PARR: Pin-Access Planning and Regular Routing for Self-Aligned Double Patterning

XIAOQING XU, The University of Texas at Austin
BEI YU, The Chinese University of Hong Kong
JHIH-RONG GAO, CHE-LUN HSU, and DAVID Z. PAN, The University of Texas at Austin

Pin access has become one of the most difficult challenges for detailed routing in advanced technology nodes, for example, in 14nm and below, for which double-patterning lithography has to be used for manufacturing lower metal routing layers with tight pitches, such as M2 and M3. Self-aligned double patterning (SADP) provides better control on line edge roughness and overlay, but it has very restrictive design constraints and prefers regular layout patterns. This article presents a comprehensive pin-access planning and regular routing framework (PARR) for SADP friendliness. Our key techniques include precomputation of both intracell and intercell pin accessibility, as well as local and global pin-access planning to enable handshaking between standard cell-level pin access and detailed routing under SADP constraints. A pin access–driven rip-up and reroute scheme is proposed to improve the ultimate routability. Our experimental results demonstrate that PARR can achieve much better routability and overlay control compared with previous approaches.

Categories and Subject Descriptors: B.7.2 [**Hardware, Integrated Circuit**]: Design Aids

General Terms: Algorithms, Design, Performance

Additional Key Words and Phrases: SADP, regular layout, pin access, net deferring

ACM Reference Format:
Xiaoqing Xu, Bei Yu, Jhih-Rong Gao, Che-Lun Hsu, and David Z. Pan. 2016. PARR: Pin-access planning and regular routing for self-aligned double patterning. ACM Trans. Des. Autom. Electron. Syst. 21, 3, Article 42 (May 2016), 21 pages.
DOI: http://dx.doi.org/10.1145/2842612

1. INTRODUCTION

In sub-20*nm* technology node, pin access has become a critical challenge for detailed routing [Hsu et al. 2014]. Due to the density increase or area reduction of the technology scaling, a limited number of routing tracks are available for the standard cell (SC) design. It makes the local SC I/O pin access challenging because the access points of each pin available for the detailed router are limited and they interfere with each other due to the extreme pitch scaling [Hsu et al. 2014; Xu et al. 2014]. Furthermore, the continued technology scaling of the lower routing layers in 14*nm* node and beyond depends on the complex design-for-manufacturability (DFM) strategies [Pan et al. 2013]. Extreme regular layout towards 1D gridded design [Smayling et al. 2013; Zhang and

This work is supported in part by the National Science Foundation, Semiconductor Research Corporation, and The Chinese University of Hong Kong (CUHK) Direct Grant for Research.
Authors' addresses: X. Xu, J.-R. Gao, C.-L. Hsu, and D. Z. Pan, Electrical and Computer Engineering Department, University of Texas at Austin, Austin, TX 78712; emails: {xiaoqingxu, jrgao, chsu1, dpan}@cerc.utexas.edu; B. Yu, Department of Computer Science and Engineering, The Chinese University of Hong Kong, NT, Hong Kong; email: byu@cse.cuhk.edu.hk.

DOI: http://dx.doi.org/10.1145/2842612

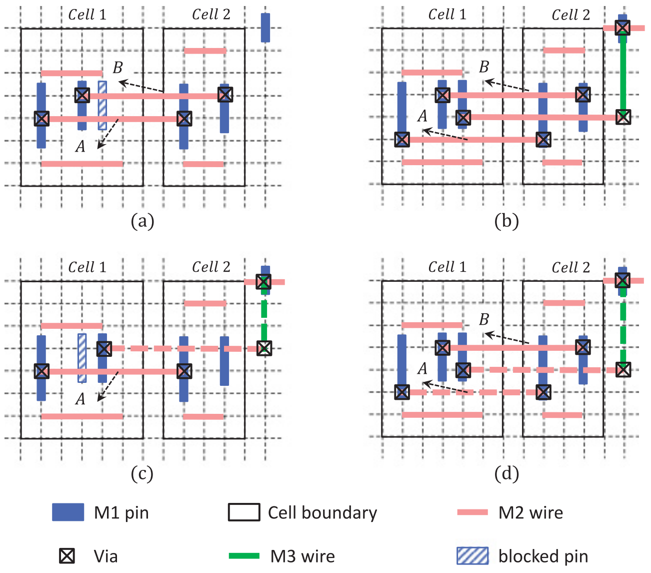

Fig. 1. Pin access for detailed routing: (a) pin-access failure; (b) pin-access success with smart-access point selection; (c) rip-up and reroute failure; (d) rip-up and reroute success.

Chu 2011] is a viable candidate for lower metal layers with increasing density. With self-aligned double patterning (SADP)–friendly 1D gridded design, the line-space array decomposition can be applied with tight control on overlay and wafer-print artifacts [Luk-Pat et al. 2013]. However, SADP-specific design rules and 1D layout patterns impose even more complicated constraints on the SC I/O pin access for the detailed router [Xu et al. 2014].

Detailed routing aims at pin access and searches for the exact route of each net. A typical detailed routing strategy performs pathfinding of the nets sequentially. For SC pin access, the access point selection of the local I/O pins of the net being routed could impact the routability of the remaining nets. A typical example is illustrated in Figure 1. The prerouted M2 wires in Figure 1(a) block the I/O pin on the right side of *Cell* 1, which makes the remaining net unroutable. A different access-point selection scheme is shown in Figure 1(b), in which the accessing points of net *A* are changed and all nets are successfully routed. The routing order of nets is also critical for routability when each I/O pin has a limited number of access points interfering each other [Ozdal 2009]. Thus, pin-access planning, including access-point selection at the SC level and net order prediction, is very important for the detailed router to achieve better pin accessibility. Meanwhile, the rip-up and reroute scheme is a classic method to improve the quality of detailed routing solutions [Dees Jr and Smith II 1981; Dees Jr and Karger 1982]. To fix the pin-access failure in Figure 1(a), two different rip-up and reroute procedures are shown in Figures 1(c) and 1(d). The routing wires involving the rip-up and reroute procedures are shown in dashed lines. In Figure 1(c), net *B* is ripped up and the rerouting results block the I/O pin in the middle of *Cell* 1 due to the M2 spacing rules. Successful rerouting can be achieved if net *A* is ripped up, as shown in Figure 1(d). Therefore, a pin access–driven rip-up and reroute scheme is strongly needed to determine the rerouting procedures.

A wide range of academic studies across various design stages have been dedicated to the pin-access issue in advanced technology nodes, including SC design [Hsu et al. 2014; Xu et al. 2014; Ye et al. 2015], placement mitigation [Taghavi et al. 2010], global

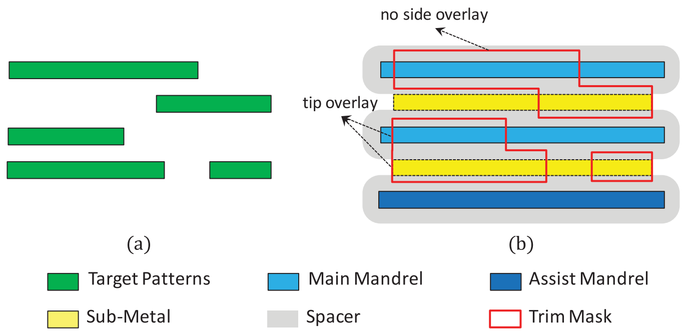

Fig. 2. The line-space array decomposition: (a) target patterns; (b) decomposition results.

routing [Alpert et al. 2013; Qi et al. 2014], and detailed routing [Nieberg 2011; Ozdal 2009]. Among them, SC I/O pin access and detailed routing play an important role due to their direct impact on detailed routability. Xu et al. [2014] addresses the I/O pin-access issue for each cell in isolation under SADP-specific constraints, but related detailed routing scheme is not explicitly presented. Ozdal [2009] proposes an escape routing strategy to improve detailed routability for the dense pin clusters instead of each SC within the design. Nieberg [2011] focuses on gridless pin access in the detailed routing stage, which cannot be directly applied in grid-based designs in advanced technology nodes [Smayling et al. 2013].

SADP, in particular, imposes a new set of difficulties on the detailed routing stage. For example, the M2 extensions along with the detailed routing are needed to achieve SADP-legal routing results [Du et al. 2013; Liu et al. 2014]. Several detailed routing algorithms have been presented to deal with the SADP-aware routing [Mirsaeedi et al. 2011; Gao and Pan 2012; Kodama et al. 2013; Du et al. 2013; Liu et al. 2014; Ding et al. 2015]. All previous works focus on 2D routing compatible with the SADP constraints. While 2D SADP-friendly patterns are susceptible to overlay [Liu et al. 2014], regular routing with 1D layout patterns provides tight control on the overlay of critical dimensions. Suppose that the regular layout patterns in Figure 2 are SADP-friendly; a strict alternation of metal track coloring can be applied [Luk-Pat et al. 2013; Xu et al. 2014]. Then, all side boundaries of target patterns are protected by a spacer, as illustrated in Figure 2(b). The pattern distortions of the line ends may still occur due to the potential overlay between *Trim mask* and *Spacer*. The overlay of the line ends is defined as *tip overlay* and the overlay of the side boundaries is defined as *side overlay* [Liu et al. 2014]. In contrast to the *side overlay*, the *tip overlay* is assumed to be noncritical, as it leads to small impact on the critical dimensions [Ma et al. 2012; Xiao et al. 2013]. We can observe that successful line-space array decomposition induces zero side overlay. In future nodes, the routing wires with tight pitches are preferred to be 1D [Smayling et al. 2013]. To apply the line-space array decomposition for the SADP process, the SADP-aware regular routing with 1D layout patterns becomes a competitive option for the detailed routing on the lower metal layers in advanced technology nodes. In addition, existing SADP-aware routing simply leaves the duty of pin access to the detailed router [Du et al. 2013; Liu et al. 2014], which is challenging for the ultimate pin accessibility.

