A Methodology for the Early Exploration of Design Rules for Multiple-Patterning Technologies

Rani S. Ghaida, Tanaya Sahu, Parag Kulkarni, Puneet Gupta
UCLA, Electrical Engineering Dept.
{rani, tanaya, puneet}@ee.ucla.edu parag@qualcomm.com

***Abstract*—Double/Multiple-patterning (DP/MP) lithography in a multiple litho-etch steps process is a favorable solution for technology scaling to the 20nm node and below. Mask-assignment conflicts represent the biggest challenge for MP and limiting them through design rules is crucial for the adoption of MP technology. In this paper, we offer a methodology for the early evaluation and exploration of layout and MP rules intended for speeding up the rules-development cycle. Using a novel wiring-estimation method, we create layout estimates with fine-grained congestion prediction. MP-conflicts are then predicted using a machine-learning approach. In this work, we demonstrate the use of the method for double-patterning lithography in litho-etch-litho-etch process; the methodology is more general, however, and can be applied for other multiple-patterning technologies including tripe/multiple-patterning with multiple litho-etch steps, self-aligned double patterning (SADP), and directed self-assembly. Results of testing the methodology on standard-cell layouts show an 81% accuracy in DP-conflicts prediction. The methodology was then used to explore DP and layout rules and investigate their effects on DP-compatibility and layout area. The methodology allows for rules optimization; for example, pushing the minimum tip-to-side same-color spacing rule value from 1.7× to 1.5× the minimum side-to-side spacing design rule (i.e., from 110nm down to 90nm) would more than double the number of DP-compatible cells in the library.**

I. Introduction

Double/Multiple-patterning (DP/MP) lithography, where layout patterns are formed in multiple separate exposure and etch steps (i.e., litho-etch-litho-etch process), is one of the most favorable solutions for scaling down technology to the 20nm node and below. For the layout to be MP-compatible, layout features must be assigned to two different masks without violating any design rules. Most importantly, features assigned to the same mask, or colored with same color, must obey the minimum same-color spacing rule[1]. Any violation of the same-color spacing is referred to as a MP conflict and achieving a conflict-free assignment is usually impossible for many layouts, especially dense layouts. In fact, it has been shown that layouts typically contain *native conflicts*, which are patterns that cannot be correctly assigned to the two masks without violating the same-color spacing[2].

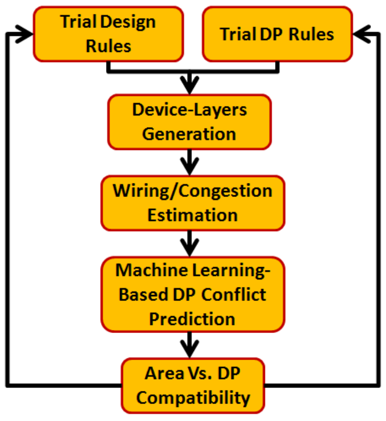

Trial Design Rules
Trial DP Rules
Device-Layers Generation
Wiring/Congestion Estimation
Machine Learning-Based DP Conflict Prediction
Area Vs. DP Compatibility

Figure 1. Overview of our methodology for exploration of DP design rules.

There are two known approaches to get rid of native conflicts. The first approach is to modify the layout so that it is possible to achieve a correct assignment and this has been investigated extensively in literature [1–4]. These works either fail to achieve a conflict-free assignment or successfully remove all the conflicts in small layouts (cell layouts) at the cost of area increase and considerable layout modifications. The second is a correct-by-construction approach. Here, MP rules (i.e., coloring and overlay rules) are accounted for during cell-layout generation and conservative rules are used at the design/cell interface to avoid any possibility of a conflict after placement and routing. Designing with MP rules is believed to be a hassle and conservative rules are expected to have a significant cost in terms of area [5, 6].

An alternative to the known approaches, which has not been investigated yet, is to construct layouts with design rules that would bring MP conflicts down to a manageable number, allowing manual or automated legalization of the layout. For this approach to be examined, a method for studying the effect of rules on MP conflicts as well as layout area is needed.

The work in [7] presents a flow for DP design rules optimization. The method consists of an optimization loop in which rules are modified, the layout is generated, and printability is analyzed. Because actual layout generation and printability analysis are time-consuming, exploring a wide range of rules and rules combinations is impractical with such approach. Moreover, it is susceptible to the specific layout generator used which makes it tough to measure the inherent "DP-friendliness" of the rules.

