Journal of
Micro/Nanolithography,
MEMS, and MOEMS

Nanolithography.SPIEDigitalLibrary.org

Layout pattern-driven design rule evaluation

Yasmine Badr
Ko-wei Ma
Puneet Gupta

SPIE.

Layout pattern-driven design rule evaluation

Yasmine Badr,[a,*] Ko-wei Ma,[b] and Puneet Gupta[a]
[a]University of California, Electrical Engineering Department, 53-109 Engineering IV Building, Los Angeles, California 90095, United States
[b]NVIDIA Inc., 2701 San Tomas Expressway, Santa Clara, California 95050, United States

Abstract. With the use of subwavelength photolithography, some layouts can have low printability and, accordingly, low yield due to the existence of bad patterns even though they pass design rule checks. A reasonable approach is to select some of the candidate bad patterns as forbidden. These are the ones with a high yield impact or low routability impact, and these are to be prohibited in the design phase. The rest of the candidate bad patterns may be fixed in the postroute stage in a best-effort manner. The process developers need to optimize the process to be friendly to the patterns of high routability impact. Hence, an evaluation method is required early in the process to assess the impact of forbidding layout patterns on routability. We propose pattern-driven design rule evaluation (pattern-DRE), which can be used to evaluate the importance of patterns for the routability of the standard cells and, accordingly, select the set of bad patterns to forbid in the design. The framework can also be used to compare restrictive patterning technologies [e.g., litho-etch-litho-etch (LELE), self-aligned double patterning (SADP), self-aligned quadruple patterning (SAQP), self-aligned octuple patterning (SAOP)]. Given a set of design rules and a set of forbidden patterns, pattern-DRE generates a set of virtual standard cells; then it finds the possible routing options for each cell without using any of the forbidden patterns. Finally, it reports the routability metrics. We present a few studies that illustrate the use cases of the framework. The first study compares LELE to SADP by using a set of forbidden patterns that are allowed by LELE but not by SADP. Another study compares LELE to extreme ultraviolet lithography from the routability aspect by prohibiting patterns that have LELE native conflicts. In addition, we present a study that investigates the effect of placing the active area of the transistors close to the P/N interface instead of close to the power rails. [DOI: 10.1117/1.JMM.13.4.043018]

Keywords: design rules; manufacturability; bad patterns; hotspots; design for manufacturing; design technology co-optimization; layouts.

Paper 14108P received Jul. 2, 2014; accepted for publication Nov. 3, 2014; published online Dec. 17, 2014.

1 Introduction

As the semiconductor industry continues to use subwavelength photolithography, new printability problems arise. Some layouts can pass design rule check, but will have bad patterns, patterns that have low printability. There are two extreme candidate solutions to the bad patterns problem. The first solution is to handle that problem in the design stage by prohibiting all candidate bad patterns from appearing in the design. However, disallowing all those patterns can make the standard cell routability very hard, and this, in turn, can lead to a tremendous increase in the standard cell area. An alternative, but also an extreme solution is to allow all bad patterns in the design phase and then later, after routing, try to legalize the layout in order to eliminate those bad patterns. Yet, at this stage, it may be too late to fix all those patterns. Thus, a hybrid approach is recommended where a set of forbidden patterns is disallowed in the design phase. Then later, after routing, try to fix the remaining bad patterns in a best-effort manner. As a result, we need to answer the question of which patterns to select as forbidden patterns. A forbidden pattern needs to have high yield impact or low routability impact. High yield impact patterns can be identified by lithography simulation. Low routability impact patterns are those in which, if forbidden in the design stage, the routability of the standard cells and the design will not be drastically hurt. In other words, we can still route the design even with those patterns being forbidden.

Another problem that is similar to the bad patterns problem is the emergence of restrictive patterning technologies, like double patterning [litho-etch-litho-etch (LELE) and self-aligned double patterning (SADP)], triple patterning, quadruple patterning, and beyond. Each of those restrictive patterning technologies has some nonmanufacturable patterns. An essential question arises for foundries: which technology to adopt for the next node.

Thus, an evaluation method is required early in the process to assess the effect of prohibiting some forbidden patterns on the routability. In this work, we propose pattern-driven design rule evaluation (pattern-DRE), which can be used to assess the sensitivity of the standard cell routability to the patterns and design rules and can be used to compare restrictive patterning technologies from the point of view of standard cell routability. It can also be used to count the occurrence of the undesired patterns as the design rules change. In addition, the framework can also be used to guide the process development on the relative importance of the various patterns and accordingly can indicate, from a design perspective, the patterns for which the process needs to be optimized. A high-level overview of the framework is shown in Fig. 1, where the framework uses a set of design rules, candidate forbidden patterns, and the transistor-level netlists of the standard cells and then reports routability metrics as an output.

*Address all correspondence to: Yasmine Badr, E-mail: ybadr@ucla.edu

0091-3286/2014/$25.00 © 2014 SPIE

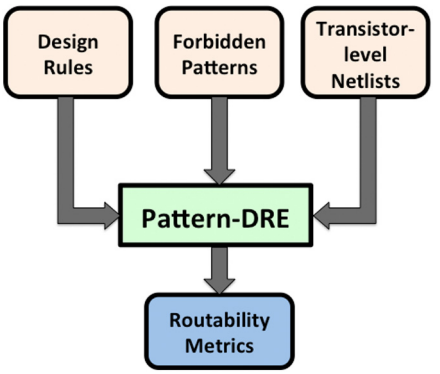

Fig. 1 Overview of pattern-driven design rule evaluation (pattern-DRE) framework.

