Concurrent Pin Access Optimization for Unidirectional Routing

Xiaoqing Xu[1], Yibo Lin[1], Vinicius Livramento[2], and David Z. Pan[1]

[1]University of Texas at Austin, Austin, TX, USA
[2]Federal University of Santa Catarina, Brazil

{xiaoqingxu,yibolin}@cerc.utexas.edu, dpan@ece.utexas.edu

ABSTRACT

In advanced technology nodes, standard cell pin access is becoming challenging due to a small number of routing tracks and complex design-for-manufacturing constraints. Pin access interference is further exacerbated by unidirectional routing, which is highly preferred to enable high-density metal patterns and comply with self-aligned multiple patterning solutions. Previous manufacturing-aware routing studies simply depend on the router or sequential planning schemes to resolve pin access interference, which introduces significant overhead on solution qualities. Therefore, we propose concurrent pin access optimization techniques to achieve fast and high-quality routing solutions. The concurrent pin access optimization is modeled as a weighted interval assignment problem, which is solved by an optimal integer linear programming formulation and a scalable Lagrangian relaxation algorithm. A concurrent pin access router is implemented while accommodating advanced manufacturing constraints, which outperforms state-of-the-art manufacturing-aware routers with better routability, fewer vias and faster runtime.

1. INTRODUCTION

As the technology node scales down to $10nm$, standard cell pin access on lower metal layers is becoming difficult due to high-density routing patterns and severe routing resources competitions among a very small number of routing tracks [1, 2]. Meanwhile, unidirectional routing is strongly recommended to provide tight control on lithographic printing for lower metal layers, such as metal-2 (M2) and metal-3 (M3), which also complies with underlying self-aligned multiple patterning (SAMP) techniques (with cut/trim masks) [3–5]. Although unidirectional layout simplifies the coloring scheme during a routing procedure [4–8], it makes local standard cell pin access more challenging due to smaller number of accessing points and more complicated neighborhood interactions. To mitigate the standard cell pin access issue, designers can perform extensive layout optimization with the assistance from local-routing enumeration and optimization [9]. However, this is constrained by modern standard cell architecture, e.g. 7.5-track or 9-track cell architecture in $10nm$ node [1], and a detailed router still takes the responsibility to resolve pin access interference and finish all net connections while accommodating complex metal line-end constraints from SAMP.

An example of unidirectional routing patterns in advanced technology nodes is shown in Figure 1, which includes 4 cells, 9 metal-1 pins ($a_1, a_2, a_3, b_1, b_2, c_1, c_2, d_1, d_2$) and 4 nets ($\{a_1, a_2, a_3\}$, $\{b_1, b_2\}$, $\{c_1, c_2\}$ and $\{d_1, d_2\}$). Suppose all nets need to be connected using unidirectional routing patterns on the M2 and M3 layers. Line-end extensions will be introduced by a router to guarantee patterning-friendly cut masks and specific design rules could be technology dependent [4, 5, 10, 11]. In Figure 1, we also observe severe routing resource competitions among different pins and nets, especially on M2 layer, where only a small number of accessing points is available to connect each standard cell pin [12]. Those accessing points for different pins also interfere with each other as they may share the same routing track. Therefore, it is important to globally allocate routing resources among different pins and nets [9, 13] so that all net connections can be finished.

DAC '17, June 18-22, 2017, Austin, TX, USA
© 2017 ACM. ISBN 978-1-4503-4927-7/17/06…$15.00
DOI: http://dx.doi.org/10.1145/3061639.3062214

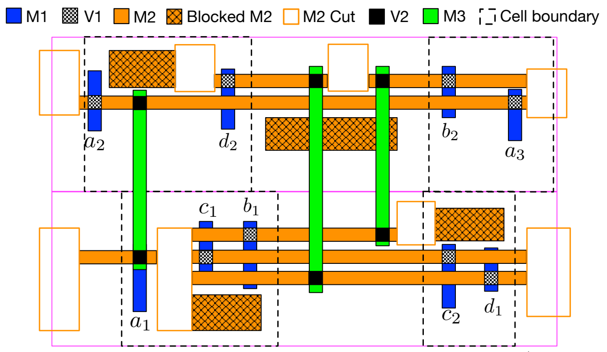

Figure 1: Unidirectional routing patterns shown on M2/M3 layer.

To deal with complex design-for-manufacturing constraints, existing SAMP-aware routing studies mainly focus on the coloring scheme of routing patterns while accommodating specific design rules for cut/trim masks with novel routing grid models [10, 12, 14–21]. Furthermore, [10] and [20] propose to address the cut mask complexities during the track assignment and global routing stages, respectively. The detailed routing procedures of existing studies follow the paradigms of sequential routing [12, 14–17, 19, 20] or negotiation-congestion-based routing schemes [18, 21]. In general, negotiation-congestion-based routing techniques can resolve routing congestions more efficiently than sequential schemes, because a detailed router avoids following a specific net ordering with a history-based heuristic [22]. However, in advanced technology nodes, detailed routing on lower metal layers is becoming more complicated than simply dealing with patterning constraints. The reasons are two-fold. First, the ever-increasing standard cell pin access interference needs to be effectively addressed so that all nets can be routed simultaneously, as shown in Figure 1. Conventional rip-up and reroute scheme generally leads to huge runtime overhead under severe routing resource competitions, which makes concurrent optimization techniques particularly important for fast routing closure. Second, routing resources on lower metal layers are primarily reserved for short nets. For short-net routing, via minimization is particularly important in advanced technology nodes [23].