In this article, we propose PARR, a comprehensive framework to explicitly address the SC I/O pin access and regular routing under SADP-specific layout constraints. Our framework starts with the pin accessibility study of a given SC library. Then, based on placement-level information, pin-access planning strategies are proposed to guide

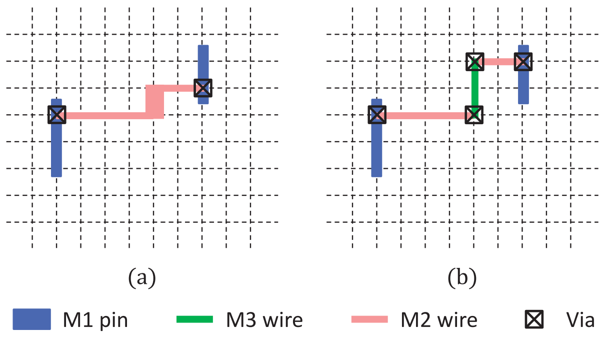

Fig. 3. 2D versus 1D routing patterns: (a) 2D routing patterns with a jog; (b) 1D routing patterns with extra vias.

the detailed router to improve ultimate pin accessibility. Our main contributions are summarized as follows:

—The local pin-access planning scheme is proposed to enable smart access point selection.
—We propose the global pin-access planning strategy based on the concept of a pin-access graph to guide the regular routing for the ultimate routability.
—The pin access–driven rip-up and reroute scheme is proposed to further improve detailed routability.

The experimental results show that regular routing with the overall pin-access planning scheme can achieve zero side overlay and highest routability compared with the state-of-the-art 2D SADP-aware router [Liu et al. 2014]. The pin access–driven rip-up and reroute scheme increases the ultimate routability with affordable cost. To the best of our knowledge, this is the first work to systematically address handshaking between SC-level pin-access planning and detailed routing stage with the SADP compliance.

The rest of this article is organized as follows. Section 2 briefly introduces the relevant background and problem formulation. Section 3 presents the cell-level pin-accessibility studies. Section 4 discuss the details of the pin-access planning strategies, pin access–driven rip-up and reroute, and overall routing flow. Section 5 demonstrates the effectiveness of the PARR framework. We present our conclusions in Section 6.

2. PRELIMINARIES AND PROBLEM FORMULATION

2.1. 1D Routing Patterns

In advanced technology nodes, regular routing wires with tight pitches are preferred to be 1D [Smayling et al. 2013]. 2D and 1D routing patterns for a simple net are shown in Figure 3. Compared with the 2D routing patterns with jogs in Figure 3(a), 1D routing patterns in Figure 3(b) have better printability but extra vias will be introduced when the detailed router changes the routing directions. In addition, different access points will be selected for the 1D routing patterns. Hence, smart pin-access planning strategies are crucial for the quality of routing solutions with 1D patterns.

2.2. Related Design Rules

In the detailed routing stage, a set of design rules needs to be followed by the detailed router so as to achieve legal regular layout patterns. Practically, these design rules lead to additional difficulties in accessing the local I/O pins of the SCs for the detailed router. Thus, related design rules are introduced, as follows.

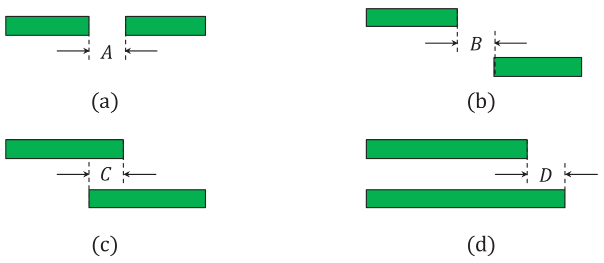

Fig. 4. Trim mask rules. (a) *OnTrackSpace*: $A \geq l_1$; (b) *OffTrackSpace*: $B \geq l_2$; (c) *OffTrackOverlap*: $C \geq l_3$; (d) *OffTrackOffset*: $D \geq l_4$ or $D = 0$.

Min-area Rule: Min-area is an important rule specifying the minimum area required for the polygons on the routing layer [Cadence 2009]. Due to the fixed width of layout patterns, as shown in Figure 2(a), we convert the min-area rule into the minimum length rule for the metal wires.

Trim Mask Rules: To achieve the line-space array decomposition for regular layout patterns, certain design rules, as shown in Figure 4, should be assumed for the SADP process to guarantee feasible line-space array decomposition. These rules originate from the manufacturability of the trim mask for the SADP process; detailed discussions can be found in Ma et al. [2012] and Xu et al. [2014]. As illustrated in Figure 2, target patterns are defined by the spacer and trim mask with the Boolean operation as "Target Patterns = Trim Mask *NOT* Spacer." Figure 4 summarizes the critical rules avoiding trim mask conflicts. Since only 1D routing patterns (horizontal patterns in Figure 4) are allowed, the side boundaries of patterns are defined by the sidewall, which means that no side overlay will be introduced, as shown in Figure 2(b). In our detailed router, we will legalize these rules for each routed metal wire along with the detailed routing procedure.

Via Rule: With 1D layout patterns on the single routing layer, only one direction, horizontal or vertical, is allowed for a routing path on the same layer. Then, changing direction of the routing path means switching the routing layer and via insertion among multilayer grids for the detailed router, as shown in Figure 3(b). Thus, the via rule is another important factor for the quality of the routing solution. The minimum center-to-center spacing rule of the vias is considered for the completeness of our pin-access planning framework, which can be extended to other complex via rules such as multiple patterning-related constraints. Since pin accesses are precomputed on the M2 layer, M1 I/O pins are brought up to the M2 layer and the via rule is imposed only on the M2 and upper layers.

2.3. Problem Definition

For an SC library, the set of feasible intracell pin accesses for each cell in isolation can be precomputed under a set of design rules on the M2 layer [Xu et al. 2014]. However, in practical designs, SCs are placed next to each other instead of being isolated, which means that the intercell pin access needs to be further addressed. Since the number of cells within a library is finite, the intracell and intercell pin access can be precomputed and stored in a look-up table (LUT). With the LUT for pin accessibility, we can propose efficient pin-access planning strategies guiding regular routing to enable handshaking between SC pin access and detailed routing. Meanwhile, design-rule legalizations are performed to guarantee legal routing results. Hence, we define the pin access–guided regular routing as follows.

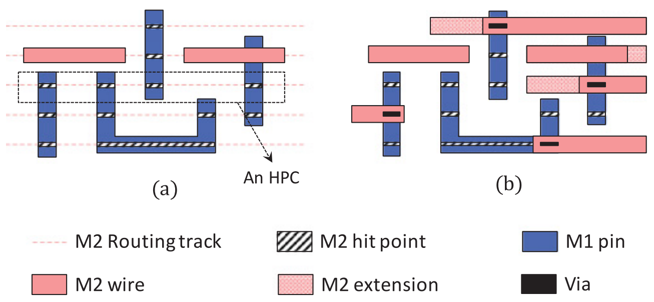

Fig. 5. The intracell pin access [Xu et al. 2014]: (a) M2 routing tracks and cell layout; (b) A VHPC with M2 wires and line-end extensions for intracell pin access.

PROBLEM 1 (PIN-ACCESS GUIDED REGULAR ROUTING). *Given a netlist, a grid routing plane, cell placement, pin-access LUT of the library and a set of design rules, the pin-access planning guided regular routing is to perform the regular routing and design rule legalization simultaneously to achieve SADP-friendly routing results.*

To evaluate the effectiveness of the proposed techniques for improving the quality of routing solutions, we have the following definition.

Definition 2.1 (*Routability*). The routability is defined as the percentage of nets routed over the total number of nets in a design.

3. PIN-ACCESSIBILITY PREDICTION

This section aims at the precomputation of pin accessibility under a set of design rules, which yields a LUT of the intercell and intracell pin accessibility for the given SC library.

3.1. Intracell Pin Access

The SC I/O pin-access problem has been explicitly addressed in Xu et al. [2014], in which pin accessibility is determined while minimizing the total amount of line-end extensions. However, line-end extensions can be automatically performed by the detailed router to fix related rule violations [Du et al. 2013; Liu et al. 2014]. In addition, the router is responsible for dynamically choosing the access direction of a specific accessing point of the SC I/O pin. Thus, a feasibility study at the SC level is enough to guide the detailed routing.