1.1 *Prior Work*

To the authors' knowledge, this is the first attempt to systematically evaluate patterns along with design rules and study the sensitivity of routability to the patterns. In our previous work,[1] we proposed DRE, a framework for the systematic evaluation of design rules, layout styles, and library architectures. However, DRE was not pattern-aware, and hence, we propose pattern-DRE in this work. Since the use-context of our framework is related to handling bad patterns (aka hotspots) and comparing patterning technologies, we discuss the work that has tackled both topics here.

Several works have addressed the problem of hotspot or pattern-aware design, i.e., using a correct-by-construction approach. The methodologies in Refs. 2 and 3 apply template-based correct-by-construction design. These methods are very promising since they can be used to achieve micro and macro levels of granularity. However, they require very high effort in the design of the templates, whether done manually or automatically by register-transfer level synthesis or other template library creation methods. Thus, these methods may not be appropriate for a technology exploration phase, which requires evaluating a lot of alternatives in a fast and automated fashion. Reference 1 used conservative rules to have correct-by-construction standard cell layouts that are compatible with LELE and SADP and performed a comparison study. However, the conservative rules used can waste a lot of area, and this can skew the results of the comparison between the patterning technologies. In Ref. 4, a lithography-aware router was proposed, which used a printability metric to guide the router. Another approach was developed in Ref. 5, which used routing path prediction along with lithography simulation and a hotspot prediction kernel to construct lithography-friendly routes.

As opposed to the correct-by-construction techniques, a lot of work focused on the detection and correction of those bad patterns after the design stage. Several works[6–13] used various techniques of machine learning or fuzzy pattern matching to identify the hotspots in the design. References 14 and 15 suggested a flow that integrates a pattern checker, a pattern fixer, and a router, such that the router completes its job, then the pattern check and fix are performed if needed, and some routes are tentatively redone. Reference 16 also used rip-up and reroute to build a router that is resolution-enhancement techniques aware. Similar to the postdesign hotspot detection, Ref. 17 proposed using—in addition to the design rule check—a pattern matcher to detect the short-range patterns that are incompatible with double patterning and applying fixes to them.

To explore design rules for multiple patterning technologies, Ref. 18 used machine learning techniques to predict the number of conflicts, which can be used to compare several sets of design rules. Reference 19 suggested optimizing the design rules for double patterning technologies in an iterative flow, where in each iteration, test layouts are generated and decomposed, lithography simulation is performed, the impact on the design is analyzed, and, accordingly, the design rules are optimized.

A lot of work focused on developing multiple patterning-aware routers, like LELE-aware routing,[20–22] triple patterning-aware routing,[23] SADP-, and SAQP-aware routing.[24,25]

However, none of these works offered a pattern-centric design rule evaluation method, and this is the main contribution of our work. The rest of this paper is organized as follows. Section 2 explains the flow of the framework, breaks down each module used into detail, and shows how pattern-DRE can be used to make decisions about forbidden patterns. In Sec. 3, we show how we validated pattern-DRE, and then we illustrate some studies that have been performed using pattern-DRE. Finally, we present the conclusions and future work in Sec. 4.

2 Pattern-DRE Flow

In this section, we explain the flow of the pattern-DRE framework, which is illustrated in Fig. 2. The input to pattern-DRE is the set of design rules, transistor-level netlist for the standard cell, and a set of forbidden patterns. Pattern-DRE generates a virtual standard cell library and studies the possible routing options for each cell while avoiding the given forbidden patterns. After generating the front-end layers, the cell may not be routable. In such a case, the standard cell is generated in a different way and the routing is reattempted until it becomes routable or we reach a certain number of trials (further details are provided in Sec. 2.1). Routability metrics are reported by the framework at the end. In addition, the count of all occurring patterns are reported. In the following subsections, the details of each block in the flow will be explained.

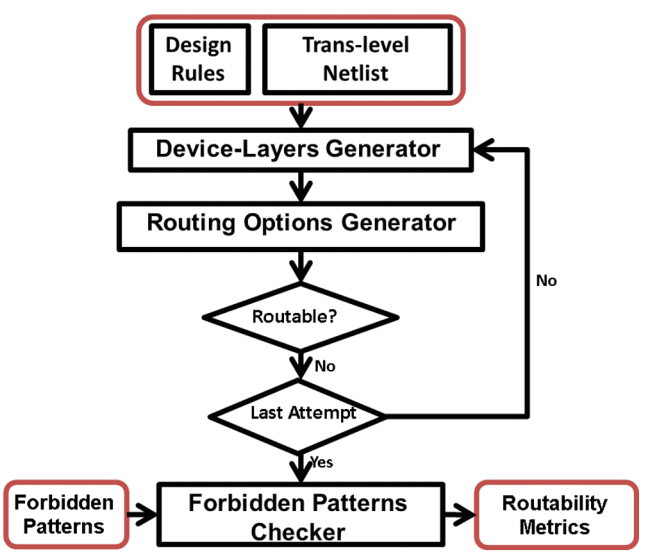

Fig. 2 : Flow of pattern-DRE.

2.1 *Device-Layers Generator*

Pattern-DRE first generates the essential device layers for the given standard cells. This includes building the required transistors based on the given design rules and transistor-level netlists for the cells. We use the device-layers generator of DRE.[1] As part of the device-layers generation, the contact locations forming the nets are generated. (A change has been performed on the gate contact locations generated by DRE such that the gate contacts are not all generated on the same horizontal level; instead, connected gates have contacts that are aligned at the same y-location for ease of routing, but unconnected gates can have their poly contacts at different y-locations. This improves the routability of the cells.) The nets along with their contact locations are used as inputs to the next module.

