Layout Pattern-Driven Design Rule Evaluation

Yasmine Badr[a], Ko-wei Ma[b] and Puneet Gupta[a]

[a]EE Department, Univ of California, Los Angeles
[b]NVIDIA Inc.
ybadr@ucla.edu, puneet@ee.ucla.edu

ABSTRACT

With the use of sub-wavelength 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 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 post-route 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. In this work, 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. LELE, SADP, SAQP, 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 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. The second study investigates the area penalty as well as the SADP-compliance that we obtain if we increase the minimum gate-to-Local-Interconnect spacing design rule.

Keywords: Design Rules, Manufacturability, Bad Patterns, Hot Spots, Design for Manufacturing, Design Technology Co-optimization (DTCO), Layouts

1. INTRODUCTION

As the semiconductor industry continues to use sub-wavelength photolithography, new printability problems arise. Some layouts can pass Design Rule Check (DRC), 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 return can lead to a tremendous increase in the standard cell area. An alternative, but also 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 that, 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 (LELE and SADP), Triple Patterning, Quadruple Patterning, and beyond. Each of those restrictive patterning technologies has some non-manufacturable 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 indicate from a design perspective, the patterns that the process needs to be optimized for. A high level overview of the framework is shown in Figure 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 output.

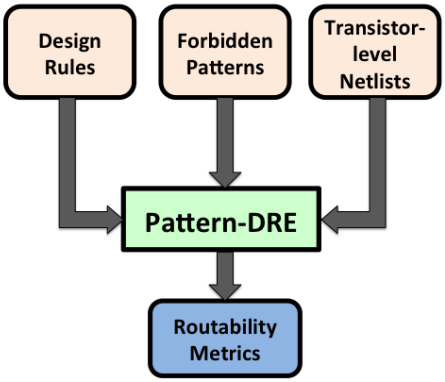


Figure 1: Overview of Pattern-DRE framework

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 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 (a.k.a "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. For example, ref. 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. 2, a lithography-aware router was proposed, which used a printability metric to guide the router. Another approach was developed in ref. 3, which used routing path prediction, along with lithography simulation and a hot spot 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[4–11] used various techniques of Machine Learning or fuzzy pattern matching to identify the hot spots in the design. Ref. 12 and ref. 13 advocated a flow that integrates a pattern checker, a pattern fixer, and a router; such that the router completes its job, and then pattern check and fix are performed if needed, and tentatively some routes are redone. Ref. 14 also used rip-up and re-route to build a router that is RET (Resolution-Enhancement Techniques) - aware. Similar to the post-design hot spot detection, ref. 15 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 apply fixes to them.

To explore design rules for Multiple Patterning technologies, ref. 16 used Machine Learning techniques to predict the number of conflicts, which can be used to compare several sets of design rules. Ref. 17 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, 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,[18–20] Triple Patterning-aware routing,[21] SADP and SAQP-aware routing.[22,23]

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 and breaks down each module used into detail, and it shows how Pattern-DRE can be used to make decisions about forbidden patterns. In section 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 section 4.

2. PATTERN-DRE FLOW

In this section, we explain the flow of the Pattern-DRE framework, which is illustrated in Figure 2. The input to Pattern-DRE is the set of design rules, transistor-level netlists for the standard cells, 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. Routability metrics are reported by the framework at the end. In addition, the count of all occurring patterns are reported at the end. In the following subsections, the details of each block in the flow will be explained.

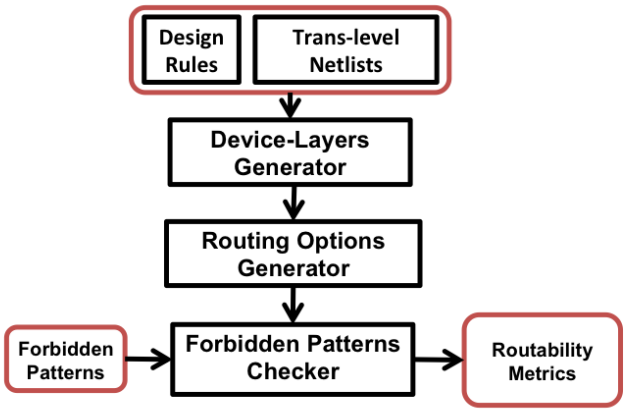


Figure 2: Flow of Pattern-DRE

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 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. The nets along with their contact locations are used as inputs to the next module.