In this work, we adapt the pin access and standard cell layout co-optimization (PICO) [Xu et al. 2014] method to determine intracell pin accessibility. For convenience, we adopt the definitions of *Hit Point (HP)* and *Hit Point Combination (HPC)* from Xu et al. [2014]. Figure 5(a) illustrates a typical SC with four I/O pins in the M1 layer and two horizontal M2 wires for within-cell connections. The overlap on the M2 routing track and an I/O pin shape is an HP for that particular I/O pin. A set of hit points with each I/O pin accessed exactly once is defined as HPC for that cell, as shown in the dash box in Figure 5(a). An HPC is considered to be a *Valid Hit Point Combination (VHPC)* if the legal M2 wires can be achieved for pin access with the adapted pin access optimization (PAO) from Xu et al. [2014]. Otherwise, it is considered to be invalid. An example of VHPC is shown in Figure 5(b), in which each I/O pin is accessed by a horizontal M2 wire and a via from M2 to M1. The PAO technique [Xu et al. 2014] determines the validness of the selected HPC with the mixed integer linear programming (MILP) formulation. Since each cell has multiple I/O pins and each pin has several hit points, as shown in Figure 5(a), one cell may have many HPCs available. Moreover, the PICO

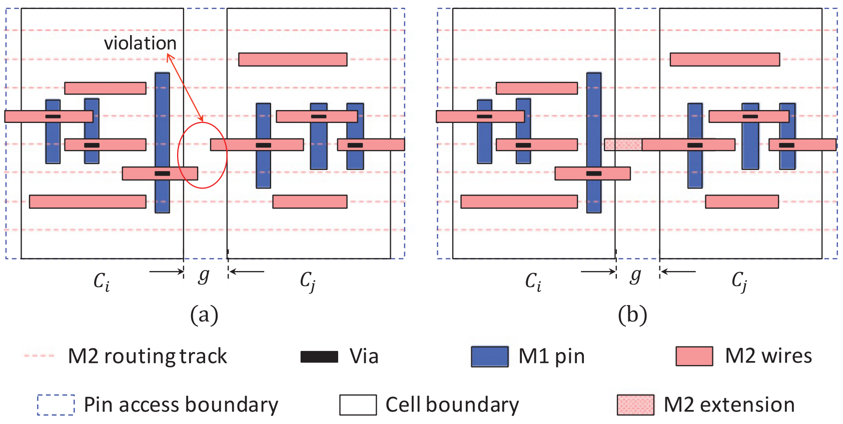

Fig. 6. The intercell pin access: (a) two cells placed next to each other and intracell pin access introduces a rule violation; (b) fixing the rule violation by line-end extension.

method [Xu et al. 2014] determines the validness of each VHPC for a cell making use of PAO. An HP is defined as a *Valid Hit Point (VHP)* if it is accessible within some VHPC. Otherwise, it is considered to be invalid. For example, the four HPs within the VHPC in Figure 5(b) are VHPs since the cell is accessible from those HPs. The intracell pin-access computation yields a 2D set of M2 wires for all VHPCs of each cell within the library, where pa_i^m denotes the m^{th} VHPC for the i^{th} cell.

3.2. Intercell Pin Access

We have discussed intracell pin access feasibility for each isolated cell. However, pin accessibility may interfere and degrade when two cells are placed next to each other. We quantify the intercell pin accessibility for a given SC pair with each cell assigned a specific VHPC. A typical example of a cell pair, denoted as $pair_{ij}$, is illustrated in Figure 6, in which c_i is placed to the left of c_j, with the gap distance as g. For pragmatic placement, g is an integer multiple of the placement pitch. The pin-access interference is expected from Figure 6(a) because the pin-access boundaries of the two cells are next to each other. To demonstrate the pin-accessibility interference, we choose m^{th} VHPC for c_i and n^{th} VHPC for c_j in Figure 6(b). The M2 wires for intracell pin access associated with the selected VHPCs are also shown in Figure 6(b), in which pin access M2 wires introduce extra rule violations even with VHPCs for c_i and c_j in isolation. To further explore intercell pin accessibility, additional line-end extensions are required to fix the violations and make $pair_{ij}$ accessible in Figure 6(b).

Actually, the additional line-end extensions based on the selected VHPCs for $pair_{ij}$ has the same formulation as the PAO in Xu et al. [2014]. Specifically, if two cells c_i and c_j are assigned m^{th} and n^{th} VHPC, and the gap distance is set to g in Figure 6(b), the feasibility of fixing extra violations can be evaluated with PAO on the set of M2 wires, that is, $pa_i^m \cup pa_j^n$. In particular, from Figure 6(a), we can see that the pin-access boundary is extended beyond the cell boundary. For practical implementation, the left and right boundaries of the cell could be extended by minimum M2 length, which preserves the validness of the HPs close to the cell boundary while satisfying the minLength rule for M2 wires. This pin-access boundary applies only to the pin accessibility prediction phase, which does not introduce additional placement or routing constraints. Since the detailed router cares only about the pin accessibility at the SC level, we only report VHPs and VHPCs to the detailed router, and each VHP is accessible from the M2 layer.

3.3. Look-Up Table Construction

Here, the LUT construction is presented for the library-wide intercell pin accessibility prediction. We assume that the M2 routing layer is horizontal and pin accessibility is precomputed on the M2 layer. Modern SC cell architecture creates horizontal power/ground rails on the M1 layer, which naturally introduces white space on the M2 layer between adjacent rows. Thus, pin-access interactions between adjacent rows are noncritical and the pin-access LUT needs to be calculated only for cells abutting horizontally. Connecting cells in adjacent rows involves M2 (horizontal) and M3 (vertical) routing layers simultaneously, which is performed by the detailed router. For each cell c_i in the library, we have computed a set of VHPCs from intracell pin-access feasibility study. We further evaluate the intercell pin-access feasibility of each cell pair ($pair_{ij}$) using the PAO technique, which will be stored in a LUT. Specifically, if two cells c_i and c_j are assigned m^{th} and n^{th} VHPC and the gap distance is set to g, the intercell pin-access feasibility can be evaluated on the set of M2 wires, that is, $pa_i^m \cup pa_j^n$. Then, $LUT(i, m, j, n, g)$ stores a Boolean value denoting whether intercell pin access is feasible or not when c_i is to the left of c_j. Here, the gap distance g is also an index of the LUT because changing the gap distance between cells has potential impact on the intercell pin accessibility. For example, in Figure 6(b), in spite of extra rule violations when c_i is to the left of c_j with gap distance as g, the violation can be fixed by additional line-end extensions. Thus, the element $LUT(i, m, j, n, g)$ will store a true value. The cell flipping is also considered and related values are stored during LUT construction.

Depending on the SC library design, the LUT size could be large or even unaffordable due to the large number of VHPCs for each SC. For the library that we used, the maximum number of VHPCs for one cell is 7745 and the average number of VHPCs for each cell is 589.1. To control the number of entries within the LUT, our LUT is constructed only on critical pin-access cells. Specifically, critical pin-access cells are those cells with a small number of VHPCs (e.g., <500) or some I/O pins of the cells have a limited number of VHPs (e.g., <5). For each cell pair in the LUT, we require both cells to be critical. We assume that if two cells are placed next to each other and only one is critical, the other cell has enough pin-access flexibility to compensate for the pin accessibility of its neighbor. Suppose that the number of pin-access critical cells is N, the maximum HPC per cell is M, and the maximum gap is g; then, the LUT size is at most $4 * N^2 * M^2 * g$. Since each LUT item is just true or false for the intercell pin accessibility, we store only the false entries for our practical implementation, which further reduces the size of the entire LUT. In addition, the computations of the entries with the LUT are independent from each other, which can be easily parallelized based on the OpenMP framework [OpenMP 2011] for speedup.

4. PIN ACCESS PLANNING–GUIDED REGULAR ROUTING

In this section, we propose the pin-access graph (PAG) to dynamically determine the pin accessibility of a single SC or a row of SCs. Then, local and global pin-access planning strategies are presented, which further guide the regular routing to improve pin accessibility. In addition, a pin access–driven rip-up and reroute scheme is proposed to increase the ultimate routability.