As shown in Fig. 2, the device-layers generator can be invoked again for a few iterations if the cell is found unroutable. In such a case, the abutment of transistors is done in a different way. The chaining step[1] then chooses a suboptimal solution with respect to area in order to give another chance for the routability of the cell.

2.2 *Routing Options Generator*

This module mimics a router and tries to find possible ways in which the nets of each cell can be routed.

Given all the nets in the cell, the routing options generator generates a list of candidate wiring solutions for each cell. Instead of routing with a specific topology, we try to enumerate all possible routing options under a single trunk Steiner tree[26] topology type. The wiring solutions for each net are generated as presented in Ref. 18. Starting with each net, the bounding box is determined according to the contact locations inside that net. If the width or height of net bounding box lies below a certain threshold, then we expand the bounding box by a few tracks in order to allow detours for the net. (In our experiments, we used a threshold of one track and we expanded the bounding box to three tracks in such a case.) In addition, if the bounding box is too skewed in a certain direction, then having a single trunk Steiner tree trunk along the short direction will lead to an unnecessarily long wire length, as shown in Ref. 18. The possible wiring solutions for each net are constructed by placing the tree trunk at each of the tracks within the bounding box and then constructing perpendicular branches from the trunk to reach out to each contact. [To avoid confusion, when we mention a wiring solution, we are referring to a way to route the net, but when we say routing option, we are pointing to one way to route all the nets in the cell (i.e., a set of wiring solutions, one for each net).] With all the wiring solutions for each net, we need to construct complete routing options for each standard cell. Not all combinations will form valid routing options for the cell because some routes from different nets can cross/intersect. An example showing a possible conflict between routes of two different nets is shown in Fig. 3(a) and another example showing a valid routing option is shown in Fig. 3(b). After we discuss the pattern representation that we use, we will illustrate how the check for conflicts between wiring solutions of nets is performed.

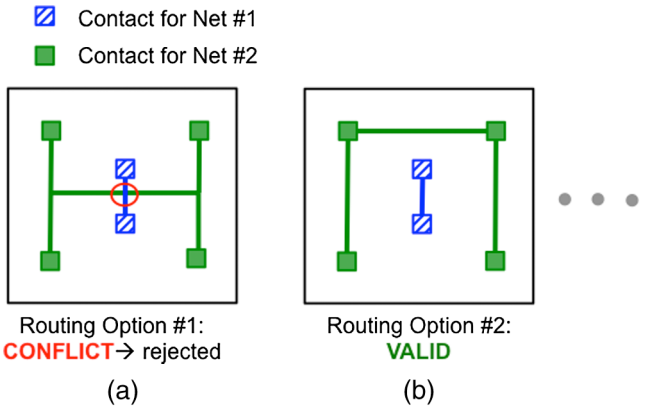

Fig. 3 An invalid routing option (a) because of a conflict between the routes of the nets and a valid routing option (b).

2.2.1 *Layout and tile/pattern representation*

The layout is represented as a 2-D matrix of tiles. Each tile has two representations: segment representation and node representation. The same representation is used for the tiles in the layout and the patterns, except that the tile has fixed size (2×2 tracks) while the size of the patterns is specified as an input. (Currently, the maximum allowed pattern size is 5×5 tracks.) All wiring is assumed to be on-track and with a uniform width.

- Segment representation: Intersection of wiring tracks break themselves into segments and the segment representation encodes the presence/absence of a wire between the tracks in the opposite direction. The rows and columns are then serialized as a binary string and the equivalent decimal number is used as the pattern segment representation. An example of the segment representation of a tile/pattern is shown in Fig. 4(a), where the rows are read first from left to right followed by columns from bottom to top (first segment occupies the least significant bit). Then the equivalent number formed by the binary string is used as the segment representation for the tile. The segment representation is required because it uniquely identifies the pattern.
- Node representation: A node is the intersection of a vertical and a horizontal track. So the node representation encodes whether or not each node is occupied. (A node is occupied if any of its neighboring segments is occupied.) An example for the node representation is shown in Fig. 4(b). The node representation is required for the conflict detection, which will be explained shortly.

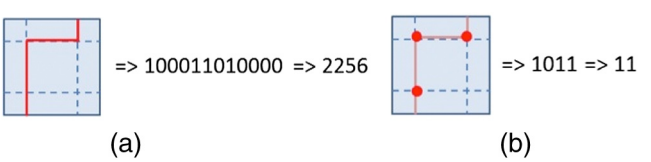

Fig. 4 Segment and node representations for tile/pattern: (a) the segment representation; columns and rows are read off into a binary string (100011010000); then the equivalent decimal number (2256) is used as the segment representation. (b) The node representation; nodes are serialized as a binary string (1011); then the decimal equivalent (11) is used as the node representation.

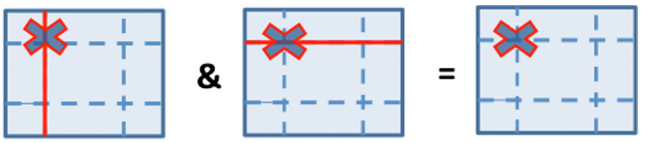

Fig. 5 Checking conflicts between routing options of different nets by ANDing node representations.