Recently, [12] has proposed pin access planning schemes to mitigate the standard cell pin access interference while accommodating metal and via patterning constraints. However, [12] follows a sequential routing procedure and pin access interference, i.e. routing resource competition, is not efficiently resolved. To address the routing challenges aforementioned, we propose a concurrent pin access optimization study for unidirectional routing. Our work is unique when compared to the previous detailed routing studies [13, 24, 25] due to following reasons. We demonstrate that concurrent pin access optimization is critical to resolving routing resources competitions, including pin access interference. We perform unidirectional routing while explicitly addressing the complex manufacturing constraints from self-aligned double patterning (SADP). Our contributions are listed as follows.

- We take advantage of the unidirectional routing style to propose track-based pin access interval generation and linear conflict set detection.
- The concurrent pin access optimization is modeled as a weighted interval assignment problem, which is further formulated as a binary integer linear programming problem.
- We propose an iterative Lagrangian relaxation algorithm to obtain scalable solutions for the concurrent pin access optimization problem.
- We implement a concurrent pin access router (CPR) to obtain much better routing results than the state-of-the-art SADP-friendly routers.

The rest of this paper is organized as follows. Section 2 briefly introduces the standard cell pin access interference and problem definition. Section 3 presents the concurrent pin access optimization scheme. Section 4 discusses the overall flow for our concurrent pin access router. Section 5 compares different routing schemes and demonstrates the effectiveness of our proposed routing approach. Section 6 concludes the paper.

2. PRELIMINARIES

2.1 Pin Access Interference

A typical outcome of routing resource competitions on the M2 layer is the standard cell pin access interference among I/O pins close to each other [9]. An example of pin access interference is shown in Figure 2(a), where pin c_1 is blocked on the M2 layer due to the routing patterns connected to other I/O pins. However, pin access interference can be avoided with superior pin access optimization results as shown in Figure 2(b). This illustrative example demonstrates that pin access optimization, including track location and spans of pin access intervals, is critical to resolving the pin access interference the on M2 layer.

In this study, we demonstrate the concurrent pin access problem can be resolved through weighted interval assignment, which is more complex than conventional track assignment problem. Track assignment is an intermediate step between global and detailed routing stages and targets at the maximum assignment of routing intervals from global routing to a set of routing tracks [26, 27]. However, for pin access interference, track assignment is impossible without routing intervals for each I/O pin, which are actually the outcomes of the detailed routing itself. This chicken-and-egg issue makes the pin access optimization problem more difficult than the conventional track assignment problem.

2.2 Problem Definition

The concurrent pin access optimization problem builds on a set of pin access intervals. We take advantage of the unidirectional routing style on lower metal layers and propose track-based pin access interval generation for each I/O pin. Concurrent pin access optimization is performed in a panel-by-panel manner, where a panel corresponds to either a row (horizontal layer) or a column (vertical layer). We define the problem of concurrent pin access optimization without routing conflict as follows.

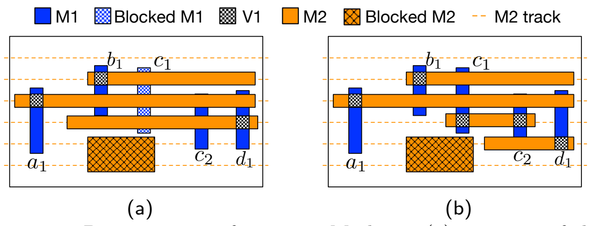

Figure 2: Pin access interference on M2 layer, (a) pin access failure, (b) pin access success [12].

Problem 1 (Concurrent Pin Access Optimization). *Given a netlist, I/O pins and a set of routing tracks, concurrent pin access optimization has to find a set of pin access intervals such that each I/O pin is connected to one pin access interval, while there is no overlap among pin access intervals belonging to different nets.*

Our approach guarantees that a feasible solution exists for Problem 1, which yields conflict-free pin access intervals for all I/O pins. We further implement a concurrent pin access router while accommodating advanced manufacturing constraints [12], which greatly reduces initial routing congestions and leads to high-quality routing solutions.

3. CONCURRENT PIN ACCESS OPTIMIZATION

A design with synthesized power/ground rails is inherently separated into panels, i.e. rows or columns on a horizontal or vertical routing layer. Without loss of generality, concurrent pin access optimization problem is solved panel-by-panel, which can also handle multiple panels simultaneously with scalable solutions.