4.1. Single-Row Pin Access Graph

In the row-structure for placement in Figure 7(a), SCs are aligned horizontally and share the same height. The power and ground rails go from the very left to the very right of the die area. Given the position for each SC and a placement row, we build the single-row PAG. As illustrated in Figure 7(b), the single-row PAG is a directed graph starting from the virtual source node (s) on the left to the virtual target node (t) on the

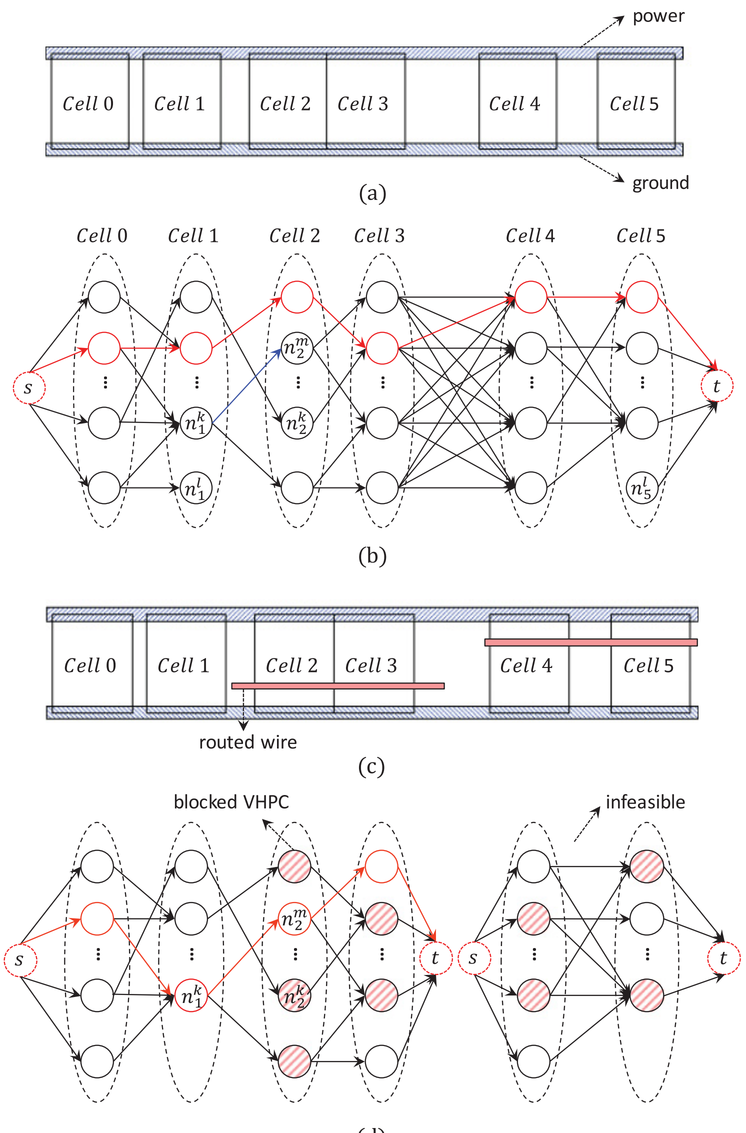

Fig. 7. Single-row pin-access graph: (a) cell placement; (b) initial pin-access graph; (c) cell placement with prerouted wires; (d) simplified pin-access graphs with blocked nodes.

right. For each SC placed within the row, we introduce a set of nodes into the PAG; each node corresponds to one of the VHPCs for that particular SC, where n_i^m denotes the node for the m^{th} VHPC for *Cell* i. We add edges between s and n_0^m for each m. Similarly, edges will be introduced between n_5^m and t for each m. No edges will be added between n_i^m and n_i^k, namely, nodes for the same SC. For neighboring SCs, such as *Cell* 1 and *Cell* 2, an edge, denoted with a blue arrow, will be added between n_1^k and n_2^m since the item $LUT(1, k, 2, m, g)$ is true, where g is the gap distance for $pair_{ij}$. In contrast, no edge is introduced between n_1^l and n_2^k since $LUT(1, l, 2, k, g)$ is false.

For the PAG, we have the following observation.

Observation 1. *The pin accessibility of the SCs on the M2 layer within the single row is equivalent to the existence of a path from s to t of the PAG associated with that particular row. We define a PAG component to be infeasible for pin access when no feasible path exists from s to t of that PAG component.*

If there exists a path from s to t, the SCs within the row are accessible using the set of VHPCs and corresponding pin-access M2 wires on the path. An example is shown in Figure 7(b). Moreover, routing wires on the M2 layer will be created on top of the SCs during the routing stage, as shown in Figure 7(c). The routing wires over the SC create routing blockages, which block some specific HPs of the SCs. This means that the associated VHPCs for the SC will be unavailable for pin access. As demonstrated in Figure 7(d), some nodes will become invalid, indicated by the dashed pink nodes due to the routing wires on top of the *Cell* 2−5.

Graph simplification. For the single-row PAGs constructed, some graph nodes have no children or parents, and can be safely removed without impacting the feasible paths from s to t within each PAG. As shown in Figure 7(b), the node n_1^l has no children and the node n_5^l has no parents, which are removed in Figure 7(d). In addition, we can perform the graph partitioning from Figures 7(b) to 7(d) since an edge exists between n_3^m and n_4^l for any (m, l) pair. The graph partitioning creates several components for each single-row PAG. Then, pin accessibility of the SCs with the row is equivalent to the existence of paths from s to t on two independent components in Figure 7. However, no feasible path exists on the right component of the PAG after the creation of prerouted wires in Figure 7, which means that prerouted wires need to be ripped up to preserve the routability of the remaining nets. To achieve a quick update on the PAGs, graph partitioning is applied to all PAGs associated with the placement rows of the design. As discussed in Section 5, the graph partitioning technique controls the size of the single component of the PAGs, which leads to feasible pin access planning–guided detailed routing within affordable runtime cost.

Furthermore, each component of the PAG is related to a set of SCs in proximity, which is bounded by a determined bounding box. Since the search for impacted components of the PAG needs to be done whenever a net is routed, we adopt R-tree [Guttman 1984] to enable the fast indexing bounding box of each component of the PAG.

4.2. Local Pin-Access Planning

Intracell pin accessibility study yields a set of VHPCs, denoted as $VHPC_i$, for each cell c_i within the library. An HP is invalid if there are no VHPCs associated with that HP. Therefore, invalid HPs are inaccessible and should be removed before the detailed routing stage, which helps to avoid unnecessary routing efforts. Next, we discuss how to differentiate among various VHPs for a particular I/O pin.

The precomputation of the intracell pin access achieves a set $VHPC_i$ for c_i. For each I/O pin for the c_i, we propose a **Dynamic Hit Point Scoring** strategy to differentiate among various VHPs for that particular I/O pin. The basic idea is that a higher score is assigned to an HP if that particular HP has a larger number of VHPCs associated with it than other HPs of the same I/O pin. Hence, we calculate the score for the h^{th} HP of the k^{th} I/O pin for c_i, namely hp_i^{kh}, as

$$score(hp_i^{kh}) = \frac{\text{number of VHPC's associated with } hp_i^{kh}}{\text{total number of VHPC's for } c_i}. \tag{1}$$

In the sequential routing scheme, the M2 wires created for routed nets become blockages, an example of which is shown in Figure 7(c). As discussed earlier, some VHPCs for blocked cells become invalid, as illustrated in Figure 7. Therefore, the score for each HP should be updated dynamically during the detailed routing stage. We update only the scores of the HPs of the cells impacted by the routed nets. For practical implementation, we adopt the "unordered_map" data structure to store the VHPCs for each cell for fast updating procedure. For the single-net routing, the router prefers selecting the HPs with higher scores for source and target pins of the net being routed.

4.3. Global Pin-Access Planning

The local pin-access planning enables smart HP selection for single-net routing. During sequential detailed routing, the routed wires block some VHPs of the I/O pins not yet routed, which degrades the routability of the remaining nets. Thus, this section addresses the global, rather than local, prediction of the pin accessibility.

For sequential detailed routing, the relative order of routing nets has a potential impact on routability, as discussed in Section 1. Here, we introduce two techniques to enable **Net Deferring** and improve pin accessibility. First, the routability of a net relates to the accessibility of the source or target pin. This motivates us to defer the routing of nets with a robust source and target pins, which both have many VHPs available. Second, the PAG for each placement row helps determine the accessibility of the cells within that row. To preserve the global pin accessibility of the remaining nets, we dynamically maintain the source-to-target path existence of each component of the PAGs. Specifically, we update the PAG based on the routed wires from the trial routes of each net. If no feasible path exists from source to sink in some PAG, we will rip up and increase the deferring cost of that net and reroute that net later. Otherwise, we keep the routing results of the net. Therefore, the weight for the order of the net n_k is calculated as follows:

$$order(n_k) = HPWL(n_k) \cdot (1 + \alpha \cdot \min\{hp_s, hp_t\}) + DCost(n_k). \tag{2}$$

In Equation (2), $HPWL(n_k)$ denotes the half-perimeter wirelength of net n_k, α is a user-defined parameter, and hp_s and hp_t denote the number of VHPs available for source and target pins, respectively. $DCost(n_k)$ is the deferring cost of n_k. With the net-deferring scheme, one net may be deferred several times due to its impact on the routability of the remaining nets. To quantify that, we have the following definition.

Definition 4.1 (*Deferring Cycle*). A deferring cycle is the maximum number of times that a net is deferred before reaching the cost upper bound.