2.2.2 *Conflict checker*

Two wiring solutions for two different nets are conflicting if their segments overlap or cross. These cases can be checked by doing an AND operation between the node representations of the routing options in each tile. If the result of the AND operation is nonzero for any tile, then there is a conflict. The reason for doing the conflict check on the node representation is that some conflict cases cannot be detected in the segment representation. For example, a vertical and horizontal wire will not have any common segments but will have common nodes. This is the case illustrated in Fig. 5.

2.2.3 *Minimum number of unroutable nets*

In some cases, the routing options generator may fail to find a conflict-free routing option for the cell. In such a case, it reports the routing option with the minimum number of unroutable nets. This problem is formulated as an integer linear program and is shown in Eq. (1).

n_i is a binary variable that is assigned to true if the i'th net is unroutable. r_{jq} is a binary variable representing whether the q'th wiring solution for the j'th net is selected. Let C be the set of pairs of conflicting wiring solutions belonging to different nets. The objective is to minimize the number of nets whose wiring solutions are conflicting with wiring solutions of other nets in the chosen routing option for the cell. The first set of constraints guarantees that if two conflicting wiring solutions (for two different nets) are in the selected routing option, then one of the two nets is selected as unroutable. A constraint is generated for every pair of conflicting routing options. The second set of constraints guarantees that

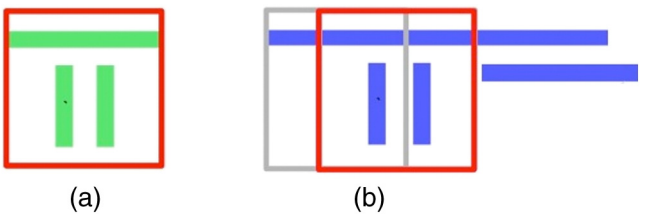

Fig. 7 Checking a routing option for forbidden patterns: (a) a forbidden pattern is shown; (b) we show a snippet of the routing option where the forbidden pattern is matched. The routing option is drawn in blue, while the gray and red boxes are two sliding windows.

for every net, exactly one wiring solution is chosen. Thus, the program has to choose the routing options in a way that minimizes the number of unroutable nets:

$$\text{minimize} \sum_i n_i$$

$$\text{subject to } r_{jq} + r_{kp} - n_j - n_k \leq 1 \quad \forall\, (r_j, r_k) \epsilon C$$

$$\sum_q r_{iq} = 1 \quad \forall\, i. \tag{1}$$

2.2.4 *Sample output*

The sample output of the routing options generator is shown in Fig. 6 where Fig. 6(a) shows the AND2_X1 cell without any of the generated routing options and two of the routing options are shown in Figs. 6(b) and 6(c).

2.3 *Forbidden Patterns Checker*

The generated routing options are checked against the given set of forbidden patterns. A window is slid over the layout with a track granularity and the pattern of required size is formed starting at each row and column combination. The tracks in the pattern are serialized and represented, as shown in Fig. 4, in order to do an easy and fast comparison with the input forbidden patterns. If the routing option contains any of these patterns, then it is discarded. For example, the routing option shown in Fig. 7(b) will be discarded if

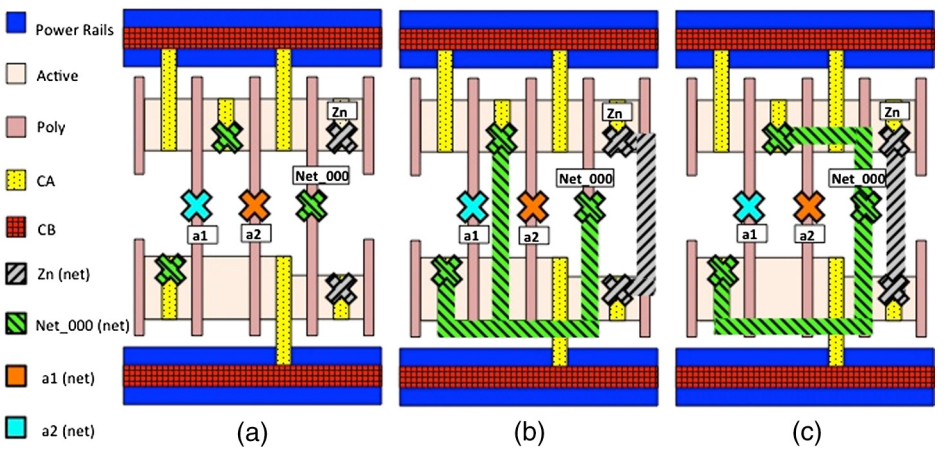

Fig. 6 Sample output of pattern-DRE on AND2_X1 cell: (a) the cell generated by pattern-DRE before the routing options are generated. Cross-markers are placed to show the contact locations to be connected. There are four nets in this cell: a1 (in light blue) is a single-contact net, a2 (in orange) is a single-contact net, Net_000 (in green) has three contacts, and Zn (in gray) has two contacts. Only two routing options are shown [(b) and (c)].

the pattern in Fig. 7(a) is forbidden. It is worth mentioning that pattern-DRE is not sensitive to the existence of forbidden patterns that can be formed across borders of two adjacent cells as a result of placement. However, it can be easily extended to handle these patterns by considering cells in pairs where each pair simulates the side-by-side placement of two cells, assuming the thick power rails can act as a shielding effect between different rows. More complex abutments can also be handled in a similar way at the cost of runtime. (For patterns that are forbidden due to multiple patterning requirements, boundary conditions can be enforced as proposed in Ref. 27 so that no coloring conflicts are formed after placement. However, we tend to avoid tailoring pattern-DRE to any particular technology, keeping it as generic as possible while the input set of forbidden patterns impose the specific requirements of the technology).