In this paper, we propose a novel methodology for early

[1]The minimum different-colors spacing rule is equivalent to the minimum spacing rule in the layout and is obeyed during the construction of the layout.
[2]Similar issues exist in other flavors of multiple patterning technologies, such as sidewall image transfer and triple patterning.

evaluation and exploration of DP design rules. The overview of the methodology is depicted in Figure 1. Given trial design rules and DP rules, the first step is to generate the layout of device layers. Next, wiring-layers layout are estimated and congestion is predicted using a novel fine-grained wiring-estimation method. The presence of DP conflicts in the layout is then predicted using a machine-learning approach. In particular, fine-grained estimates of wiring congestion and estimates of layout features (e.g., lineends and L and T-shapes) and their distribution are given to the machine learning (ML) model, a feed-forward back-propagation Artificial Neural Network (ANN).

To the best of our knowledge, this paper is the first to offer a methodology for the exploration of DP rules at early stages of process development. *Although the focus of this paper is on DP, the methodology is more general and can be applied to explore rules of other layout-restrictive technologies, such as triple patterning, self-aligned double patterning, and directed self-assembly.*

We make the following contributions.

- We present the first work on evaluation and exploration of DP rules intended for speeding up the rules-development cycle.
- We propose a novel method for estimating the layout and wiring congestion at a fine-grained level.
- We offer a method that predicts the presence of DP native conflicts based on machine learning, using neural networks, and requires basic layout information like congestion and feature distribution.
- We study the effects of rules and layout styles on DP compatibility of cells and report preliminary results on this topic.

The remaining paper is organized as follows. Section II describes our layout estimation approach that predicts wiring congestion and distribution of layout features. The machine learning-based method for DP conflict prediction in the estimated layouts is presented in Section III. Different rule-exploration studies are performed in Section IV. The paper is concluded with a summary in Section V.

II. Probabilistic Layout and Congestion Estimation

Our approach relies on layout estimation, rather than actual generation, to enable early exploration of a wide range of design rules. We first estimate the device layers of cell layouts to predict contact-points locations and area of front-end layers. For this estimation, the design-rules exploration framework of [8] was used. The framework employs layout-topology generation methods that were shown to be fast and accurate [9], which makes it well suited for our approach. For more details, the reader is referred to [9]. The next step is to estimate the layout for back-end layers used within the cells (i.e., M1 in our experiments).

A. Background

At the design level, many techniques exist in literature to probabilistically estimate congestion without performing actual routing. All the approaches [10–13] effectively *smear* a net across its possible routes and, then, compute congestion for each tile in the design grid based on the probabilistic contributions of each net that can pass through that tile. A minimum spanning-tree (MST) is also used to break multi-pin nets into constituent two-pin net-segments.

Probabilistic congestion estimation techniques have been shown to successfully guide design optimization choices at

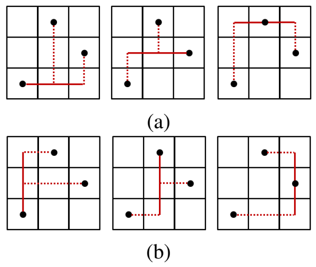


Figure 2. Example of possible wiring solutions with single-trunk Steiner-tree topology for a three-pin net using (a) horizontal trunks and (b) vertical trunks.

various stages of physical design, including placement [10–12] and logic-synthesis [14, 15]. Our novel approach extends and improves upon Steiner heuristic called the single trunk Steiner tree (STST) [16], leveraging it for wiring and congestion estimation at the standard-cell level. Due to its linear computation time (as compared to $O(nlogn)$ or more commonly $O(n^2)$ for MST based approaches) the STST has been previously used for wirelength estimation [17–19], with [17] enhancing it to provide a $O(nlogn)$ approximation, which is on average within 6% of the optimal. We show that our STST based approach allows for fast, yet accurate congestion/wiring estimation as well as prediction of the usage of specific layout shapes and patterns in standard cells. The advantages of using probabilistic STST-based wiring are:

1) The runtime for computing each probabilistic route is $O(N)$, where N is the number of pins on the net. Does not require breaking multi-pin nets into constituent two pen nets, avoiding expensive computations such as MST, RST, RSMT, etc.
2) Each route computation is independent of the other, hence can be easily parallelized if needed.
3) Allows for prediction of local usage of specific patterns and shapes, which is key to predicting multiple-patterning coloring conflicts.