3.1 Pin Access Interval Generation

We introduce track-based pin access interval generation for each I/O pin, where pin access intervals are generated for each pin within its net bounding box. Related notations are defined in Table 1. Figure 3(a) enumerates the pin access intervals for pin a_1, which is part of the net containing three pins (a_1, a_2, and a_3). The left and right edge of the net bounding box is by a_2 and a_3, respectively. For each I/O pin, a *minimum pin access interval* is the smallest metal strip to cover the pin, while a *maximum pin access interval* is the largest metal strip available within the net bounding box[1]. On track t_1, the maximum pin access interval of pin a_1 starts from the left edge to right edge of the net bounding box. On track t_2, the maximum pin access interval has to stop before the routing blockage. On track t_3, it shows the general scenario for the pin access interval generation for a_1. The net bounding box contains I/O pins (b_1 and d_1) that do not belong to the same net. For a_1, b_1 and d_1 are referred as diff-net pins. Then, when generating the pin access intervals for a_1, it is important to enumerate all the pin access intervals that end at the vertical cutting lines of each diff-net pin, including $I_1^{a_1}$ and $I_2^{a_1}$. The reason is that $I_1^{a_1}$ is the maximum length of interval that can be used to access a_1 without blocking b_1. We do not enumerate all grids points between a_1 and b_1 because a router has the flexibility to choose any grid point on $I_1^{a_1}$ and accommodate metal/via constraints. Similarly, $I_2^{a_1}$ is generated to consider the pin access interference between a_1 and d_1. This principle of generation controls the number of pin access intervals while exactly capturing the pin access interference among I/O pins. There are 8 pin access intervals generated for pin a_1 across 3 tracks.

[1]If M2 routing is not favored for long nets, we can constrain pin access interval generation for each pin using an estimated M2 routing bounding box for its corresponding net, instead of using the net bounding box.

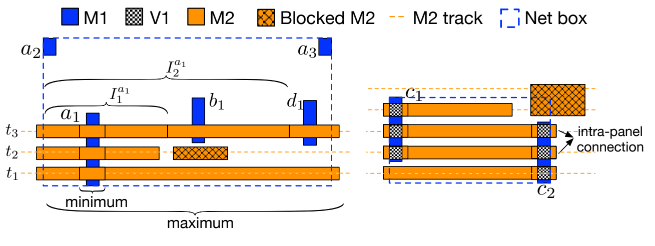

Figure 3: Pin access interval generation, (a) for pin a_1 across 3 tracks, (b) for pin c_1 and c_2 with intra-panel connection.

Suppose a given pin p with m pins on the left and n pins on the right within the net bounding box, the number of pin access intervals generated is $\mathcal{O}(m \cdot n)$ to enumerate all possible left and right edges of a pin access interval. The minimum and maximum pin access intervals can also be generated for pins c_1 and c_2 in Figure 3(b). In particular, those pin access intervals resulting in intra-panel connections shall be preferred in the pin access optimization result, because they connect same-net pins without using external routing resources outside of the panel. Suppose conflict-free pin access intervals are assigned to I/O pins (one interval per pin) with maximum and balanced interval length, this provides much better pin accessibility for a router. Specifically, a router can choose any grid point on the pin access interval of an I/O pin to access that particular pin, without introducing extra pin access interference, i.e. routing congestions. This also helps reduce via numbers because pin access intervals discourage switching metal layers for congestion reductions.

3.2 Linear Conflict Set Detection

After track-based pin access interval generation, the pin access intervals may overlap/conflict with each other. We define a set of intervals to be conflict set if the intersection of the intervals is non-empty. Figure 4 shows an example of conflict interval sets. For a routing track in Figure 4(a), all the intervals on the track are shown in Figure 4(b). There are six conflict interval sets on the track, i.e., $C_0, C_1, \ldots, C_5$. Routing intervals $I_0^{a_1}$, $I_1^{a_1}$, $I_2^{a_1}$, $I_3^{a_1}$ and $I_4^{a_1}$ define the first conflict interval set C_0, because they share a common horizontal range. Similarly, pin access intervals $I_1^{a_1}$, $I_2^{a_1}$, $I_3^{a_1}$, $I_4^{a_1}$ and $I_2^{d_1}$ form another conflict set C_1. The target of conflict detection is to collect all the conflict interval sets on a track without redundancy. In general, this can be realized by generating a vector of intervals and scanning from the left to the right to detect the overlaps. The number of conflict interval sets generated is linear to the size of pin access intervals [26]. As I/O pins overlap a set of tracks, we further collect conflicts among all routing tracks to obtain a complete set of conflict interval sets.

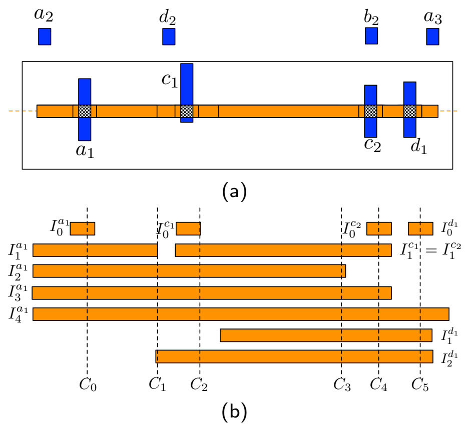

Figure 4: Linear conflict set detection for pin access intervals, which can be applied to multiple tracks obtain complete conflict sets.

Table 1: Notations

P	the set of pins
p_j	j^{th} pin in P
I_i	i^{th} pin access interval
I	the set of pin access intervals
$I_i^{p_j}$	i^{th} pin access interval for p_j
S_j	the complete pin access interval set for p_j on multiple tracks
l_i	the length for I_i
$f(I_i)$	the profit function for I_i
x_i	the binary variable for I_i
C_m	the m^{th} conflict pin access interval set
λ_m	the Lagrangian multiplier for C_m
C	sets of conflict interval set $\{C_m\}$

3.3 Weighted Interval Assignment Problem

Among all pin access intervals for all pins, the concurrent pin access optimization is a weighted interval assignment problem, which aims at assigning conflict-free intervals to I/O pins while maximizing the overall interval length. If a router treats conflict-free intervals as partial routes, maximum and balanced pin access interval lengths among all I/O pins lead to better pin accessibility during a routing procedure. The concurrent pin access problem requires that each pin is assigned exactly one interval to guarantee pin access feasibility, while track assignment targets at the maximum assignment of routing intervals among a set of tracks [26, 27]. This means previous track assignment solutions can not be directly applied here.