2.4 *Routability Metrics*

The output of the framework is the routability metrics. The framework reports the number of routable cells, the total number of routing options, and the distance from a routable library. The first two metrics are indicative of the ease of routing the cell without the forbidden patterns. The reason we need the number of routing options may not be obvious. While two sets of design rules can produce the same number of routable cells, they actually may not have the same routability impact. So if prohibiting a set of patterns drastically affects the number of routing options and only leaves a few options, then this means that the set of forbidden patterns has a high routability impact. As a result of eliminating a large number of routing options, there is a low chance of a post-route fix for other patterns which were not forbidden in the design stage.

For example, if we want to compare the two sets of forbidden patterns: set A and set B, then we run pattern-DRE twice, once for each set. If set A, as a set of forbidden patterns, leads to a lower number of routable cells, then set A has a higher routability impact. If both of them have same number of routable cells but set A leads to a lower number of routing options, then this means that it has a higher routability impact than set B. The same method can be used to study the sensitivity of routability to specific patterns, where set A and set B will only differ by two patterns (one in set A exchanged by the other in set B). Pattern-DRE also reports the distance from a routable library: the smallest-cardinality subset of forbidden patterns that, if allowed, will lead to a routable library. This can be used to give higher priority to certain patterns during process development (i.e., the process can be optimized in order to be friendly to these patterns in order to have a routable library). This metric is calculated from the cells that are unroutable due to the existence of forbidden patterns, not because of conflicts between the nets. To calculate this metric, we keep track of the sets of forbidden patterns that have occurred in routing options in each cell, then the combinations of these sets from different cells are analyzed in order to find the smallest subset of forbidden patterns that, if allowed, will lead to routable cells. In addition, pattern-DRE also reports the nonzero number of times each pattern occurs in the layout. For cells that are unroutable because of conflicts between nets (not because of forbidden patterns), pattern-DRE reports the minimum number of unroutable nets.

Area and routability are interdependent; therefore, when the two scenarios under comparison are different only in the set of forbidden patterns applied, we compare routability at the same cell area in order to have a fair comparison based on routability. So when we are comparing two scenarios (one of them considered baseline scenario), we restrict the iterations of the device-layer generation such that the area of the standard cells would not be larger than the area in the baseline scenario, and if the cell is not routable within this limited number of iterations, it is considered unroutable.

Pattern-DRE does not study the performance impact, if any, due to changed parasitics. However, since the area is preserved/matched between the scenarios under comparison, the performance impact is expected to be small.

3 Experiments

In this section, we first explain how the framework has been validated; then we present a few studies to give examples of how the pattern-DRE framework can be used.

3.1 *Validation*

To validate the framework, several benchmarking comparisons were performed. First, the generated standard cells were compared, in terms of area, to cells of the Nangate open standard cell library,[28] using same design rule values. (Although we know that the Nangate library will have a worse quality than a commercial library, we used it because it is available for research purposes, while commercial sub-45 nm libraries with layouts and design rules are not available to us at this time).

The average error in area between standard cells generated by DRE and those of Nangate was 2%. To validate the routing options generator, the average wirelength of the cell routing options was compared to the wirelength of a rectilinear Steiner minimal tree routing algorithm.[29] On the average, our routing options generator produced a 12% higher wirelength, but was 44× faster. For the pattern occurrences, we found that the patterns that occupied 82.4% of the Metal1 layer in Nangate layouts took up 81.5% of the pattern-DRE Metal1 layer. Also, the cosine similarity between the pattern count vectors from pattern-DRE and Nangate was 0.86. To calculate the cosine similarity, we counted the number of times each pattern occurs in the Nangate cells. (We used only the pattern-DRE routable cells in the validation.) From pattern-DRE, we calculated the average number of pattern occurrences per routing option for each cell and summed that average count for all routable cells. Then we calculated the cosine similarity between the pattern count vectors from Nangate and pattern-DRE.

These validation attempts show that the pattern-DRE estimates are good enough in comparison to actual layouts. In addition, pattern-DRE is very fast in comparison to other evaluation methods involving manual design/tweaks that would require weeks. Pattern-DRE can process 92 cells in 45 h for a maximum of seven front-end layer generation iterations to find a routable solution. Without finding the minimum number of unroutable nets, the 92 cells can be processed in 17 h. If we use only one iteration (and do not regenerate the transistors in a different way if the cell is unroutable), and if we disable the minimum number of unroutable nets calculation, then 92 cells are solved in 40 min with 11.5% less routable cells.

3.2 *Litho-Etch-Litho-Etch Versus Self-Aligned Double Patterning*

In this experiment, we performed a simple comparison between LELE and SADP. It is known that SADP has less susceptibility to overlay error than LELE.[30] To make better use of the overlay advantage of SADP, we assume that the process does not allow the formation of the side of the polygons using trim mask (which is subject to overlay error). Thus, if we assume that the distance between the corner of the stitched polygon and the tip on the same mask exceeds the minimum spacing, then any small odd cycle between two tips and a side like the one shown in Fig. 8 can be resolved in LELE by introducing a stitch, but it cannot be resolved in SADP where stitches are prohibited.

To perform this experiment, we generated a list of 258 forbidden patterns. Each pattern has two columns and two rows and needs a stitch to be resolved. Examples of such patterns are shown in Fig. 9. All these patterns are SADP-incompliant but are LELE-compliant. Thus, we conducted the SADP experiment using those patterns as forbidden ones, but the LELE experiment is done without any forbidden patterns.