B. Probabilistic wiring-solution generation

Once the front-end layers are created and contact locations are determined, we generate a number of possible wiring solutions for each net (unlike [9], where the wiring of each net is estimated with a single solution). For each cell layout, the generation of wiring solutions undergoes the following steps.

1) Create a grid.
2) Pick a net that is not yet processed.
3) Enumerate possible wiring solutions following a single-trunk Steiner tree topology.
4) Update track utilization for each involved tile.
5) Repeat steps (2) to (4) until all nets are processed.

The grid consists of tiles sized in terms of the number of horizontal and vertical wiring tracks. The tile size is configurable; a large tile size would lead to fast running time but coarse-grained congestion estimates; while a small tile size would lead to fine-grained congestions estimates at the cost of running time. In our experiments, we used a tile size of two vertical tracks and two horizontal tracks[3].

In step (3), we enumerate possible wiring solutions for the net that follow a single-trunk Steiner tree topology. In general, only wiring solutions within the net's bounding box are considered as in the example of Figure 2. When a bounding

[3]If the width (height) of the cell is not an exact multiple of the computed tile-width (tile-height), then the tiles in the last column (row) have smaller horizontal (vertical) track capacities as compared to the rest of the grid.

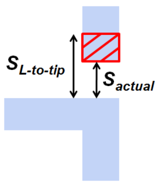


Figure 3. Example of DR violation and associated extra track utilization.

Table I
SPACING RULES IN VERTICAL AND HORIZONTAL DIRECTIONS CONSIDERED IN OUR MODELING OF RULE VIOLATIONS IMPACT ON WIRING CONGESTION.

Tip-to-tip min spacing
Tip-to-side min spacing
L-shape *outer corner* to tip min spacing
L-shape *inner corner* to tip min spacing
L-shape to side min spacing
T-shape *inner corner* to tip min spacing
T-shape *side* to tip min spacing
T-bend *side* to side min spacing
Cross *inner corner* to tip min spacing

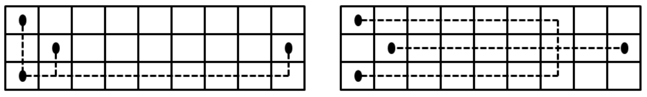
box is too small only few solutions will be generated. Having just a few solutions, a single one in the extreme case, makes those almost fixed rather than potential solutions. To solve this problem, any bounding box with a number of rows/columns falling below a threshold is expanded by a fixed factor (i.e., we allow detours for such nets). For nets having bounding boxes with very skewed aspect ratio, we ignore wiring solutions with common trunks along the shorter direction as such solutions have very poor wirelength (see Figure 4). Each possible solution for a net is assumed to be equally probable (e.g., the probability of each wiring solution in the example of Figure 2 is $\frac{1}{6}$).

After the possible wiring solutions for the net are generated, we determine in step (4) the track utilization in the horizontal and vertical directions for all tiles in the bounding box. Here, track utilization is the occupied track length multiplied by the probability of occurrence. Consider the center tile in the example of Figure 2. The occupied track length in the horizontal direction is roughly 2.5 times the tile-width and the probability of occurrence for each occupied track segment is $\frac{1}{6}$. The track utilization in the vertical direction is calculated similarly and has the same value in this example.

C. Feature distribution and design-rule violations

Once all nets have been processed (by repeating steps (2) to (4) for all nets), we determine the distribution of the different features including tips, line segments, and L and T-shapes. The probability of a feature occurrence at a particular location is the probability of its corresponding wiring solution.

Given the distribution of features, design-rules violations are identified and their effect is modeled as extra track utilization. To illustrate, consider the example design-rule violation shown in Figure 3. Here, the minimum L-shape-to-tip spacing rule, $S_{L-to-tip}$, is violated. To account for this violation, the utilization is updated to include the amount with which the rule is violated (i.e., the shaded region in Figure 3) multiplied by the probability of pattern occurrence (i.e., the two shapes occurring simultaneously). The spacing rules that are considered for violation checking in our method are given in Table I.



(a): Wire length of 12 units. (b): Wire length of 21 units.

Figure 4. Example of an unbalanced net and two of its possible solutions: (a) a typical horizontal trunk-based solution and (b) a pessimistic vertical trunk-based solution.