To obtain the optimal solution of the current pin access optimization, we formulate the weighted interval assignment problem as an integer linear programming problem. Related notations are defined in Table 1. The objective is to achieve the maximum and balanced length of pin access intervals for all pins without any conflict. For the ILP formulation in Formula (1), a pin access interval I_i corresponds to a binary x_i that indicates whether I_i is selected or not. The objective (1a) explicitly favors the selection of an interval covering multiple I/O pins by counting its corresponding variable multiple times. To obtain an interval selection with balanced interval length, we set $f(I_i) = \sqrt{l_i}$ because the square root function generates more balanced solutions while maximizing the interval length, compared to a linear function. Constraint (1b) denotes only one interval can be selected for pin p_j. Furthermore, only one pin access interval within any conflict interval set should be selected. A simple way to forbid conflicts is to add a constraint $x_i + x_{i'} \leq 1$ for every conflict interval pair $(I_i, I_{i'})$, which introduces large numbers of constraints, i.e., the number of constraints will be quadratic to the number of pin access intervals. To avoid the size explosion of constraints, we utilize the linear conflict interval set detection from Section 3.2. For each conflict set C_m, only one constraint is added to avoid all possible pin access interval conflicts from C_m. This means at most one interval in C_m can be selected and the equivalent linear constraint is shown in (1c). The complete ILP formulation is given as follows.

$$\textbf{max} \quad \sum_{p_j \in P} \sum_{I_i \in S_j} f(I_i) \cdot x_i \tag{1a}$$

$$\textbf{s.t.} \quad \sum_{I_i \in S_j} x_i = 1, \qquad \forall p_j \in P, \tag{1b}$$

$$\sum_{I_i \in C_m} x_i \leq 1, \qquad \forall C_m \in C, \tag{1c}$$

$$x_i \in \{0, 1\}, \qquad \forall I_i \in I. \tag{1d}$$

Due to the minimum pin access interval generation of each I/O pin shown in Figure 3, we have the following theorem.

Theorem 1: *The ILP formulation (1) has a feasible solution.*

PROOF. We can select the minimum pin access interval for each I/O pin, which leads to a set of pin access intervals without conflict. It corresponds to a feasible solution for (1). $\square$

3.4 Lagrangian Relaxation

The main challenge in solving the ILP comes from the conflict constraints in Eq. (1c). To overcome the scalability issue of the ILP formulation, we propose a Lagrangian relaxation (LR)-based approach, which relaxes the conflict constraints to the objective as penalties. The problem formulation for the LR problem is shown in Formula (2). A set of Lagrangian multipliers (LMs) $\{\lambda_m\}$ is introduced to relax the conflict constraints, while the equality constraints are kept in Eq. (1b). In particular, as the ILP problem is feasible according to Theorem 1, the LR problem in Eq. (2a) is bounded during the iterative solving procedure.

$$\textbf{max} \quad \sum_{p_j \in P} \sum_{I_i \in S_j} f(I_i) \cdot x_i - \sum_{C_m \in C} \lambda_m \cdot (\sum_{I_i \in C_m} x_i - 1) \tag{2a}$$

$$\textbf{s.t.} \quad \text{(1b) and (1d).} \tag{2b}$$

For the LR-based algorithm, we first solve the concurrent pin access optimization problem without conflict constraints in Eq. (1c), i.e., all λ_m are set to 0. As will be discussed later, each iteration of LR is solved using an efficient greedy algorithm. After obtaining an initial solution, the pin access interval assignments with conflict constraint violations are detected. For any violation, we gradually increase the penalty in the objective by adjusting the corresponding λ_m. The method for updating λ_m based on the current solution is critical for the convergence of the LR-based algorithm. To guarantee convergence, we adopt the subgradient descent method for updating LMs, defined in Eq. (3), where k is the number of iterations and t_k is the step size [28]. In our experiments, we set t_k to $1/k^\alpha \cdot L_m$, where α is 0.95 and L_m is the length of intersection among all conflict intervals in C_m.

$$\lambda_m^{k+1} = \max(0, \lambda_m^k + t_k \cdot (\sum_{I_i \in C_m} x_i - 1)). \tag{3}$$

Algorithm 1 LR Sub-Routines

```
 1: function maxGains($I, gains$)
 2:     Define $sel$ as selected intervals;
 3:     Sort $I$ in the non-increasing order of $gains$;
 4:     Select intervals from $I$ to $sel$ until all pins are covered;
 5:     Return $sel$;
 6: end function
 7: function penalize($sel, C, penalties$)
 8:     Define $vio \leftarrow 0$ as the violation number;
 9:     for each conflict $C_m \in C$ do
10:         if more than 1 interval selected in $sel$ from $C_m$ then;
11:             $vio \leftarrow vio + 1$;
12:             Update $penalties[i]$ for each $I_i \in C_m$ (Eq. (3));
13:         end if
14:     end for
15:     Return $vio$;
16: end function
```