The experiment was performed with 22 nm rules and planar CMOS transistors. In the results, we focused on the 78 routable cells with the given design rules (out of the input 92 cells). The baseline scenario for this experiment is the LELE case, which does not have any forbidden patterns. Results are shown in Table 1. The reported columns are as follows:

- Routable cells: number of cells that have one or more routing options.
- Routing options: total number of routing options for all cells.
- Difference in routing options: the difference (as a percentage) between the number of routing options in the current scenario and in the baseline scenario.

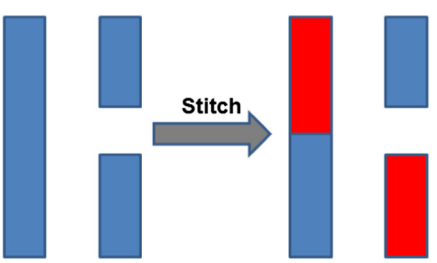

Fig. 8 An odd cycle that can be resolved in litho-etch-litho-etch (LELE) (if distance between the corner to corner after stitch is greater than the spacing rule) by introducing a stitch, but cannot be resolved in self-aligned double patterning (SADP), which does not allow stitches.

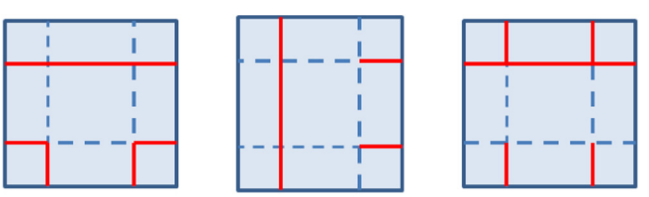

Fig. 9 Three samples of the 258 forbidden patterns used to conduct the LELE versus SADP experiment. Each of these patterns requires a stitch, so these patterns are assumed to be SADP-incompliant but LELE-compliant.

Table 1 Comparison results of self-aligned double patterning (SADP) (i.e., when the two tips and side odd cycles are prohibited) and litho-etch-litho-etch (LELE) (when those odd cycles are not prohibited).

	Routable cells	Routing options	Decrease in routing options	Distance from routable library
SADP	77	2766	17.1%	1
LELE	78	3338	0%	0

In some cases, as explained in Sec. 2, pattern-DRE may attempt a different device-layers design in order to find a routing solution which, in turn, can affect the area. Thus, to have a fair comparison based on routability, we restricted the iterations of pattern-DRE so that the cells generated for the SADP scenario have the same areas as those of the LELE scenario. According to the experiment results and with this selection of forbidden patterns, we sacrifice 1.3% of the routable cells and 17.1% of the routing options for the sake of the overlay advantage of SADP. The minimum distance to a routable library was one forbidden pattern, i.e., if we remove one pattern, 1.3% of the cells that are unroutable will be routable. It is worth mentioning that we present these studies as examples to demonstrate how to use the framework, and the objective is not the actual comparison of these processes. Accordingly, we emphasize that in order to make a proper decision, it is required to enumerate all SADP-incompliant patterns that are compliant to LELE and use them as forbidden patterns, then enumerate all LELE-incompliant patterns that are compliant to SADP (if any), and compare the results of these two scenarios. Instead of exhaustive pattern enumeration, the generation of those patterns can be performed by a Monte Carlo method to produce an enormous number of patterns at random, and LELE and SADP decomposition are to be performed on those patterns. Then patterns that are LELE-compliant and not SADP-compliant are to be used as forbidden patterns for SADP, and vice versa. In our case, we did not have access to a commercial SADP decomposer, so we selected those patterns in the way explained above for demonstration purposes. However, in the next section, we show how we generate the forbidden patterns in a more precise manner.

Note that pattern-DRE, as a framework, is not aware of multiple patterning. However, the applied forbidden patterns impose the restrictions of the patterning scheme that is used. For example, when it is desired to test SADP, the forbidden patterns should be the patterns that are not allowed by SADP like the patterns that require stitches. (Foundries are encouraged to download the framework[31] and try it with their own patterns and rules).

3.3 *Litho-Etch-Litho-Etch Versus Extreme Ultraviolet Lithography*

In this section, we perform a comparison between LELE and extreme ultraviolet lithography (EUVL). In this experiment, we used patterns of size 4×4, i.e., patterns of larger range than the ones used in Secs. 3.2 and 3.4. By using these longer-range patterns, we can detect more decomposition conflicts that cannot be represented in smaller patterns.

Table 2 Comparison results of LELE and extreme ultraviolet lithography (EUVL).

	Routable cells	Routing options	Decrease in routing options	Distance from routable library
LELE	72	1440	56.9%	16
EUVL	78	3338	0%	1

We assume EUVL does not have any forbidden patterns. For LELE, we first ran pattern-DRE without any forbidden patterns to get all the patterns that were used in the routing options. Then we ran a commercial LELE decomposer[32] on those patterns and used the patterns with unsuccessful decomposition as forbidden patterns, and reran pattern-DRE. After applying the forbidden patterns, it is possible to have new patterns that were not generated before due to exploring different front-end options and possible expansion of the bounding box of the nets to find a routing solution. To check that the patterns used in case of LELE are all LELE-compliant, we ran the commercial decomposer again on the final patterns, and none of them had LELE decomposition conflicts. The results in Table 2 show that by using LELE instead of the unconstrained EUV, we sacrifice routability of 7.8% of the cells and 56.9% of the routing options at the same cell area. (Note that the results for LELE in this experiment are different from those in Table 1 because here we use 4×4 patterns that are incompliant to LELE, as explained above.) The minimum distance to a routable library is 16 forbidden patterns, i.e., if we remove these 16 patterns, we reclaim the routability of 7.8% of the cells that are unroutable.