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 of routing with a specific topology, we try to enumerate all possible routing options under a single trunk Steiner tree[24] topology type. The wiring solutions for each net are generated as presented in Ref. 16. Starting with each net, the bounding box is determined according to the contact locations inside that net. If

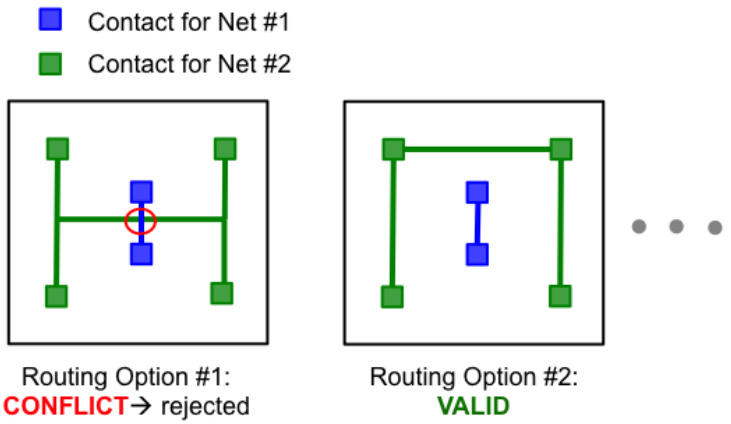


Figure 3: An invalid routing option (on the left) because of a conflict between the routes of the nets and a valid routing option (on the right)

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 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 unnecessarily long wire length, as shown in Ref. 16. 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. 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 on the left in Figure 3 and another example showing a valid routing option is shown on the right. After we discuss the pattern representation that we use, we will illustrate how the check for conflicts, between wiring solutions of nets, is performed.

Layout and Tile/ Pattern Representation

The layout is represented as a 2D 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 (2x2 tracks), while the size of the patterns is specified as an input‡. 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 Figure 4a, where the rows are read first from left to right followed by columns from bottom to top (first segment occupies 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.

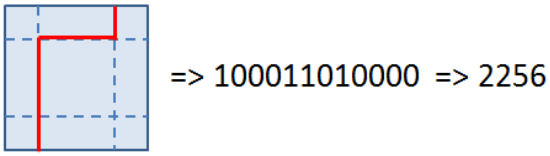
- Node Representation: A node is the intersection of a vertical and a horizontal track. So the node representation encodes whether or not each node is occupied (A node is occupied if any of its neighboring segments

*In our experiments we used a threshold of one track and we expanded the bounding box to three tracks in such a case.

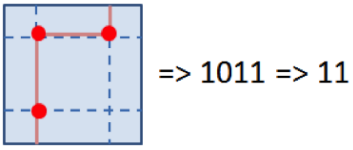
†To avoid confusion, when we mention "wiring solution", we are referring to a way to route the net, but when we say "routing option" we are pointing to one way to route all the nets in the cell (i.e. a set of wiring solutions; one for each net).

‡Currently the maximum allowed pattern size is 5x5 tracks.

is occupied.). An example for the node presentation is shown in Figure 4b. The node representation is required for the conflict detection, which will be explained shortly.



(a) 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) Node representation. The nodes are serialized as a binary string (1011). Then, the decimal equivalent (11) is used as the node representation.

Figure 4: Segment and Node Representations for Tile/Pattern

Conflict Checker

Two wiring solutions for two different nets are conflicting if their segments overlap or cross. These cases can be checked by doing AND operation between the node representations of the routing options in each tile. If the result of the AND operation is non-zero for any tile, then there is a conflict. The reason for doing the conflict check on the node representation is that some conflict cases can not be detected on 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 Figure 5.

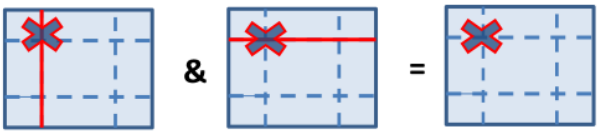


Figure 5: Checking conflicts between routing options of different nets by ANDing Node Representations

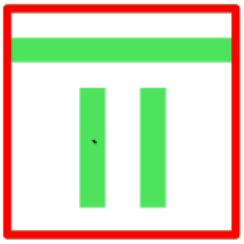
2.3 Forbidden Patterns Checker

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 Figure 4a, 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 Figure 6b will be discarded if the pattern in Figure 6a is forbidden.