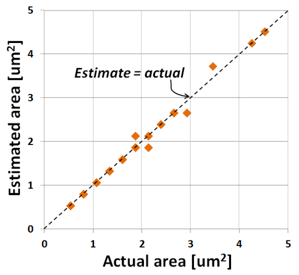
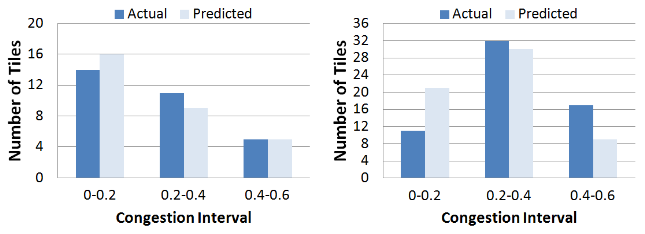


Figure 5. Comparison of the estimated cell area of our approach with that of actual cell-layouts using the same design rules.



(a): NOR3_X4 cell. (b): FA_X1 cell.

Figure 6. Comparison between actual and predicted congestion.

D. Modeling of congestion and its impact on area

Congestion is the ratio of occupied to available track-length and is reported for each tile and in the horizontal and vertical directions separately. The occupied track length is modeled as the sum of the utilization and track length blocked by wiring in the orthogonal direction. And, the available track length is inferred directly from the tile size.

An over-congested tile (congestion greater than 1) indicates a problematic region for completing the wiring successfully. Such regions are very likely to necessitate a layout-area increase. For cells with over-congested tiles, the total congestion overflow is calculated. The extra track-length requirement is fulfilled by increasing the cell area. Since the cell height is fixed, the cell width is increased with the minimum cell unit-widths to meet the requirement. Lastly, congestion in over-congested tiles is updated to the value 1 and the total congestion overflow is distributed equally among the newly added tiles.

E. Method validation

The accuracy of our layout estimation approach is verified with comparison with real cell-layouts. The layout estimation method is run for the entire Nangate 45nm Open Cell Library [20] (110 cells) using the same design rules (i.e., FreePDK 45nm process [21]). A comparison of the estimated and actual cell areas, depicted in Figure 5, show an absolute error of less than 2% on average and a runtime of 38 minutes in real time (on a single CPU)[4]. A comparison of the number

[4]Though the grid-based congestion estimation is fairly accurate, we use the approach in [8] for estimating the actual impact of design rules on area since it gave better accuracy for area estimation (1% vs 2% for Nangate).

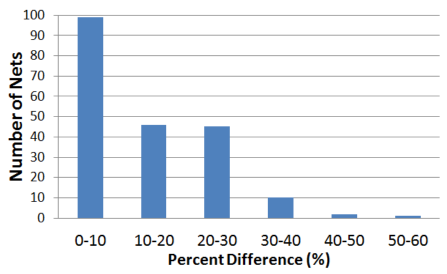


Figure 7. Percent difference between our wirelength estimates and that of FLUTE.

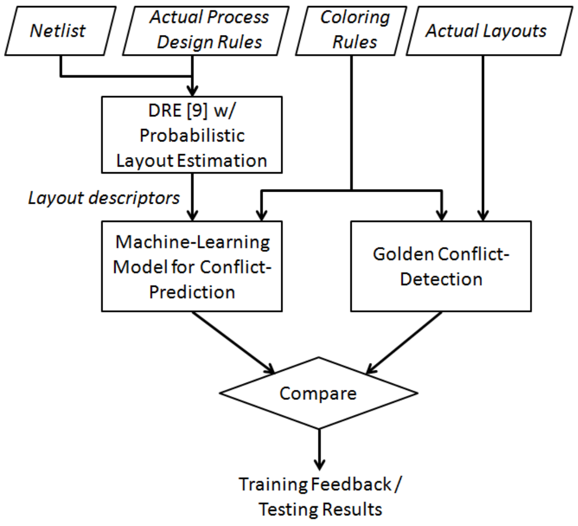

Netlist
Actual Process Design Rules
Coloring Rules
Actual Layouts
DRE [9] w/ Probabilistic Layout Estimation
Layout descriptors
Machine-Learning Model for Conflict-Prediction
Golden Conflict-Detection
Compare
Training Feedback / Testing Results

Figure 8. Overview of our machine learning based approach for conflict prediction.

of tiles in each congestion-interval for two standard cells[5] is reported in Figure 6. Our approach does a good job of identifying areas with high congestion, more specifically the number of tiles with high congestion, which matter the most for our DP conflict prediction.

Our method was also compared to FLUTE [22], a rectilinear Steiner minimal tree routing algorithm. Our wirelength estimates, obtained by averaging across all generated solutions, and runtime are compared with that of FLUTE for nets in the Nangate library [20]. Figure 7 summarizes the results. Our approach leads to 11.94% higher wirelength on average and has $44X$ faster runtime compared with FLUTE.