3.4 *Diffusion Location*

In this study, we investigate the effect of the location of the diffusion within the cells, assuming an SADP process. This experiment is different from the previous ones in the sense that we do not compare two different processes, but we use pattern-DRE to compare two front-end choices for the same process (SADP), and we use the same SADP forbidden patterns used in Sec. 3.2. We study two options for the diffusion location: diffusion being as close as possible to the P/N interface versus as close as possible to the power rails. Placing the diffusion close to the P/N interface allows a larger value of line end extension, which is more robust against overlay error but is expected to lead to less routability since the diffusion contacts will be closer to the poly contacts. In addition, changing the location of the diffusion can affect stress, especially in the presence of tensile and compressive nitride liners, which in turn affects performance.[33] Results in Table 3 show that by locating the diffusion close to the P/N interface instead of close to the power rails, we lose 5.1% of the routable cells, and 68.9% of the routing options. Note that in this experiment, the two scenarios under comparison have different device-layer designs, so it is not possible to force the same area, but the areas turned out to be very similar.

3.5 *Area Versus Routability*

In a lot of cases, it is possible to gain more routability by increasing the cell area. In this experiment, we varied the maximum number of chaining iterations (see Sec. 2.1) and checked the number of routable cells (out of 78 total routable cells) as well as the total cell area. We used the same LELE setup used in Sec. 3.3. The result is plotted in Fig. 10. The results are interesting since they show that with a very little increase in area, we can get great routability benefits (21.8% routability improvement).

Table 3 Comparison between routability of cells with diffusion placed close to power rails and close to P/N interface.

Diffusion location	Routable cells	Routing options	Decrease in routing options	Total cell area (um^2)
Close to power rails	78	2772	0	39.7
Close to P/N interface	74	861	68.9	39.6

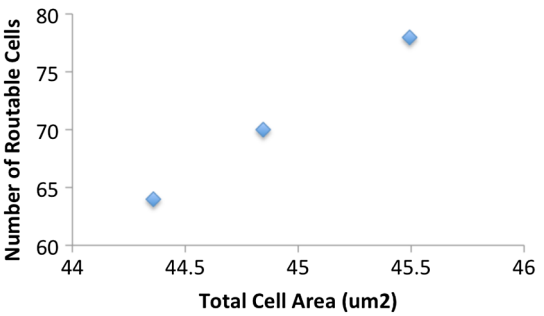

Fig. 10 Total cell area versus number of routable cells.

4 Conclusion

In this paper, we introduced pattern-DRE, a pattern-aware design rule evaluator. Pattern-DRE can be used to optimize pattern-based design rules and identify important patterns that need to be focused on, by patterning technology. It can also be used to compare restrictive patterning technologies. As examples, we used pattern-DRE to evaluate SADP versus LELE, and LELE versus EUVL. It was also used to evaluate the choice of the diffusion location within the cell, being either close to the power rails or close to the P/N interface. Our ongoing work considers a method for pattern-based design rule evaluation for back-end layers.

Acknowledgments

This work was partly supported by IMPACT+ center (http://impact.ee.ucla.edu), SRC, and NSF CAREER award.

References

1. R. S. Ghaida and P. Gupta, "DRE: a framework for early co-evaluation of design rules, technology choices, and layout methodologies," *IEEE Trans. Computer-Aided Design Integr. Circuits Syst.* **31**(9), 1379–1392 (2012).
2. T. Jhaveri et al., "Maximization of layout printability/manufacturability by extreme layout regularity," *Proc. SPIE* **6156**, 615609 (2006).
3. T. Jhaveri et al., "Co-optimization of circuits, layout and lithography for predictive technology scaling beyond gratings," *IEEE Trans. Computer-Aided Design Integr. Circuits Syst.* **29**(4), 509–527 (2010).
4. M. Cho et al., "ELIAD: efficient lithography aware detailed router with compact post-opc printability prediction," in *Proc. 45th Annual Design*