In each iteration to solve the LR subproblem, the objective defined in Eq. (2a) is a weighted summation of x_i with fixed LMs. Let the weight of each x_i represent the gain of the interval. We restate the Formula (2) as follows. Given a set of pins and corresponding intervals with different gains, select one interval for each pin such that the total gain is maximized. We develop a greedy algorithm to compute the maximum total gain efficiently, shown as function `maxGains` in Algorithm 1. All the intervals are sorted in the non-increasing order of *gains* in line 3. We break the ties of equal gains by the number of same-net pins an interval covers. We prefer selecting pin access intervals covering more same-net I/O pins because intra-panel connections are preferred as mentioned in Section 3.1. As one pin can only be assigned one pin access interval, we skip an interval if its corresponding pin has already been assigned another interval in line 4. The penalty of each interval I_i is defined as the summation of its corresponding LMs in Eq. (2a). In function `penalize`, we scan through each conflict (C_m) in lines 9 to 14. If more than 1 interval is selected within C_m, the corresponding penalty is updated in line 12.

Algorithm 2 LR-based Pin Access Optimization

```
Input: Pins ($P$), pin access interval vector ($I$), complete conflict interval sets ($C$) and iteration upper bound ($UB$).
Output: A vector of pin access intervals $PI$ for $P$.
 1: Initialize $profits$ as the profit vector for $I$ using Eq. (2a);
 2: Define $penalties$ as penalty vector with zeros for $I$;
 3: Define $k \leftarrow 0$, $min\_vio \leftarrow \infty$;
 4: while $min\_vio > 0$ and $k < UB$ do;
 5:     $sel \leftarrow$ maxGains($I, profits - penalties$);
 6:     $vio \leftarrow$ penalize($sel, C, penalties$);
 7:     if $vio < min\_vio$ then
 8:         $min\_vio \leftarrow vio$, $PI \leftarrow sel$;
 9:     end if
10: end while
11: Greedy conflict removal among $PI$;
12: Return $PI$;
```

An iterative LR approach generates good convergence behavior if we can optimally solve LR subproblem. Given notations in Table 1, the optimality condition for the greedy algorithm is analyzed as follows.

Theorem 2: *The greedy LR sub-routines in Algorithm 1 solves Formula (2) optimally if $S_i \cap S_j = \emptyset, \forall p_i, p_j \in P$.*

PROOF. $S_i \cap S_j = \emptyset, \forall p_i, p_j \in P$ means that no pin access interval is associated with more than one pin. Therefore, we greedily select pin access interval with the maximum profit for each pin, which delivers optimal solution to Formula (2). □

Although intra-panel routing connections may introduce dependency among pin access interval set for I/O pins, i.e. breaking the optimality condition in Theorem (2), the greedy algorithm empirically works well in computing the solution and reaching convergence in our experiments. The LR-based algorithm is shown in Algorithm 2. We first compute the profit of each interval in line 1. Within each iteration of the kernel loop from line 4 to line 10, the relaxed problem is solved by our greedy algorithm in `maxGains` in line 5. In line 6, we compute the penalty for each interval and update LMs according to Eq. (3) in `penalize`. The LR iterations continue until there is no violation or the iteration upper bound is reached. We observe small oscillations on the solutions when the number of violations approaches zero, so we record the best solution throughout the LR iterations in lines 7 to 9. The LR scheme usually reduces violations rapidly, but it cannot guarantee zero conflicts. Greedy refinement is performed to remove remaining conflicts in line 11. For any conflict interval set, this refinement process tries to shrink conflict intervals into minimum intervals to remove that particular conflict.

4. CONCURRENT PIN ACCESS ROUTING

The routing resource competitions among those pin access intervals from concurrent pin access optimization have been efficiently resolved. Therefore, we implement concurrent pin access router (CPR) to take advantage of these pin access intervals and obtain net connections. The pin access intervals are treated as partial routes connected to its associated pins, which are fed into a negotiation-congestion-based router [21]. As shown in Figure 5(a), the M2 pin access intervals from pin access optimization are connected to pins a_1, a_2 and a_3. Some detour may happen as metal-1 (M1) pins are connected to pin access intervals (partial routes) on the M2 layer. In Figure 5(b), the routing results

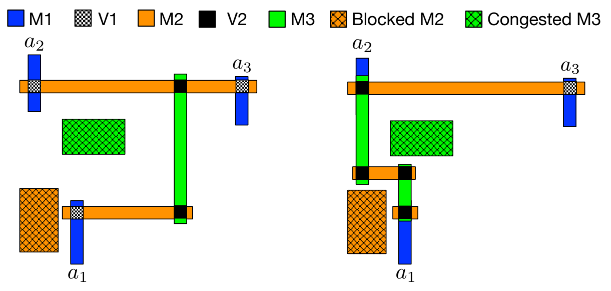

Figure 5: (a) Routing with pin access optimization introducing extra routed wirelength; (b) Routing without pin access optimization introducing extra routed vias.

are obtained directly with a negotiation-congestion-based routing scheme. This introduces more vias because the router tries to avoid congested routing regions and routing blockages with unidirectional routing patterns. However, via minimization is particularly important for short-net routing in advanced technology nodes [23].