III. DP Conflict Prediction Using Machine Learning

A machine-learning approach is used to predict DP conflicts in cell layouts as well as at cell boundaries in the design. Artificial neural networks (ANN) have been shown to work well for prediction problems [23]. As a result, we develop an ANN-based classifier, more specifically a feed-forward back-propagation multi-layer perceptron, and use it to predict whether a layout, with a given set of design rules, will have a DP native-conflict (essentially requiring layout redesign).

A. Overview

An overview of our machine learning-based approach for predicting the presence of DP native conflicts in the layout is

[5]With 0.2 intervals, i.e., with the same form of input to the machine learning-based conflict predictor.

Table II
Layout descriptors investigated to build the machine-learning model.

Descriptor	Details
Tile horizontal congestion	# of tiles at each congestion-interval
Tile vertical congestion	# of tiles at each congestion-interval
Tile overall congestion	# of tiles at each congestion-interval
Tile wiring property	# of tiles with 1D, 2D, and no wiring
Layout congestion	vertical, horizontal, and overall congestions
Tile tips	# of tiles at each tip-interval
L / T-shapes	# of tiles at each L/T-shape-interval
Highly packed tiles	# of tiles w/ high congestion and # of tips
Min same-color spacing	Side-to-side, tip-to-side, & tip-to-tip rules

depicted in Figure 8. The machine-learning model is calibrated (as well as tested) using the estimated layouts and congestion maps obtained from the method described in Section II. Characteristics of the estimated layout – as well as DP coloring rules – are given to the model as inputs and are referred to as layout descriptors. These descriptors are carefully chosen so as to correlate well with the existence of DP conflicts.

The model's target data, which is the real-known data that the prediction is compared with in the training and testing of the model, is the actual presence/absence of DP conflicts in *real layouts*. The identification of DP conflicts in real layouts was performed using the golden method of odd-cycle detection in the conflict graph. Specifically, color violations were detected using Calibre SVRF [24] and conflict-graph construction and a depth-first search algorithm were implemented in C++ with OpenAccess database to detect odd cycles *in the real layouts*. The Neural Network Toolbox in MATLAB was used in the development of the ANN model [25].

DP conflicts may occur within cell-layouts but also at the interface between cells post-placement. As a result, for the training and testing sets of our model, we use layouts of standard cells as well as layouts of the interface between different cells, which are taken from post-placement benchmark designs and are referred to as *cell-boundary layout snippets*.

The training of the model was performed using cell layouts from Nangate 45nm Open Cell Library [20] and cell-boundary snippets from benchmark designs synthesized using the same library. The testing of the model was performed on a different set of layouts for cells of the same library and cell-boundary layout snippets from different benchmark designs.

B. Input layout descriptors

Layout descriptors resulting from our layout/congestion estimation method include forms of the following layout characteristics.

- Horizontal wiring congestion.
- Vertical wiring congestion.
- Number and location of tips.
- Number and location of L and T-shapes.

An exhaustive set of layout descriptors, summarized in Table II and inferred from the above characteristics, is investigated and training is repeated with various combinations of descriptors. The descriptors set was pruned depending on the trend of training accuracy.

The descriptors are fused together into a single input vector for each training/testing layout. Based on training accuracy it was observed that horizontal congestion, vertical congestion and tips had good influence on the prediction accuracy. L and T-shapes were excluded from the feature set because they did

Table III
SUMMARY OF THE PROPERTIES OF THE ANN MODEL

Number of layers	3
Number of neurons	50
Number of inputs	19
Supported min same-color spacing rules	S2S, T2S, T2T
Training-range of min same-color spacing rules	80-150nm
Training error	8.7%

not affect the training accuracy[6].

C. Training the classifier

Efficient training of an ANN depends on various factors such as the nature of the data set and the topology of the network. The basic ANN structure has three types of layers: input, hidden and output. Using multiple hidden layers, or even a single hidden layer, with too many neurons results in an increased risk of convergence to a local minimum (over-fitting) and, hence, poor performance on unseen samples. The number of neurons was chosen, by first starting with a large number and gradually reducing to a point where best possible results were obtained.

The model is cross-validated during training so that it is generalized across variations in the training patterns and over-fitting is avoided. The Levenberg-Marquardt algorithm [26, 27] has been shown to be most efficient in attaining a good mean squared error [28] and, as a result, was used for training our model.