In one deferring cycle, if routed M2 wires of some net make some PAG component infeasible for pin access, the routing of that net will be deferred and current routing patterns will be cleared to preserve the routing of remaining nets. An example has been illustrated in Figure 8. Only 4 nets, including $\{a_1,a_2\}$, $\{b_1,b_2\}$, $\{c_1,c_2\}$, and $\{d_1,d_2\}$, are explicitly drawn with routing patterns; other nets, including net_set1 and net_set2, in the design are shown here for clear demonstration. The original order of routing is $\{a_1,a_2\}$, $\{b_1,b_2\}$, $\{c_1,c_2\}$, net_set1, $\{d_1,d_2\}$, net_set2 based on Equation (2). As shown in Figure 8(a), the net-deferring strategy can tell that the pin d_1 is blocked right after the routing of $\{c_1,c_2\}$ is complete. With PAGs constructed, blocking pin d_1 corresponds to the fact that the PAG component associated with the pin is infeasible for pin access, as discussed in Figure 7(d). Instead of waiting until the nets in net_set1 are routed to determine that $\{d_1,d_2\}$ is unroutable, the net-deferring strategy will increase the *DCost* of $\{c_1,c_2\}$, and net $\{d_1,d_2\}$ will be routed before $\{c_1,c_2\}$. As shown in Figure 8(b), the routing has been successfully achieved with the net-deferring scheme. In particular, the net-deferring scheme is dynamic, which means that the net will be deferred whenever necessary.

The detailed net-deferring scheme is shown in Algorithm 1. From Line 1 to Line 6, the minimum heap for routing nets is built based on the order of each net computed using Equation (2). In each loop, when the *net_heap* is not empty, the *net* with minimum order

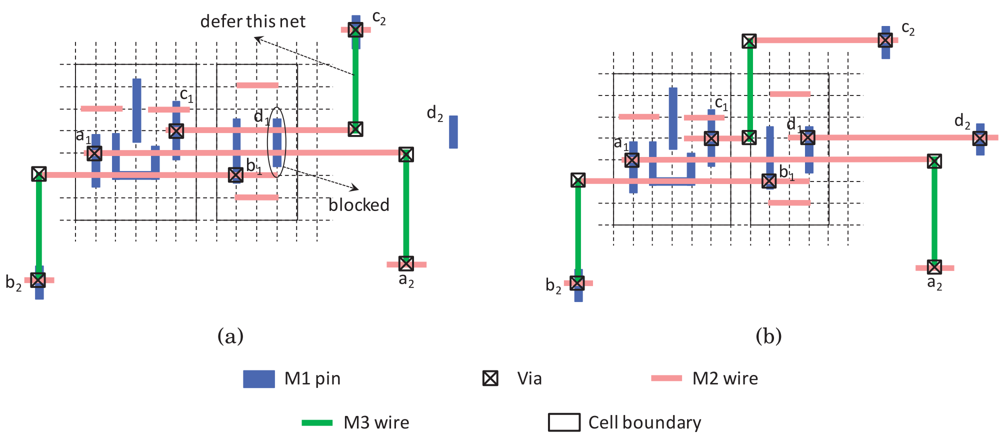

Fig. 8. A net-deferring example: (a) defer the routing of net $\{c_1,c_2\}$; (b) successful routing.

ALGORITHM 1: Net-Deferring Algorithm

```
Require: a set of nets ($Nets$), maximum deferring cost ($maxCost$), increasing unit for deferring
    cost ($unit$) and pin-access graphs for placement rows ($PAGs$);
Output: a set of nets ($nets\_deferred$)
 1: Define $net\_heap$ as the minimum heap for $Nets$;
 2: for each net $n_k$ in $Nets$ do;
 3:     Set $DCost(n_k) = 0$;
 4:     Compute $order(n_k)$ based on Equation (2);
 5:     $insert\_heap(n_k, net\_heap)$;
 6: end for
 7: while $net\_heap$ is not empty do;
 8:     Define $net = extract\_min(net\_heap)$;
 9:     Perform A* search for $net$ and achieve trial routes;
10:     Update impacted components of $PAGs$;
11:     if $PAGs$ are infeasible for pin access then;
12:         $DCost(net) = DCost(net) + unit$;
13:         if $DCost(net) < maxCost$ then;
14:             Defer $net$ and update $PAGs$;
15:             $insert\_heap(net, net\_heap)$;
16:         else
17:             Add $net$ to $nets\_deferred$;
18:         end if
19:     end if
20: end while
21: return $nets\_deferred$;
```

is extracted from the *net_heap* and single-net routing is performed in Line 8 and Line 9. From the routing wires of the *net*, the impacted components of PAGs are updated on Line 10. Impacted components are those components containing nodes blocked by the newly created M2 wires, as shown in Figures 7(c) and 7(d). As discussed in Section 4.1, R-tree enables quick search of the impacted components when new routing wires are created. If the routing results of the *net* break the pin accessibility of the *PAG* from Line 11 to Line 15, the deferring cost is increased and the *net* is deferred and pushed back to the *net_heap* when the accumulated deferring cost is within the maximum bound. Otherwise, the *net* is added to the *nets_deferred* in Line 17. In the end, the *nets_deferred* will be returned as the input of the pin access–driven rip-up and reroute.

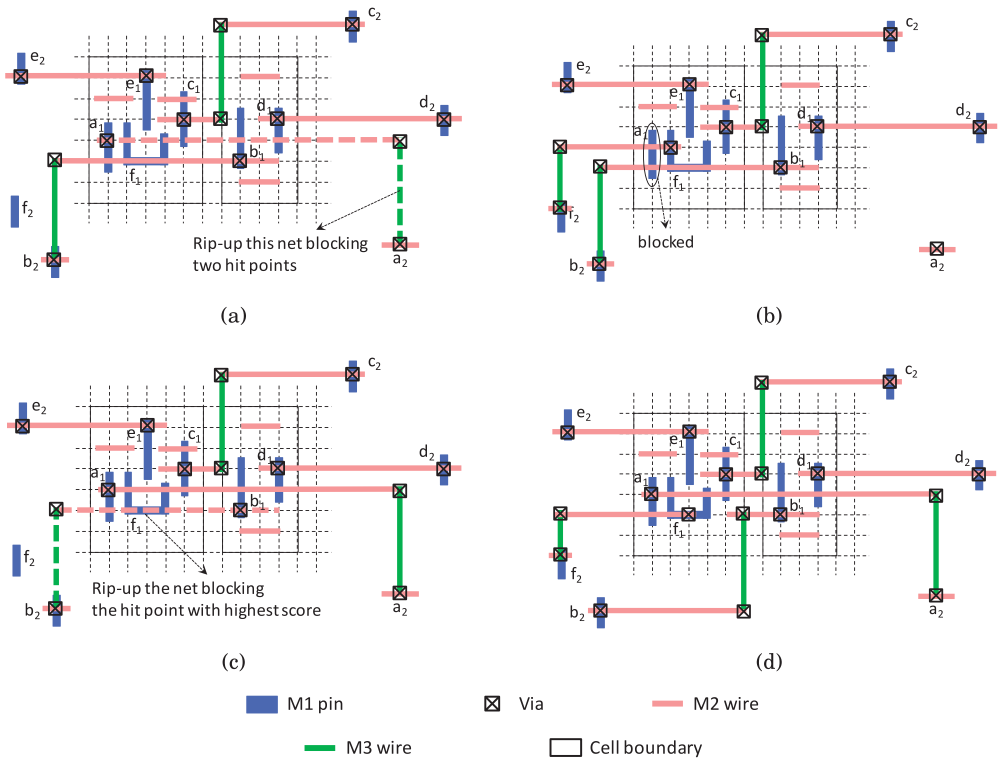
e₂
e₁
c₁
c₂
a₁
d₁
d₂
b₁
f₁
f₂
b₂
a₂
Rip-up this net blocking two hit points
(a)
blocked
(b)
Rip-up the net blocking the hit point with highest score
(c)
(d)
M1 pin
Via
M2 wire
M3 wire
Cell boundary

Fig. 9. A rip-up and reroute example: (a) rip up net $\{a_1,a_2\}$; (b) reroute net $\{f_1,f_2\}$ and pin a_1 is blocked; (c) rip-up net $\{b_1,b_2\}$; (d) successful rip-up and reroute.

4.4. Pin Access–Driven Rip-Up and Reroute

The local and global pin-access planning schemes guide the detailed router for better pin accessibility. There might be some remaining nets, that is, *nets_deferred* from Algorithm 1, that are not yet routed. Thus, an additional pin access–driven rip-up and reroute scheme is strongly needed to improve the ultimate routability.