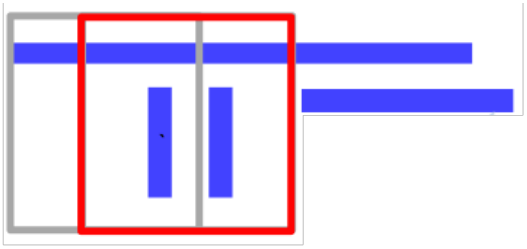
2.4 Routability Metrics

The output of the framework is the routability metrics. The framework reports the number of routable cells and the total number of routing options. Both of these are indicative of the ease of routing the cell without the forbidden patterns. The reason why 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 high routability-impact. As a result of eliminating a large number of routing options, there is low chance of a post-route fix for other patterns, that 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 fewer number of routable cells,



(a) Forbidden Pattern Example



(b) Matching a forbidden pattern in a routing option. The routing option is drawn in blue, while the gray and red boxes are two sliding windows.

Figure 6: Checking a Routing Option for Forbidden Patterns

then set A has a higher routability-impact. If both of them have same number of routable cells but set A leads to fewer number of routing options, then this means that it has higher routability -impact then set B. The same method can be used to study 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).

In addition, Pattern-DRE also reports the non-zero number of times each pattern occurs in the layout.

3. EXPERIMENTS

In this section, we first explain how the framework has been validated, then we present a few studies to present two experiments 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 Nangate open standard cell library,[25] using same design rule values. 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.[26] On the average, our routing options generator produced 12% higher wirelength, but it was 44x faster. For the pattern occurrences, we found that the patterns that occupied 80% of Metal1 layer in Nangate layouts took up 73.4% of the Pattern-DRE Metal1 layer. Also the cosine similarity between the pattern counts vectors from Pattern-DRE and Nangate was 0.82. To calculate the cosine similarity, we counted the number of times each pattern occurs across the tiles in the Nangate cells[§]. 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 counts 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 any other evaluation method involving manual design/tweaks; Pattern-DRE can process 92 cells in 40 minutes.

3.2 Litho-Etch-Litho-Etch (LELE) vs. Self Aligned Double Patterning (SADP)

In this experiment we performed a simple comparison between Litho-Etch-Litho-Etch (LELE) and Self-Aligned Double Patterning (SADP) . It is known that SADP has less susceptibility to overlay error than LELE.[27] 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 Figure 7 can be resolved in LELE by introducing a stitch, but it can not be resolved in SADP where stitches are prohibited.

[§]We only used the Pattern-DRE-routable cells in the validation

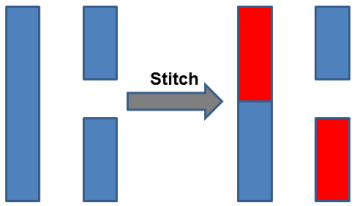


Figure 7: An Odd cycle that can be resolved in LELE (if distance between the corner to corner after stitch is greater than the spacing rule) by introducing a stitch, but can't be resolved in SADP which does not allow stitches.

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 Figure 8. All these patterns are SADP-incompliant but LELE-compliant. Thus we conducted the SADP experiment using those patterns as forbidden ones, but the LELE experiment is done without any forbidden patterns.

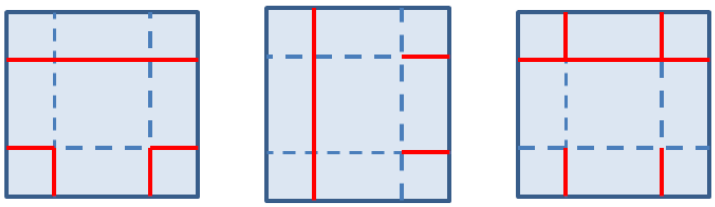


Figure 8: Three samples of the 258 forbidden patterns used to conduct the LELE vs. SADP Experiment. Each of these patterns requires a stitch, so these patterns are assumed to be SADP-incompliant but LELE-compliant.

The experiment was performed with 22nm rules and planar CMOS transistors. In the results, we focused on the 69 routable cells with the given design rules (out of the input 92 cells). The baseline scenario for this experiment is the LELE case which doesn't have any forbidden patterns. Results are shown in Table 1. The reported columns are:

- Routable cells: number of cells which have one or more routing options
- Routing Options: total number of routing options for all cells
- Unique patterns: total number of distinct patterns that appear in the routing options of the cell layouts
- Pattern Instances: total number of instances of patterns that appear in the routing options of the cell layouts
- Unique forbidden patterns: total number of distinct forbidden patterns (the SADP-incompliant patterns in this example) that appear in the routing options of the cell layouts
- Forbidden pattern instances: total number of instances of forbidden patterns (the SADP-incompliant patterns in this example) that appear in the routing options of the cell layouts
- Difference in routing options: the difference (as a percentage) between the number of routing options in the current scenario and in the baseline scenario

The results show that while only one cell was changed from routable to unroutable after forbidding those patterns, the number of routing options decreased by 11.6% (Table 1).