For robust training, we ensured that all the input features have nearly same order of magnitude to avoid input features with larger magnitudes from dominating the learning process. For the model to perform well on unseen data and to avoid any learning bias, the selection of cells and cell-boundary snippets layouts for use in the training was made at random. And, for the same purpose, almost the same number of positive (layouts with conflicts) and negative (layouts without conflicts) samples were used in the training. It is important to note that, if the frequencies of layouts with/without conflicts differ widely in a certain technology, either the number of samples of each type (i.e., positive/negative) can be modified or samples can weighted according to the relative frequencies to improve the model's accuracy.

The tile size also has an impact on DP conflicts. A small tile size implicitly ignores long-range conflicts; whereas, a large tile size will have inaccuracy due to ignoring the inherent locality of conflicts. Different tile sizes were examined and a tile size of two tracks in each direction showed best results.

The properties of the model are summarized in Table III[7]. In practice, the minimum side-to-side (S2S), tip-to-side (T2S), and tip-to-tip (T2T) same-color spacing rules may have unequal values. As a result, the model was designed to support different values for these rules, which are inputs to the model.

D. Testing and approach validation

To obtain the ANN outputs in binary form[8], a thresholding function is applied. The discrimination threshold was varied to obtain the Receiver Operating Characteristic (ROC) curve for the model. Figure 9 shows a comparison between ROC-curves of our model, random guess, and a multivariate linear

[6]One explanation for this is that including both horizontal and vertical congestion already captures the same information given by L and T-shapes.

[7]We use hyperbolic tangent-sigmoid for the hidden-layer activation function and a linear function for the output-layer activation function.

[8]Originally, outputs are decimal numbers because a linear function is used for the output activation function.

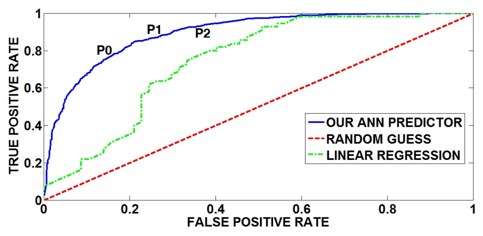


Figure 9. ROC curve for our ANN model compared with those of linear regression and random guess.

Table IV
RESULTS OF TESTING THE ANN MODEL ON CELL LAYOUTS.

True positives	1270
False negatives	271
Positives detection rate	82.5%
False positives	283
True negatives	1138
Negatives detection rate	80%

regression-based model. The area under the ROC curve is a measure of the accuracy of the classifier. It can be clearly seen that our classifier performs significantly better than random guess and the regression-based classifier.

The ROC curve manifest the trade-off between true and false positive rates. If the discrimination threshold is chosen so as to operate in P2 region, then the ANN model over-predicts DP conflicts; if the model operates in P0 region, DP conflicts will be under-predicted. A reasonable trade-off was chosen for the testing, more specifically 82.5% true and 20% false positive rates (i.e., region P1).

The testing results of the ANN model are shown in Table IV. The model achieves a positives-detection rate of 82.5% and a negatives-detection rate of 80%.

IV. DP DESIGN-RULE EXPLORATION

In this section, we use our methodology to explore design rules, especially DP rules, and study their effects on DP compatibility of layouts. It is important to note that the results presented in this section show the strength of the methodology but are not necessarily generalizable. *Design rules, layouts, and DP-conflicts have complex interaction, therefore, the results will strongly depend on the precise rule values, layout styles and the library architecture.*

DP-compatibility of the cell library does not guarantee DP-compatibility of the design after cell placement. To study the general compatibility of layouts with DP, the compatibility metric should involve some notion of DP conflicts at cell-boundaries in addition to DP conflicts in standard-cell layouts. Furthermore, because DP conflicts in cells are typically harder to fix than conflicts at cell-to-cell interfaces (which may be fixed in many cases by cell-placement perturbation), the two types of conflicts should be weighted when forming the metric for the overall DP-compatibility of the design. Hence, we use the following metric:

$$Design\ Conflicts = \alpha \times CC + (1 - \alpha) \times CB, \quad (1)$$

where α is the weighting factor, CC is the fraction of conflicting cells in the library, and CB is the fraction of conflicting cell-boundaries in the benchmark design. For our experiments, we use a weighting factor α of 0.5. Nevertheless, the weighting factor can be adjusted in accordance with the relative importance of conflicts within cells and conflicts across cell boundaries.