As discussed in Figures 1(c) and 1(d), different rip-up and reroute procedures may lead to pin-access success or failure, depending on the nets to be ripped up and the outcome of the A* searches. To improve solution quality, a typical rip-up and reroute procedure involves various trials or several iterations [Dees Jr and Smith II 1981; Dees Jr and Karger 1982]. Our pin access–driven rip-up and reroute scheme follows this direction with local pin-access information. Figure 9 gives two rip-up and reroute examples, in which only 6 nets, that is, $\{a_1,a_2\}$, $\{b_1,b_2\}$, $\{c_1,c_2\}$, $\{d_1,d_2\}$, $\{e_1,e_2\}$, and $\{f_1,f_2\}$, are drawn for clear demonstrations. As shown in Figure 9(a), the U-shape pin f_1 is blocked by the routing patterns from two nets: $\{a_1,a_2\}$ and $\{b_1,b_2\}$. If the net $\{a_1,a_2\}$ is ripped up in Figure 9(a), the rerouting result in Figure 9(b) shows that pin a_1 is blocked, which leads to the rerouting failure. In Figure 9(c), if we rip up the net $\{b_1,b_2\}$ blocking the hit point with the highest score from Equation (1), successful rip-up and reroute can be achieved, as shown in Figure 9(d). Thus, the pin access–driven rip-up and reroute scheme makes use of the dynamic hit point scoring in Equation (1) to determine the rip-up and reroute procedures. Note that the pin access–driven rip-up and reroute scheme is different from the net-deferring scheme because only local pin-access information is used to guide the rip-up and reroute iterations. Since global pin-access planning

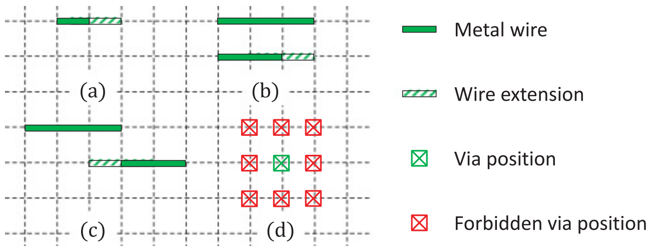

Fig. 10. Legalizations: (a) minLength, (b) parallel line ends, (c) anti-parallel line ends, and (d) via rule.

is not an optimal strategy, the pin access–driven rip-up and reroute scheme aims at improving the routability with an extra iterative step. The details of our rip-up and reroute algorithm are skipped here because it can be achieved by incorporating the greedy metric from Equation (1) into the traditional rip-up and reroute procedures [Dees Jr and Smith II 1981; Dees Jr and Karger 1982].

4.5. Overall Routing Flow

Our routing strategy adopts the grid-based routing model and targets the 1D layout patterns friendly to the design rules introduced in Section 2.2. The schemes related to design-rule legalization are demonstrated in Figure 10. The line-end extensions are performed to fix the violations relevant to minLength rule and trim mask rules for SADP in Figures 10(a) to 10(c). Since both intracell and intercell pin accessibility are precomputed on the M2 layer, we report only those M2 HPs of I/O pins accessible from the M2 layer to the router in the preprocessing phase. Thus, for the routing of each net, all source/target M1 pins can be accessed through HPs on the M2 layer. Considering the spacing rule for the via layer, we impose the forbidden grids once a legal via is inserted for the routed net. In Figure 10(d), neighboring grids surrounding the via position are forbidden to be used for the remaining nets. The via rule is imposed on the M2 and upper layers.

We focus on the routing-grid model, which can easily be extended to the multilayer routing framework. Then, our detailed router follows the paradigm of the A* search, which is guided by the local and global pin access planning strategies. The cost of the routing grid is calculated while performing the A* search. If we consider a routing path from grid g_i to grid g_j, the cost of grid g_j, denoted as $C(g_j)$, can be computed as follows:

$$C(g_j) = C(g_i) + \theta \cdot (1 - score(hp_j)) + \eta \cdot C(forbid(j)) + \beta \cdot C(WL_{ij}) + \gamma \cdot C(Via_{ij}). \quad (3)$$

In Equation (3), $score(hp_j)$ is the dynamic hit point score for g_j if g_j is a source or target grid. Otherwise, $score(hp_j)$ is set to 1. In general, the A* search prefers routing grids with lower cost. Thus, the term $score(hp_j)$ enables the local pin-access planning, which prefers selecting the HPs with higher scores for the source or target pins of the net being routed. $C(forbid(j))$ is the forbidden cost for the grids if the grid g_j is within the prohibited region of some prerouted wires [Luk-Pat et al. 2013; Du et al. 2013]. This cost is set to help the design-rule legalization for the trim mask rules. $C(WL_{ij})$ and $C(Via_{ij})$ are the amount of wirelength and vias for the routing path from g_i to g_j. $\beta, \gamma, \theta, \eta$ are user-defined parameters to adjust weights for cost.

The overall routing flow is demonstrated in Figure 11. The intracell and intercell pin accessibility for an SC library is precomputed and stored in the LUT. Then, based on the placement-level information and the LUT, the PAGs are constructed for the design being routed. With the PAGs, the net-deferring algorithm makes use of the local and global pin-access planning strategies to achieve better pin accessibility. For those

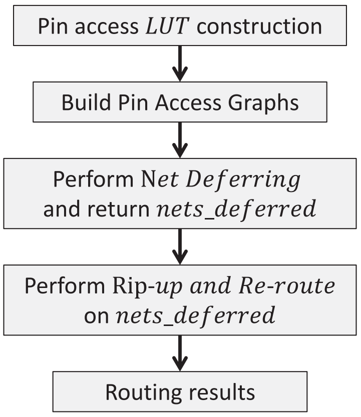
Pin access LUT construction
Build Pin Access Graphs
Perform N$et\ Deferring$ and return $nets\_deferred$
Perform Rip-$up\ and\ Re$-$route$ on $nets\_deferred$
Routing results

Fig. 11. Overall routing flow.

Table I. Benchmarks Statistics

Ckt	ecc	efc	ctl	alu	div	top
Net#	1671	2219	2706	3108	5813	22201
Cell#	1302	1197	1725	1802	3260	12576
Size(um^2)	21 x 21	20 x 19	24 x 24	20 x 19	31 x 31	57 x 56

unrouted nets, the pin access–driven rip-up and reroute scheme improves the ultimate routability with several reroute iterations.

5. EXPERIMENTAL RESULTS

We have implemented PARR in C++ and all experiments are performed on a Linux machine with a 3.4GHz Intel(R) Core and 32GB memory. With the help from Liu et al. [2014], the 2D SADP-aware routing results are generated on a Linux machine with a 2.0GHz CPU and 72GB memory. We modify and scale the NanGate 45*nm* open-cell library [NanGate 2012] to represent the pin-access scenario in advanced technology nodes, in which M2 wires may be introduced in the SC layout design [Xu et al. 2014]. For the LUT construction, we store only the false entries in our implementation since each entry is just true or false for the intercell pin-accessibility checking. The number of entries in the LUT constructed in our implementation is around $1.8 * 10^6$. In addition, we parallelized the computations of the entries in the LUT with the OpenMP framework [OpenMP 2011], which reduces the construction runtime from a few hours in Xu et al. [2014] to around 30min. As illustrated in Table I, modules from OpenSparc T1 are synthesized with Design Compiler [Synopsys 2012]. The placement results are generated using Cadence SOC encounter [Cadence 2012] with utilization rate set to 0.7. All benchmarks are scaled and compacted to 10*nm*-representative dimensions. Since our work targets improving pin accessibility in the detailed routing stage, global nets are treated as prerouted nets and are beyond the scope of our detailed routing framework. The bounding box of a global net crosses more than M horizontal or vertical routing tracks and $M = 40$ in our implementation. We focus on the two-layer (M2 and M3) regular routing; the routing directions of M2 and M3 are horizontal and vertical, respectively. We adopt Gurobi [2014] as our MILP solver for pin-accessibility prediction. The upper bound on the gap distance of a cell pair is set to $g = 2$. The width and space of the M2 and M3 wires are assumed to be 24*nm*. The minimum length of the metal wires is set to 48*nm*. The minimum center-to-center spacing of the vias is set to 96*nm*.

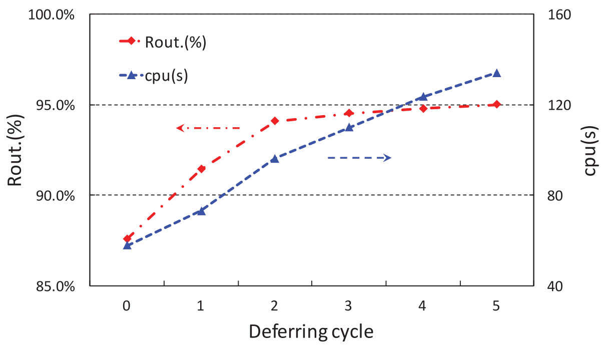

Fig. 12. Routability and CPU versus deferring cycle.

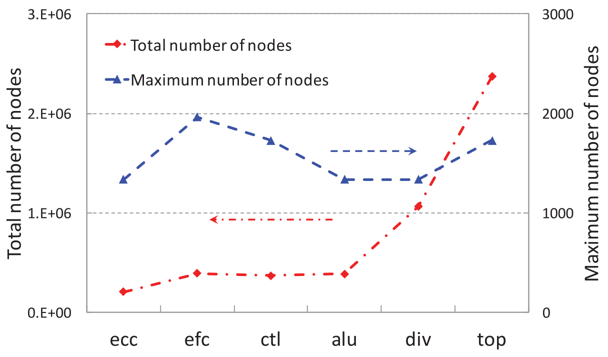

Fig. 13. Total number of nodes for components in PAGs and maximum number of nodes for each component versus different benchmarks.