According to the experiment results and with this selection of forbidden patterns, we sacrifice one routable cell but 11.7% of the routing options for the sake of the overlay advantage of SADP. However, we emphasize that

	Routable Cells	Routing Options	Unique Patterns	Pattern Instances	Unique forbidden patterns	Forbidden Pattern Instances	Decrease in Routing Options
SADP	65	1036	372	68614	0	0	-11.68%
LELE	66	1173	404	77174	10	197	0%

Table 1: Comparison results of SADP (i.e. when the 2 tips and a side odd cycles are prohibited) and LELE (when those odd cycles are not prohibited)

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. These patterns strongly depend on the process, and in our experiments, we only used the generated set for demonstration purposes. The generation of those patterns can be performed by generating an enormous number of layout clips and running an LELE and an SADP decomposer to find the problematic patterns that exist in one set and not in the other and feeding those patterns as input to the framework as forbidden patterns.

3.3 Gate-to-Local-Interconnect Spacing with SADP compliance

In this study, we compare two sets of design rules, with different values of gate to Local Interconnect (LI) spacing rule[¶]. The objective of the experiment is to evaluate how much area penalty and SADP-compliance we gain by increasing gate to LI spacing rule. We ran three scenarios, each with different value for that rule. In all scenarios, we assume an SADP process, and thus we use the same forbidden patterns as the ones we used in Section 3.2. Results of this experiment are shown in Table 2. The experiment shows that with 19.7% increase in area, we can have 6% more routable cells, and 50.7% increase in number of routing options.

Gate to LI (nm)	Routable Cells	Routing Options	Unique Patterns	Pattern Instances	Area (um^2)	Change in Routing Options (%)	Change in Area (%)
25	65	1036	372	68614	33	0	0
32	67	1342	369	87322	36.1	29.5%	9.4%
38	69	1559	334	116951	39.5	50.7%	19.7%

Table 2: Comparison results of two sets of design rules with a different minimum gate to contact spacing, for an SADP process

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, identify important patterns which need to be focused on, by patterning technology. It can also be used to compare restrictive patterning technologies. As an example, we used Pattern-DRE to compare SADP and LELE.

In our future work, we want to integrate with a lithography simulator to consider the yield-severity of patterns. In addition, we intend to use a more flexible routing options generator since the single trunk Steiner tree topology is sometimes too restrictive in handling long nets.

ACKNOWLEDGMENTS

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

[¶]We use fixed poly pitch, thus the poly pitch value was also changed accordingly

REFERENCES

[1] Ghaida, R. S. and Gupta, P., "Dre: A framework for early co-evaluation of design rules, technology choices, and layout methodologies," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* **31**(9), 1379–1392 (2012).

[2] Cho, M., Yuan, K., Ban, Y., and Pan, D. Z., "Eliad: Efficient lithography aware detailed router with compact post-opc printability prediction," *Design Automation Conference (DAC)* , 504–509 (2008).

[3] Ding, D., Gao, J.-R., Yuan, K., and Pan, D. Z., "Aeneid: A generic lithography-friendly detailed router based on post-ret data learning and hotspot detection," *Design Automation Conference (DAC)* , 795–800 (2011).

[4] Yu, Y.-T., Lin, G.-H., Jiang, I. H.-R., and Chiang, C., "Machine-learning-based hotspot detection using topological classification and critical feature extraction," *Design Automation Conference (DAC)* (2013).

[5] Ding, D., Yu, B., Ghosh, J., and Pan, D. Z., "EPIC: Efficient prediction of ic manufacturing hotspots with a unified meta-classification formulation," *Asia and South Pacific Design Automation Conference (ASPDAC)* (2012).

[6] Ding, D., Torres, A. J., Pikus, F. G., and Pan, D. Z., "High performance lithographic hotspot detection using hierarchically refined machine learning," *Asia and South Pacific Design Automation Conference (ASPDAC)* (2011).

[7] Wuu, J.-Y., Pikus, F. G., and Marek-Sadowska, M., "Efficient approach to early detection of lithographic hotspots using machine learning systems and pattern matching," *SPIE Advanced Lithography* (2011).