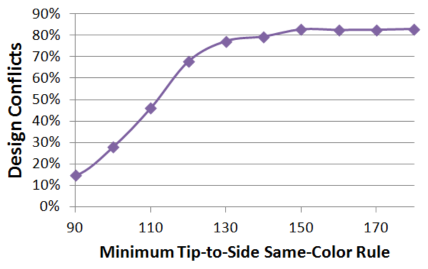


Figure 10. *Design Conflicts* for range of tip-to-side same-color spacing rule.

Table V
BENCHMARK DESIGNS USED IN OUR EXPERIMENTS AND THEIR CORRESPONDING NUMBER OF CELL INSTANCES AND UNIQUE CELL TYPES.

Circuit	Description	Cell instances	Cell types
mips	processor core	19868	73
tv80	processor core	6429	72
or1200	processor core	3077	55
des	cryptography core	1475	28

A. Testing setup

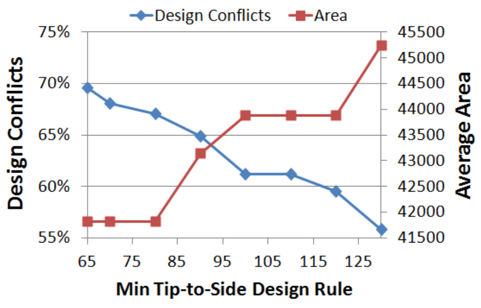
Our design-rule exploration experiments are performed on the M1 layer in Nangate 45nm standard-cell library [20] with FreePDK 45nm design rules [21] (65nm for the minimum spacing in M1). Although we have used a 45nm setup, we correspondingly scaled the minimum same-color spacing rules for DP to make the experiments realistic.

For our baseline experiment, we use a cell-height of 10 M2-tracks, 1D polysilicon (poly), and local interconnect (LI) to perform gate-to-gate connections[9]. Our methodology, layout/congestion estimation followed by the machine-learning model, is run to predict the presence of DP conflicts in M1 layouts. For evaluating design area, we use four benchmark designs from [29] synthesized using Nangate 45nm Open Cell Library [20] and the area evaluator of [8]. Descriptions, cell-instance counts, and number of cell-types for all four design are given in Table V. Prediction of DP conflicts in cell-boundary layout snippets were performed for the same benchmark designs. Since our experiments are performed on the M1 layer and Nangate library has wide horizontal M1 power rails (enough to prevent conflicts between features across the rails), post-placement DP conflicts spanning multiple cells may only occur at the vertical interface between cells. In our experiments, we limit the analysis to cell-boundary snippets at vertical interfaces between cells and we use a snippet size of *Cell Height* $\times$ *8 M1 Pitches*, i.e., four M1 tracks from each side of the interface. For performing experiments on different layers or different types of power rails, however, the layout snippets will need to include snippets at the horizontal interface between cells as well.

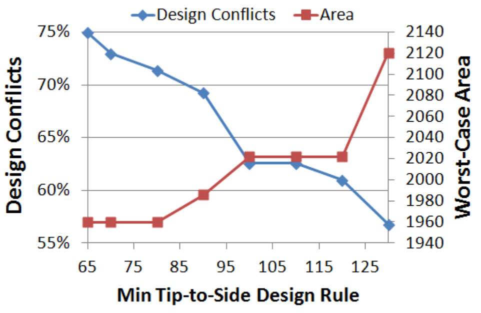
B. DP design-rule exploration

Three spacing rules may have a significant impact on DP-compatibility: side-to-side, tip-to-side, and tip-to-tip minimum spacing rules. With DP, each of these rules has two versions: minimum same-color spacing and minimum design rule in the layout (equivalent to minimum different-colors spacing). Among all three rules, tip-to-side spacing is of particular interest for exploration; tip-to-tip spacing hardly occurs in M1 layouts (for logic) and side-to-side spacing is usually a pre-defined target for the technology.

[9]Although LI is not needed at 45nm, we investigate its effect on reducing M1 complexity and, consequently, DP-compatibility.



(a): Average area across all designs



(b): worst-case design area

Figure 11. *Design Conflicts* and area of benchmark designs for range of tip-to-side design rule.