SADP-related parameters are the same as those in Luk-Pat et al. [2012] and Xu et al. [2014]. The spacer deposit width is set to 24*nm*. The width and height of the routing grid is set to 48*nm*. For trim mask design, the minimum resist width and space are set to 44*nm* and 46*nm*, respectively [Luk-Pat et al. 2012]. The user-defined parameters in Equations (2) and (3) are set to $\alpha = 0.15, \theta = 4, \eta = 10, \beta = 1, \gamma = 5$. The iteration upper bound is set as $l_0 = 3$ for the pin access–driven rip-up and reroute scheme.

As discussed in Section 4.3, one single net may be deferred several times. The deferring cost bound is set further to avoid too much rerouting efforts. We define *deferring cycle* as the maximum number of times that a net is deferred before reaching the deferring cost upper bound. The “routability” is defined as the number of nets routed over the total number of nets in the design. The trade-off between routability, runtime, and deferring cycle is illustrated in Figure 12 for the benchmark “alu.” As the deferring cycle increases, both the routability and runtime increase monotonically. The net-deferring scheme improves the routability significantly during first few deferring cycles. After that, the runtime increases quasilinearly while the amount of routability improvement degrades. Thus, the *deferring cycle* is set to 3 for the global pin-access planning scheme, which is enough for routability improvement exploration. For the results in Table II, the increasing *unit* for deferring cost is set to 300 for better routability.

Apart from the A* search for the trial routes, the extra computational cost for pin-access planning schemes is mainly associated with the construction and queries of the PAGs. Figure 13 shows the node size of the PAGs constructed for various benchmarks. Since graph partitioning is applied to each single-row PAG, the PAGs of each design

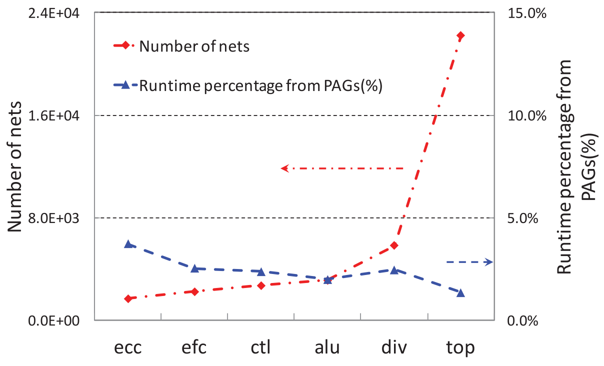

Fig. 14. Number of nets and runtime percentage from PAG construction and queries versus different benchmarks.

consist of various components. As shown in Figure 13, the total number of nodes of the PAGs increases as the size of the design increases. On the other hand, the maximum number of nodes for each component is bounded by 2000. This means that the runtime of determining feasible paths in each component of the PAGs is bounded by a small constant in Algorithm 1. Figure 14 illustrates the analysis of the percentage of runtime associated with the PAG construction and queries. The number of nets increases monotonically as the design complexity increases. However, the runtime percentage from PAG construction and queries is bounded by a small constant, that is, 5%. The trend of runtime percentage decrease from PAG construction and queries means that the routing cost dominates the overall computational cost as the design complexity increases.

Our work targets at the two-layer 1D detailed routing, which means that no jogs are allowed on the single-layer routing. The main focus of this work is on pin-access planning during the detailed routing stage, while considering the difficulties from SADP-specific constraints. We believe that comparing with a third-party SADP-aware router helps to empirically validate the effectiveness of our proposed methodologies. We choose the router in Liu et al. [2014] as our baseline router due to its reported performance and efficiency and we obtain related routing results on our set of benchmarks with the help from Liu et al. [2014]. In Table II, we compare different SADP-aware detailed routing schemes, including the baseline SADP-aware routing from Liu et al. [2014], regular routing with local pin-access planning (LPAP), regular routing with local pin-access planning and rip-up and reroute (LPAP + RR), regular routing with local and global pin-access planning (LPAP + GPAP) and regular routing with local and global pin-access planning and rip-up and reroute (LPAP + GPAP +RR), that is, overall routing flow. LPAP was originally proposed in Xu et al. [2014] and routing solutions from the schemes of LPAP and LPAP + GPAP are given in Xu et al. [2015]. The schemes of LPAP + RR and LPAP + GPAP + RR are proposed to demonstrate the effectiveness of pin access–driven rip-up and reroute, which also leads to a complete routing solution. Overall, it is difficult to achieve 100% routability within reasonable runtime owing to problem complexity. To quantify the difference in via count among various routing schemes, we average the number of vias from M3 to M2 over the number of routed nets, namely, via number per routed net, which is listed as V.p.n. The total wirelength, listed as WL*, is the summation of the half perimeter wirelength for unrouted nets and actual wirelength for routed nets in terms of routing grid count. OLL denotes the total side overlay length [Liu et al. 2014]. Rout. denotes the percentage of routed nets over total number of nets in the design and CPU denotes the runtime in seconds.

Table II. Comparison of Detailed Routability for Regular Routing with Different Routing Methodologies

	Liu et al. [2014]					LPAP					LPAP + RR					LPAP + GPAP					LPAP + GPAP + RR				
Ckt	V.p.n	WL*	OLL	Rout.	CPU(s)	V.p.n	WL*	OLL	Rout.	CPU(s)	V.p.n	WL*	OLL	Rout.	CPU(s)	V.p.n	WL*	OLL	Rout.	CPU(s)	V.p.n	WL*	OLL	Rout.	CPU(s)
ecc	2.31	41497	2775	91.1%	16.77	2.51	45102	0	91.2%	14.21	2.59	46248	0	94.4%	23.09	2.66	46588	0	96.4%	19.98	2.67	46519	0	97.4%	26.06
efc	2.31	54459	4703	82.5%	100.5	2.25	56457	0	88.1%	22.86	2.45	58848	0	92.1%	35.20	2.40	57834	0	94.9%	34.52	2.45	58717	0	96.2%	45.40
ctl	2.24	67470	5255	87.3%	93.80	2.26	71643	0	88.3%	22.39	2.45	72984	0	92.7%	36.05	2.42	72388	0	95.3%	37.14	2.50	73112	0	96.8%	47.31
alu	2.26	68491	5713	79.4%	143.4	2.22	73430	0	87.5%	28.32	2.44	76424	0	92.0%	53.09	2.44	75679	0	95.2%	45.92	2.47	75814	0	95.8%	61.92
div	2.29	139309	11267	79.4%	253.5	2.34	150356	0	87.9%	57.79	2.55	156615	0	91.8%	100.5	2.51	155704	0	94.6%	106.0	2.58	157625	0	96.2%	132.2
top	N/A	N/A	N/A	N/A	N/A	2.32	496228	0	88.2%	253.7	2.47	514499	0	92.7%	378.5	2.47	513366	0	95.3%	763.2	2.52	516902	0	96.0%	791.0
Avg.	0.901	0.902		0.870	1.943	0.915	0.960		0.918	0.361	0.984	0.997		0.960	0.567	0.982	0.992		0.988	0.912	1.000	1.000		1.000	1.000

Table II demonstrates the strength of the pin-access planning schemes for regular routing over 2D detailed routing from Liu et al. [2014]. For the LPAP, the net order is computed with Equation (2), with the term $DCost(n_k)$ ignored. Our router legalizes the routing wires for each routed net sequentially under the given set of design rules. The routing wires are all protected by the spacer after line-space array decomposition [Luk-Pat et al. 2013], which leads to the zero side overlay. Moreover, the LPAP achieves better routability and shorter runtime by avoiding the extra efforts to maintain the overlay constraint graph from Liu et al. [2014]. With LPAP, our router gives better routability with competitive even shorter runtime compared with Liu et al. [2014]. The cost is a 1.4% increase in the V.p.n and a 5.8% increase in the WL*. Since the regular routing does not allow jogs, extra vias are needed when routing wires change directions compared with the 2D routing scheme. We believe that the 5.8% increase in the wirelength lower bound is reasonable considering the better routability and shorter runtime from the LPAP scheme. Furthermore, there is some variance in the routability improvement with LPAP, and further improvement on the pin accessibility of all benchmarks is quite necessary for the ultimate routability. For example, for ecc and ctl, the LPAP scheme gives similar routability as Liu et al. [2014], which means that the LPAP scheme could be improved further. Therefore, we compare two different approaches on top of LPAP, that is, LPAP + RR and LPAP + GPAP. Compared to the LPAP approach, troutability has been improved by 4.8% and 7.0% with the LPAP + RR and LPAP + GPAP approach, respectively. The routability improvement is still at the cost of slight degradations on the V.p.n and WL*. Moreover, the LPAP + GPAP approach is better than LPAP + RR with 2.8% routability improvement and better results on V.p.n and WL*. In terms of runtime, all approaches are with similar scalability for small benchmarks, but LPAP + GPAP consumes larger runtime in the "top" benchmark due to heavy computational cost of the trial routes from the net-deferring procedure, while LPAP + RR stops at the stage of no routability improvement or iteration upper bound. The routing results with LPAP + GPAP + RR are further demonstrated in Table II. The LPAP + GPAP + RR scheme achieves the best routability (96.4%, on average) for all benchmarks, which is an over 10% increase from Liu et al. [2014], 9.2% increase from the LPAP scheme, 4.0% increase from the LPAP + RR scheme, and 1.2% increase from the LPAP + GPAP scheme. To achieve this high routability, there are some degradations on V.p.n and WL* and runtime increase compared with other schemes. Despite these overheads, we believe that LPAP + GPAP + RR is a reasonable trade-off for the ultimate routability improvement. This work aims at pin accessibility, that is, routability, and design rule clean routing results with some cost from other metrics. For different routing strategies, note that the WL* and V.p.n for Liu et al. [2014], LPAP, LPAP + RR or LPAP + GPAP might increase if similar routability was achieved since the remaining nets are difficult to be routed. The runtime from LPAP + GPAP and LPAP + GPAP + RR is higher than other strategies because the net-deferring procedure may perform trial routes of a net several times. To the best of our knowledge, this is the first work explicitly addressing the pin-access planning strategies during the detailed routing stage, and we can achieve the best routability with the proposed techniques. Future work may include reducing runtime overhead and achieving more scalable pin-access planning solutions.