Automation Conference, pp. 504–509, ACM, Anaheim, California (2008).
5. D. Ding et al., "AENEID: a generic lithography-friendly detailed router based on post-RET data learning and hotspot detection," in *Proc. 48th Design Automation Conference*, pp. 795–800, ACM, New York (2011).
6. Y.-T. Yu et al., "Machine-learning-based hotspot detection using topological classification and critical feature extraction," in *Proc. 50th Annual Design Automation Conference*, pp. 67:1–67:6, ACM, Austin, Texas (2013).
7. D. Ding et al., "EPIC: efficient prediction of IC manufacturing hotspots with a unified meta-classification formulation," in *Design Automation Conference (ASP-DAC), 2012 17th Asia and South Pacific*, pp. 263–270, IEEE, Sydney, NSW, Australia (2012).
8. D. Ding et al., "High performance lithographic hotspot detection using hierarchically refined machine learning," in *Proc. 16th Asia and South Pacific Design Automation Conference*, pp. 775–780, IEEE Press, Yokohama, Japan (2011).
9. J.-Y. Wuu, F. G. Pikus, and M. Marek-Sadowska, "Efficient approach to early detection of lithographic hotspots using machine learning systems and pattern matching," *Proc. SPIE* **7974**, 79740U (2011).
10. D. Ding et al., "Machine learning based lithographic hotspot detection with critical-feature extraction and classification," in *IEEE Int. Conf. on IC Design and Technology*, pp. 219–222, IEEE, Austin, Texas (2009).
11. J. Ghan et al., "Clustering and pattern matching for an automatic hotspot classification and detection system," *Proc. SPIE* **7275**, 727516 (2009).
12. A. Kahng, C.-H. Park, and X. Xu, "Fast dual-graph-based hotspot filtering," *IEEE Trans. Computer-Aided Design Integr. Circuits Syst.* **27**(9), 1635–1642 (2008).
13. N. Ma et al., "Automatic hotspot classification using pattern-based clustering," *Proc. SPIE* **6925**, 692505 (2008).
14. V. Dai et al., "DRC plus: augmenting standard DRC with pattern matching on 2D geometries," *Proc. SPIE* **6521**, 65210A (2007).
15. D. Jang et al., "In-design process hotspot repair using pattern matching," *Proc. SPIE* **8327**, 83270S (2012).
16. J. Mitra, P. Yu, and D. Z. Pan, "RADAR: RET-aware detailed routing using fast lithography simulations," in *Design Automation Conference, 2005. Proceedings. 42nd*, pp. 369–372, IEEE, Anaheim, California (2005).
17. V. Dai et al., "Pattern matching for identifying and resolving non-decomposition-friendly designs for double patterning technology (DPT)," *Proc. SPIE* **8684**, 868409 (2013).
18. R. S. Ghaida et al., "A methodology for the early exploration of design rules for multiple-patterning technologies," in *Proc. Int. Conf. on Computer-Aided Design*, Vol. 12, pp. 50–56, ACM, San Jose, California (2012).
19. Y. Deng et al., "DPT restricted design rules for advanced logic applications," *Proc. SPIE* **7973**, 79730H (2011).
20. X. Gao and L. Macchiarulo, "Enhancing double-patterning detailed routing with lazy coloring and within-path conflict avoidance," in *Proc. Conference on Design, Automation and Test in Europe*, pp. 1279–1284, European Design and Automation Association, Dresden, Germany (2010).
21. K. Yuan, K. Lu, and D. Z. Pan, "Double patterning lithography friendly detailed routing with redundant via consideration," in *Design Automation Conference, 2009. DAC'09. 46th ACM/IEEE*, pp. 63–66, IEEE, San Francisco, California (2009).
22. I. S. Abed and A. G. Wassal, "Double-patterning friendly grid-based detailed routing with online conflict resolution," in *Proc. Conference on Design, Automation and Test in Europe*, pp. 1475–1478, EDA Consortium, Dresden, Germany (2012).
23. Q. Ma, H. Zhang, and M. D. Wong, "Triple patterning aware routing and its comparison with double patterning aware routing in 14 nm technology," in *Proc. 49th Annual Design Automation Conference*, pp. 591–596, ACM, San Francisco, California (2012).
24. J.-R. Gao and D. Z. Pan, "Flexible self-aligned double patterning aware detailed routing with prescribed layout planning," in *Proc. 2012 ACM International Symposium on Physical Design*, pp. 25–32, ACM, Monterey, California (2012).
25. C. Kodama et al., "Self-aligned double and quadruple patterning-aware grid routing with hotspots control," in *Design Automation Conference (ASP-DAC), 2013 18th Asia and South Pacific*, pp. 267–272, IEEE, Yokohama, Japan (2013).
26. J. Soukup, "Circuit layout," *Proc. IEEE* **69**(10), 1281–1304 (1981).
27. L. Liebmann, D. Pietromonaco, and M. Graf, "Decomposition-aware standard cell design flows to enable double-patterning technology," *Proc. SPIE* **7974**, 79740K (2011).
28. Si2, "NanGate FreePDK45 Generic Open Cell Library," 2009, http://www.si2.org/openeda.si2.org/projects/nangatelib (19 November 2014).
29. C. Chu and Y.-C. Wong, "Flute: fast lookup table based rectilinear Steiner minimal tree algorithm for VLSI design," *IEEE Trans. Computer-Aided Design Integr. Circuits Syst.* **27**(1), 70–83 (2008).
30. Z. Xiao et al., "A polynomial time exact algorithm for overlay-resistant self-aligned double patterning (SADP) layout decomposition," *IEEE Trans. Computer-Aided Design Integr. Circuits Syst.* **32**(8), 1228–1239 (2013).
31. Y. Badr, K.-W. Ma, and P. Gupta, "Pattern-DRE v1.0," http://nanocad.ee.ucla.edu/Main/DownloadForm (19 November 2014).
32. Double Patterning tool, Calibre 2013.1, Mentor Graphics.
33. V. Joshi et al., "Stress aware layout optimization," in *Proc. of the 2008 Int. Symp. on Physical Design*, pp. 168–174, ACM (2008).

Yasmine Badr is a PhD student in the Electrical Engineering Department at the University of California, Los Angeles (UCLA). She received her BSc and MSc degrees in computer engineering from Cairo University. Before joining UCLA, she was a part-time research engineer at Mentor Graphics, Cairo. Her research interests are in algorithms for design for manufacturing (DFM).

Ko-wei Ma is currently a hardware engineer at NVIDIA Corp. He got his BSc degree from National Taiwan University and his MS degree from UCLA.

Puneet Gupta is a faculty member of the Electrical Engineering Department at UCLA. He received his PhD in 2007 from the University of California, San Diego. He co-founded Blaze DFM Inc. and has authored over 100 papers and 15 patents. He currently leads the multiuniversity IMPACT+ center (http://impact.ee.ucla.edu), and his research interests lie in design-manufacturing interface and the hardware-software interface for variability.