Given the netlist and routing grid plane, CPR starts with concurrent pin access optimization on lower metal layers, which efficiently resolves routing resource competitions and greatly reduces the number of grid congestions for the subsequent routing steps. Treating pin access intervals as partial routes, the negotiation-congestion-based routing is performed to finish all net connections, where design rule violations are mitigated using forbidden grid costs [21]. To make the negotiation-congestion-based routing aware of standard cell pin access, during the routing of each net, only the pins and associated pin access intervals of the current net are available, while the pins and associated pin access intervals from remaining nets are treated as blockages. We further perform line-end extensions and rip-up and reroute to accommodate the manufacturing constraints listed in [12] and enable SADP-friendly cut masks shown in Figure 1. CPR is extendable to technology-dependent manufacturing constraints, e.g. SAMP with unidirectional routing.

5. EXPERIMENTAL RESULTS

We implement concurrent pin access router (CPR) in C++ and all experiments are performed on an 8-core Linux machine with 3.4GHz Intel®core and 32GB memory. The iteration upper bound (UB) for LR is set as 200. For the grid cost computation, the base cost is set as 1 for metal and via grids. The forbidden cost is set as 10 for via grids. We perform the concurrent pin access optimization on the M2 layer, where one standard cell row (10 x M2 tracks) is one routing panel. All experiments are performed using the same benchmarks, design rule settings and evaluation metrics listed in [12]. We quantify routing solutions with the number routed nets over the total number of nets ("Rout."), the number of vias ("Via#") and wirelength ("WL"). "Via#" is the total number of vias for all nets estimated by via per routed net [12]. "WL" is the summation of half perimeter wirelength of unrouted nets and actual grid wirelength for routed nets [12].

5.1 Quantifying the LR Suboptimality

The comparisons on runtime scalability and solution quality in terms of objective value between LR and ILP are shown in Figs. 6(a) and 6(b), respectively. For Figure 6(a), we observe that the runtime of ILP is not scalable to a large number of pins and the runtime of LR algorithm has much better scalability. As shown Figure 6(b), the objective value achieved with LR algorithm is pretty close to the optimal value returned by ILP solution. Therefore, our proposed LR algorithm delivers a scalable solution to the concurrent pin access optimization problem.

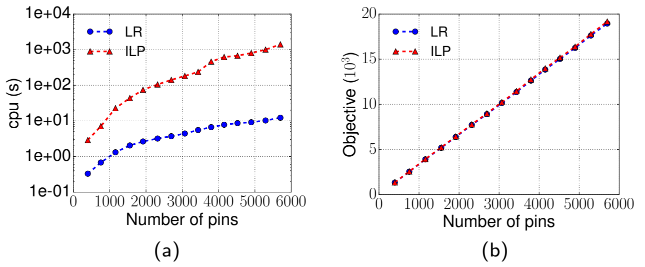

Figure 6: Comparisons between LR and ILP for different number of pins, (a) runtime, (b) objective value.

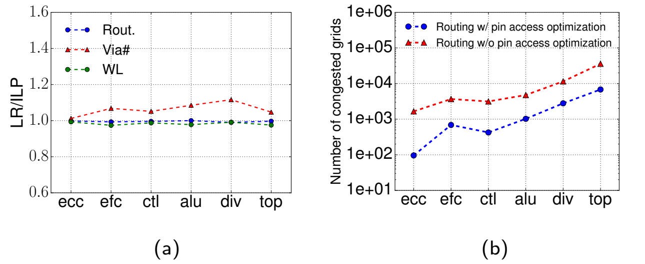

Figure 7: (a) LR over ILP after obtaining the routing solutions, (b) the number congested routing grids before rip-up and reroute stage.

Figure 7(a) further demonstrates the comparisons on routing solution qualities obtained with LR-based and ILP-based pin access optimizations. We observe that the "Rout." and "WL" are similar across different designs with LR-based and ILP-based pin access optimization schemes. Furthermore, the "Via#" for the routing solution with LR-based pin access optimization is around 5% more than that with ILP-based pin access optimization. The reason is that ILP-based optimization delivers optimal and balanced pin access intervals for each I/O pin, which leaves more flexibilities for via locations during the routing phase. The LR-based optimization depends on a greedy conflict removal phase to generate legal pin access intervals for each I/O pin, which penalizes the solution qualities of pin access optimization. We believe the LR-based pin access optimization represents desirable tradeoff between solution qualities and runtime scalability compared to the ILP-based approach.

5.2 Comparison with Related Work

Table 2 compares concurrent pin access router (CPR) with two state-of-the-art SADP-friendly routers [12,21]. [12] proposes sequential pin access planning schemes with design rule legalizations during sequential routing of each net. [21] devises a negotiation-congestion-based router with modified routing grid models for SADP, with explicit considerations of pin access and routing blockages[2]. To make the negotiation-congestion-based routing aware of standard cell pin access, during the routing of each net, only the pins of the current net are available, while the pins from remaining nets are treated as blockages [21]. For [21], the design rules violations are mitigated with forbidden via grid cost and rip-up and reroute iterations. CPR combines the concurrent pin access optimization results with the same negotiation-congestion-routing scheme as [21]. Although [21] and our router can initially finish all net connections (100% routability), they introduce many design rule violations because design rule violations can only be mitigated through grid cost manipulations. We treat those nets introducing violations as unrouted nets, which generates design-rule-clean routing results for fair comparisons.