[8] Ding, D., Wu, X., Ghosh, J., and Pan, D., "Machine learning based lithographic hotspot detection with critical-feature extraction and classification," *IEEE International Conference on IC Design and Technology (ICIDT)* (2009).

[9] Ghan, J., Ma, N., Mishra, S., Spanos, C., Poolla, K., Rodriguez, N., and Capodieci, L., "Clustering and pattern matching for an automatic hotspot classification and detection system," *SPIE Advanced Lithography* (2009).

[10] Kahng, A., Park, C.-H., and Xu, X., "Fast Dual-Graph-Based Hotspot Filtering," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* **27**, 1635 –1642 (2008).

[11] Ma, N., Ghan, J., Mishra, S., Spanos, C., Poolla, K., Rodriguez, N., and Capodieci, L., "Automatic hotspot classification using pattern-based clustering," *SPIE Advanced Lithography* (2008).

[12] Dai, V., Yang, J., Rodriguez, N., and Capodieci, L., "DRC Plus: augmenting standard DRC with pattern matching on 2D geometries," *SPIE Advanced Lithography* **6521**, 65210A (2007).

[13] Jang, D., Ha, N., Jeon, J., Kang, J.-H., Paek, S. W., Choi, H., Kim, K. S., Lai, Y.-C., Hurat, P., and Luo, W., "In-design process hotspot repair using pattern matching," *SPIE Advanced Lithography* , 83270S–83270S (2012).

[14] Mitra, J., Yu, P., and Pan, D. Z., "Radar: Ret-aware detailed routing using fast lithography simulations," *Design Automation Conference (DAC)* , 369–372 (2005).

[15] Dai, V., Capodieci, L., et al., "Pattern matching for identifying and resolving non-decomposition-friendly designs for double patterning technology (dpt)," *SPIE Advanced Lithography* , 868409–868409 (2013).

[16] Ghaida, R. S., Sahu, T., Kulkarni, P., and Gupta, P., "A methodology for the early exploration of design rules for multiple-patterning technologies," *International Conference on Computer-Aided Design* , 50–56 (2012).

[17] Deng, Y., Ma, Y., Yoshida, H., Kye, J., Levinson, H. J., Sweis, J., Coskun, T. H., and Kamat, V., "Dpt restricted design rules for advanced logic applications," *SPIE Advanced Lithography* , 79730H–79730H (2011).

[18] Gao, X. and Macchiarulo, L., "Enhancing double-patterning detailed routing with lazy coloring and within-path conflict avoidance," *Design, Automation and Test in Europe (DATE)* , 1279–1284 (2010).

[19] Yuan, K., Lu, K., and Pan, D. Z., "Double patterning lithography friendly detailed routing with redundant via consideration," *Design Automation Conference (DAC)* , 63–66 (2009).

[20] Abed, I. S. and Wassal, A. G., "Double-patterning friendly grid-based detailed routing with online conflict resolution," *Design, Automation and Test in Europe (DATE)* , 1475–1478 (2012).

[21] Ma, Q., Zhang, H., and Wong, M. D., "Triple patterning aware routing and its comparison with double patterning aware routing in 14nm technology," *Design Automation Conference (DAC)* , 591–596 (2012).

[22] Gao, J.-R. and Pan, D. Z., "Flexible self-aligned double patterning aware detailed routing with prescribed layout planning," *International Symposium on Physical Design (ISPD)* , 25–32 (2012).

[23] Kodama, C., Ichikawa, H., Nakayama, K., Kotani, T., Nojima, S., Mimotogi, S., Miyamoto, S., and Takahashi, A., "Self-aligned double and quadruple patterning-aware grid routing with hotspots control.," *Asia and South Pacific Design Automation Conference (ASPDAC)* , 267–272 (2013).

[24] Soukup, J., "Circuit layout," *Proceedings of the IEEE* **69**(10), 1281–1304 (1981).

[25] *Nangate open cell library v1.3. 2009 http://www.si2.org/openeda.si2.org/projects/nangatelib.*

[26] Chu, C. and Wong, Y.-C., "Flute: Fast lookup table based rectilinear steiner minimal tree algorithm for vlsi design," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* **27**(1), 70–83 (2008).

[27] Xiao, Z., Du, Y., Zhang, H., and Wong, M. D., "A polynomial time exact algorithm for overlay-resistant self-aligned double patterning (sadp) layout decomposition," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* **32**(8), 1228–1239 (2013).