6. CONCLUSION

In this article, we propose a comprehensive framework, including pin-access LUT construction for a given library, and local and global pin-access planning to improve the pin accessibility during SADP-aware regular routing. The pin access–driven rip-up and reroute scheme is further proposed to improve the ultimate routability. To the best of our knowledge, this is the first work to systematically enable handshaking between standard cell-level pin access and detailed routing stage. Compared to the 2D SADP-aware detailed router, our approach can achieve significant improvement in terms of the overlay and routability. Our future work includes incorporating the netlist information into the pin-access planning stage and achieving more scalable solutions by reducing runtime overhead.

ACKNOWLEDGMENTS

The authors would like to thank Dr. Wen-Hao Liu from Cadence for his helpful discussions and feedback on this work.

REFERENCES

C. J. Alpert, Z. Li, C. N. Sze, and Y. Wei. 2013. Consideration of local routing and pin access during VLSI global routing. Retrieved April 2, 2016 from http://www.google.com/patents/US20130086544 US Patent App. 13/252,067.

Cadence. 2009. LEF/DEF Language Reference. Retrieved April 2, 2016 from ftp://ftp.sitsemi.ru/pub/Cadence/lefdefref.pdf.

Cadence. 2012. Cadence SOC Encounter. Retrieved April 2, 2016 from http://www.cadence.com/.

William A. Dees Jr and Patrick G. Karger. 1982. Automated rip-up and reroute techniques. In *ACM/IEEE Design Automation Conference (DAC'82)*. 432–439.

William A. Dees Jr and Robert J. Smith II. 1981. Performance of interconnection rip-up and reroute strategies. In *ACM/IEEE Design Automation Conference (DAC'81)*. 382–390.

Yixiao Ding, Chris Chu, and Wai-Kei Mak. 2015. Detailed routing for spacer-is-metal type self-aligned double/quadruple patterning lithography. In *ACM/IEEE Design Automation Conference (DAC'15)*. 69:1–69:6.

Yuelin Du, Qiang Ma, Hua Song, James Shiely, Gerard Luk-Pat, Alexander Miloslavsky, and Martin D. F. Wong. 2013. Spacer-is-dielectric-compliant detailed routing for self-aligned double patterning lithography. In *ACM/IEEE Design Automation Conference (DAC'13)*. 93:1–93:6.

Jhih-Rong Gao and David Z. Pan. 2012. Flexible self-aligned double patterning aware detailed routing with prescribed layout planning. In *ACM International Symposium on Physical Design (ISPD'12)*. 25–32.

Gurobi. 2014. GUROBI. Retrieved April 2, 2016 from http://www.gurobi.com/html/academic.html.

Antonin Guttman. 1984. *R-Trees: A Dynamic Index Structure for Spatial Searching*. Vol. 14. ACM, New York, NY.

Meng-Kai Hsu, Nitesh Katta, Homer Yen-Hung Lin, Keny Tzu-Hen Lin, King Ho Tam, and Ken Chung-Hsing Wang. 2014. Design and manufacturing process co-optimization in nano-technology. In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD'14)*. 574–581.

Chikaaki Kodama, Hirotaka Ichikawa, Koichi Nakayama, Toshiya Kotani, Shigeki Nojima, Shoji Mimotogi, Shinji Miyamoto, and Atsushi Takahashi. 2013. Self-aligned double and quadruple patterning-aware grid routing with hotspots control. In *IEEE/ACM Asia and South Pacific Design Automation Conference (ASPDAC'13)*. 267–272.

Iou-Jen Liu, Shao-Yun Fang, and Yao-Wen Chang. 2014. Overlay-aware detailed routing for self-aligned double patterning lithography using the cut process. In *ACM/IEEE Design Automation Conference (DAC'14)*. 50:1–50:6.

Gerard Luk-Pat, Alex Miloslavsky, Ben Painter, Li Lin, Peter De Bisschop, and Kevin Lucas. 2012. Design compliance for spacer is dielectric (SID) patterning. In *Proceedings of SPIE*. 83260D–83260D.

Gerard Luk-Pat, Ben Painter, Alex Miloslavsky, Peter De Bisschop, Adam Beacham, and Kevin Lucas. 2013. Avoiding wafer-print artifacts in spacer is dielectric (SID) patterning. In *Proceedings of SPIE*. 868312–868312.

Yuangsheng Ma, Jason Sweis, Hidekazu Yoshida, Yan Wang, Jongwook Kye, and Harry J. Levinson. 2012. Self-aligned double patterning (SADP) compliant design flow. In *Proceedings of SPIE*. 832706–832706.

Minoo Mirsaeedi, J. Andres Torres, and Mohab Anis. 2011. Self-aligned double-patterning (SADP) friendly detailed routing. In *Proceedings of SPIE*. 79740O–79740O.

NanGate. 2012. NanGate FreePDK45 Generic Open Cell Library. Retrieved April 2, 2016 from http://www.si2.org/openeda.si2.org/projects/nangatelib.

Tim Nieberg. 2011. Gridless pin access in detailed routing. In *ACM/IEEE Design Automation Conference (DAC'11)*. 170–175.

OpenMP. 2011. The OpenMP API Specification for Parallel Programming. Retrieved April 2, 2016 from http://openmp.org/wp/. (2011).

Muhammet Mustafa Ozdal. 2009. Detailed-routing algorithms for dense pin clusters in integrated circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 28, 3, 340–349.

David Z. Pan, Bei Yu, and J.-R. Gao. 2013. Design for manufacturing with emerging nanolithography. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 32, 10, 1453–1472.

Zhongdong Qi, Yici Cai, and Qiang Zhou. 2014. Accurate prediction of detailed routing congestion using supervised data learning. In *Proceedings of the IEEE International Conference on Computer Design (ICCD'14)*. IEEE, 97–103.

Michael C. Smayling, Koichiro Tsujita, Hidetami Yaegashi, Valery Axelrad, Tadashi Arai, Kenichi Oyama, and Arisa Hara. 2013. Sub-12nm optical lithography with 4x pitch division and SMO-lite. In *Proceedings of SPIE*. 868305–868305.

Synopsys. 2012. Synopsys Design Compiler. Retrieved April 2, 2016 from http://www.synopsys.com. (2012).

Taraneh Taghavi, Charles Alpert, Andrew Huber, Zhuo Li, Gi-Joon Nam, and Shyam Ramji. 2010. New placement prediction and mitigation techniques for local routing congestion. In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD'10)*. 621–624.

Zigang Xiao, Yuelin Du, Haitong Tian, and Martin D. F. Wong. 2013. Optimally minimizing overlay violation in self-aligned double patterning decomposition for row-based standard cell layout in polynomial time. In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD'13)*. 32–39.

Xiaoqing Xu, Brian Cline, Greg Yeric, Bei Yu, and David Z. Pan. 2014. Self-aligned double patterning aware pin access and standard cell layout co-optimization. In *ACM International Symposium on Physical Design (ISPD)*. 101–108.

Xiaoqing Xu, Bei Yu, Jhih-Rong Gao, Che-Lun Hsu, and David Z. Pan. 2015. PARR: Pin access planning and regular routing for self-aligned double patterning. In *ACM/IEEE Design Automation Conference (DAC'15)*. 28:1–28:6.

Wei Ye, Bei Yu, David Z. Pan, Yong-Chan Ban, and Lars Liebmann. 2015. Standard cell layout regularity and pin access optimization considering middle-of-line. In *ACM Great Lakes Symposium on VLSI (GLSVLSI'15)*. 289–294.

Yanheng Zhang and Chris Chu. 2011. RegularRoute: An efficient detailed router with regular routing patterns. In *ACM International Symposium on Physical Design (ISPD'11)*. 45–52.

Received July 2015; revised September 2015; accepted October 2015