CPR obtains 1.5% routability improvement, 23.8% via number and 16.0% wirelength reductions compared to [12] as shown in

[2]The authors of [21] make their router explicitly consider design rules in [12] and run on a Linux machine with 2.4GHz Intel(i5) core and 8GB memory.

Table 2: Comparisons on solution qualities of different routing approaches

Ckt	Net#	Size(um^2)	Sequential pin access planning [12]				Routing w/o pin access optimization [21]				CPR			
			Rout.(%)	Via#	WL	cpu(s)	Rout.(%)	Via#	WL	cpu(s)	Rout.(%)	Via#	WL	cpu(s)
ecc	1671	21x21	96.41	6482	46588	19.98	94.55	5409	38428	10.00	97.25	4907	40465	2.01
efc	2219	20x19	94.91	8558	57834	34.52	92.83	7989	52329	15.60	96.80	7418	51973	3.69
ctl	2706	24x24	95.27	10573	72388	37.14	92.42	9327	64217	17.80	96.86	8757	63900	3.69
alu	3108	20x19	95.17	11645	75679	45.92	93.37	10496	64604	20.10	97.01	9371	62249	5.24
div	5813	31x31	94.60	22829	155704	106.0	92.12	21001	139811	47.70	95.89	19665	139201	24.32
top	22201	57x56	95.33	82644	513366	763.2	93.44	73487	434051	147.4	96.79	65167	436972	40.37
Avg.			95.28	23789	153593	167.8	93.12	21285	132240	43.1	96.77	19214	132460	13.22
Ratio			0.985	1.238	1.160	12.69	0.962	1.108	0.998	3.26	1.000	1.000	1.000	1.000

Table 2. The reason is that the baseline router in [12] depends on pin access planning schemes to improve routability, which resolves routing resource competitions in a greedy manner to generate design-rule-clean routing results. Although net deferring technique enables dynamic net reordering, the routing scheme in [12] is still sequential in nature, which introduces significant overhead in terms of runtime, wirelength and via number. Our approach provides the maximum and balanced amount of horizontal M2 partial routes connected to each pin, which leads to more flexible via locations and pin access. We further compare with [21] (routing w/o pin access optimization), the data of which are directly from the authors of [21]. CPR improves the "Rout." by 3.8% and reduces the "Via#" by 10.8%. And the trade-off is only 0.2% "WL" degradation. In particular, this "WL" degradation is negligible considering 3.8% more routability from CPR because remaining nets from [21] are difficult to route and may introduce more WL overhead. Via# reductions and slight WL degradations are both due to the partial routes from pin access intervals shown in Figure 5(a). From the solutions of [21], there exist routing patterns shown in Figure 5(b), which means the router tries to switch between metal layers to resolve routing congestion and minimize routing cost. CPR reduces "Via#" by $> 10\%$ with pin access optimization as shown in Figure 5(b) in spite of some detour. In advanced technology nodes, via minimization is particularly important as the via resistance has been increasing dramatically [23].

The runtime of our router consists of the runtime from concurrent pin access optimization and unidirectional routing. In Table 2, our router is significantly faster (12.7×) than [12] and estimated to be 2.5× faster than [21] considering different machine configurations. The major reason is that the routing resource competitions on the M2 layer are concurrently resolved using pin access optimization approach, which greatly reduces the initial congested routing grids for the router. Congested routing grids are those grids occupied by more than one net. The sequential pin access planning depends on detours and rip-up and reroute to avoid congestions, which makes the runtime increases significantly as the design size becomes larger. [21] depends on history cost and rip-up and reroute to reduce congestions, which is computationally more expensive than our pin access optimization approach in terms of congestion reductions. We demonstrate efficient congestion reductions from concurrent pin access optimization in Figure 7(b). The negotiation-congestion routing consists of two stages, i.e. independent routing stage and rip-up and reroute stage [21]. The rip-up and reroute stage takes congested routing grids as input and resolves grid congestion one by one. With concurrent pin access optimization, we achieve a 5-10× reduction in terms of congested routing grids, which greatly reduces the rip-up and reroute efforts. This further generates significant speed-up compared to [12] and [21].

6. CONCLUSION

The concurrent pin access optimization is modeled as a weighted interval assignment problem and solved optimally with an ILP formulation. We further propose an LR-based algorithm for scalable solutions to the weighted interval assignment problem, which leads to similar routing solution qualities as an ILP-based approach. Due to the 5-10× reduction in initial routing congestions, CPR generates much better routing solutions than the state-of-the-art SADP-friendly routers.

Acknowledgement

We would like to thank Dr. Bei Yu from CUHK for his helpful discussions. This work is supported in part by NSF, SRC, SPIE BACUS and university graduate continuing fellowship.

7. REFERENCES

[1] L. Liebmann, A. Chu, and P. Gutwin, "The daunting complexity of scaling to 7nm without EUV: Pushing DTCO to the extreme," in *Proc. of SPIE*, 2015, pp. 942 702–942 702.

[2] D. Z. Pan, L. Liebmann, B. Yu, X. Xu, and Y. Lin, "Pushing multiple patterning in sub-10nm: are we ready?" in *DAC*, 2015, pp. 197:1–197:6.

