolkit-enabled GPU acceleration for modern VLSI placement," in *Proc. DAC*, 2020, pp. 1–6.
[32] N. K. Darav, A. Kennings, A. F. Tabrizi, D. Westwick, and L. Behjat, "Eh? Placer: A high-performance modern technology-driven placer," *ACM Trans. Design Autom. Electron. Syst.*, vol. 21, no. 3, pp. 1–27, 2016.
[33] Y. Xu, Y. Zhang, and C. Chu, "FastRoute 4.0: Global router with efficient via minimization," in *Proc. ASPDAC*, 2009, pp. 576–581.

**Tiago Augusto Fontana** received the B.S. and M.S. degrees in computer science from the Federal University of Santa Catarina, Florianópolis, Brazil, in 2016 and 2018, respectively, where he is currently pursuing the Ph.D. degree in computer science.

Since 2015, he has been a member of the Embedded Computing Laboratory, Federal University of Santa Catarina, conducting research activities on physical design, focusing mainly on the signal-routing and placement steps of standard cells.

**Erfan Aghaeekiasaraee** received the B.Sc. degree from the University of Guilan, Rasht, Iran, in 2012, the M.Sc. degree in computer engineering from the University of Mohaghegh Ardabili, Ardabil, Iran, in 2015, and the Ph.D. degree in electrical engineering from the University of Calgary, Calgary, AB, Canada, in 2023.

His research interest focuses on designing and developing optimized algorithms in physical design flow, such as routing and placement.

**Renan Netto** received the B.Sc., M.Sc., and Ph.D. degrees in computer science from the Federal University of Santa Catarina, Florianópolis, Brazil, in 2015, 2017, and 2021, respectively.

He is currently a Lead Software Engineer with Cadence, San Jose, CA, USA. His main interests are the application of machine learning for physical design applications.

**Sheiny Fabre Almeida** received the B.Sc. and M.Sc. degrees in computer science from the Federal University of Santa Catarina, Florianópolis, Brazil, in 2016 and 2019, respectively, where he is currently pursuing the Ph.D. degree.

His current research interests include machine learning and parallel optimizations for routability improvement during the physical design placement step.

Mr. Almeida was a recipient of the Third Place in ICCAD 2017 Multideck Standard Cell Legalization Contest.

**Upma Gandhi** received the B.Tech. degree in computer science from the University of Delhi, New Delhi, India, in 2017, and the M.Sc. degree in electrical and computer engineering from the University of Calgary, Calgary, AB, Canada, in 2019s, where she is currently pursuing the Ph.D. degree.

Her research focuses on optimizing the routing process of VLSI physical design step in VLSI using reinforcement learning algorithms.

**Laleh Behjat** (Senior Member, IEEE) received the Ph.D. degree from the University of Waterloo, Waterloo, ON, Canada, in 2002.

She is a Professor with the Department of Electrical and Software Engineering, University of Calgary, Calgary, AB, Canada, and the Natural Sciences and Engineering Council of Canada Research Chair for Women in Science and Engineering—Prairie Region. Her research focuses on developing mathematical techniques and software tools for automating the design of digital integrated circuits.

**José Luís Güntzel** (Senior Member, IEEE) received the M.Sc. and Ph.D. degrees in computer science from the Federal University of Rio Grande do Sul, Porto Alegre, Brazil, in 1993 and 2000, respectively.

He is a Full Professor with the Department of Informatics and Statistics, Federal University of Santa Catarina, Florianópolis, Brazil. His research interests include physical design automation of VLSI circuits and algorithms and VLSI architectures for video coding.