[3] D. Z. Pan, B. Yu, and J.-R. Gao, "Design for manufacturing with emerging nanolithography," *TCAD*, vol. 32, no. 10, pp. 1453–1472, 2013.

[4] W. Gillijns *et al.*, "Impact of a SADP flow on the design and process for N10/N7 metal layers," in *Proc. of SPIE*, 2015, pp. 942 709–942 709.

[5] Y. Wang *et al.*, "Study of cut mask lithography options for sub-20nm metal routing," in *Proc. of SPIE*, 2015, pp. 94 260J–94 260J.

[6] X. Xu, B. Cline, G. Yeric, and D. Z. Pan, "Standard cell pin access and physical design in advanced lithography," in *Proc. of SPIE*, 2016, pp. 97 800P–97 800P.

[7] B. Yu, X. Xu, S. Roy, Y. Lin, J. Ou, and D. Z. Pan, "Design for manufacturability and reliability in extreme-scaling vlsi," *Science China Information Sciences*, vol. 59, no. 6, p. 061406, 2016.

[8] X. Xu and D. Z. Pan, "Toward unidirectional routing closure in advanced technology nodes," *IPSJ Transactions on System LSI Design Methodology*, vol. 10, pp. 2–12, 2017.

[9] X. Xu, B. Cline, G. Yeric, B. Yu, and D. Z. Pan, "Self-aligned double patterning aware pin access and standard cell layout co-optimization," *TCAD*, vol. 34, no. 5, pp. 699–712, 2015.

[10] S.-Y. Fang, "Cut mask optimization with wire planning in self-aligned multiple patterning full-chip routing," in *ASPDAC*, 2015, pp. 396–401.

[11] K. Han, A. B. Kahng, and H. Lee, "Evaluation of BEOL Design Rule Impacts Using an Optimal ILP-based Detailed Router," in *DAC*, 2015, pp. 68:1–68:6.

[12] X. Xu, B. Yu, J.-R. Gao, C.-L. Hsu, and D. Z. Pan, "PARR: Pin access planning and regular routing for self-aligned double patterning," in *DAC*, 2015, pp. 28:1–28:6.

[13] M. M. Ozdal, "Detailed-routing algorithms for dense pin clusters in integrated circuits," *TCAD*, vol. 28, no. 3, pp. 340–349, 2009.

[14] M. Mirsaeedi, J. A. Torres, and M. Anis, "Self-aligned double-patterning (SADP) friendly detailed routing," in *Proc. of SPIE*, 2011, pp. 79 740O–79 740O.

[15] J.-R. Gao and D. Z. Pan, "Flexible Self-aligned Double Patterning Aware Detailed Routing with Prescribed Layout Planning," in *ISPD*, 2012, pp. 25–32.

[16] C. Kodama *et al.*, "Self-Aligned Double and Quadruple Patterning-Aware Grid Routing with Hotspots Control," in *ASPDAC*, 2013, pp. 267–272.

[17] F. Nakajima *et al.*, "Detailed routing with advanced flexibility and in compliance with self-aligned double patterning constraints," in *Proc. of SPIE*, 2013, pp. 86 840A–86 840A.

[18] Y. Du, Q. Ma, H. Song, J. Shiely, G. Luk-Pat, A. Miloslavsky, and M. D. Wong, "Spacer-is-dielectric-compliant detailed routing for self-aligned double patterning lithography," in *DAC*, 2013, pp. 93:1–93:6.

[19] I.-J. Liu, S.-Y. Fang, and Y.-W. Chang, "Overlay-aware detailed routing for self-aligned double patterning lithography using the cut process," in *DAC*, 2014, pp. 50:1–50:6.

[20] Y.-H. Su and Y.-W. Chang, "Nanowire-aware routing considering high cut mask complexity," in *DAC*, 2015, pp. 138:1–138:6.

[21] Y. Ding, C. Chu, and W.-K. Mak, "Detailed routing for spacer-is-metal type self-aligned double/quadruple patterning lithography," in *DAC*, 2015, pp. 69:1–69:6.

[22] L. McMurchie and C. Ebeling, "PathFinder: A Negotiation-based Performance-driven Router for FPGAs," in *ACM Symposium on FPGAs*, 1995, pp. 111–117.

[23] A. Domic, "Some observations on the physical design of the next decade," in *ISPD*, 2016, pp. 61–61.

[24] Y. Zhang and C. Chu, "RegularRoute: An efficient detailed router with regular routing patterns," in *ISPD*, 2011, pp. 45–52.

[25] M. Ahrens, M. Gester, N. Klewinghaus, D. Muller, S. Peyer, C. Schulte, and G. Tellez, "Detailed routing algorithms for advanced technology nodes," *TCAD*, vol. 34, no. 4, pp. 563–576, 2015.

[26] R. Kay and R. A. Rutenbar, "Wire packing: a strong formulation of crosstalk-aware chip-level track/layer assignment with an efficient integer programming solution," in *ISPD*, 2000, pp. 61–68.

[27] S. Batterywala, N. Shenoy, W. Nicholls, and H. Zhou, "Track Assignment: A Desirable Intermediate Step Between Global Routing and Detailed Routing," in *ICCAD*, 2002, pp. 59–66.

[28] M. Held, P. Wolfe, and H. P. Crowder, "Validation of subgradient optimization," *Mathematical programming*, vol. 6, no. 1, pp. 62–88, 1974.