Our first experiment consists of investigating the minimum tip-to-side same-color spacing rule[10]. In Figure 10, *Design Conflicts* is plotted as a function of the rule value. As expected, we see an increase in conflicts as we increase the same-color rule value. The plot identifies a range of rule values with high impact on DP conflicts, namely values from 90 to 120nm, and another range with small impact on DP conflicts, namely values from 120nm to 180nm. Such information can be used to guide efforts on the process-development side to enhance the DP-compatibility of layouts. For example, pushing the rule value from 1.7$\times$ to 1.5$\times$ the minimum side-to-side spacing design rule (i.e., from 110nm down to 90nm) would more than double *Design Conflicts*. It is worth noting that the plot shows imperfect monotonicity. This limited noise is due to the fact that the machine-learning model does not make perfectly accurate predictions and few incorrect predictions may cause such noise when the total number of cells in the library is limited (110 cells for Nangate).

In another experiment, we study the conflict/area tradeoff with changing the minimum tip-to-side design rule. For a same-color spacing fixed at 130nm (2$\times$ min spacing for all three same-color rules) and for each rule-value, *Design Conflicts* and the benchmark-designs area are depicted in Figures 11. Interestingly, a non-linear trend is observed in both cases for conflicts as well as design areas, which reveals optimization opportunities. For example, increasing the tip-to-side rule from the original value of 65nm to 80nm can reduce *Design Conflicts* by 3% with almost no area increase. Furthermore, 9% reduction in *Design Conflicts* is observed when the tip-to-side design rule is increased to 100nm but the area overhead in this case is 4.7% on average.

The proposed methodology is also applied to study the effects of other design rules (layout styles) on DP-compatibility.

[10]This rule affects the coloring of tip-to-side patterns and should not be confused with the minimum tip-to-side design rule, which defines the spacing of tip-to-side patterns in the layout.

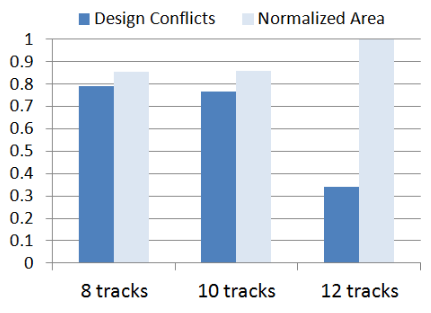


Figure 12. *Design Conflicts* and design area for 8-track, 10-track, and 12-track cell-heights.

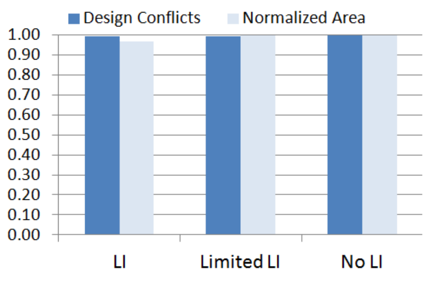


Figure 13. *Design Conflicts* and design area by using: local interconnects (LI) to perform poly-to-poly connections, LI to perform connections of only neighboring poly gates, and no LI.

In one experiment, we vary the cell-height while keeping all other design rules fixed and using a same-color spacing of 130nm. Three cell-heights were investigated: 8, 10 (baseline), and 12 M2 tracks. Results, depicted in Figure 12, show that a 10-track cell-height for Nangate library seems to be a good compromise between area and DP-compatibility.

A study of different local interconnect (LI) schemes is also conducted using our methodology. Assuming polysilicon is one-dimensional, three cases were investigated: LI replaces M1 to perform gate-to-gate connections (if possible), LI replaces M1 to perform short connections of neighboring gates only (i.e., "Limited LI"), and "no LI" where all gate-to-gate connections are performed on M1. The results depicted in Figure 13 show that DP-conflicts as well as area are insensitive to the LI scheme. An important conclusion is that adding a horizontal LI layer, to wire poly and relax the M1 layer, does not bring any noticeable benefits with this particular process (i.e., FreePDK 45nm process [21]).

V. Conclusions

We presented the first work on early evaluation and exploration of multiple patterning rules intended for speeding up the rules-development cycle. The proposed methodology consists of a novel layout/congestion estimation method and a machine-learning based DP-conflict predictor. The methodology was used to explore DP and layout rules and investigate their effects on DP-compatibility and layout area. *Although the focus of this paper is on DP, the methodology is more general and can be applied to explore rules of other layout-restrictive technologies, such as multiple patterning, self-aligned double patterning, and directed self-assembly*. Essentially, all that is needed is a layout conflict checker to train the machine learning model. Our ongoing work also explores using the same methodology to assess the impact of "blocking" usage of certain layout patterns.

References

[1] C. Hsu, Y. Chang, and S. R. Nassif, "Simultaneous layout migration and decomposition for double patterning technology," in *IEEE Intl. Conf. on Computer-Aided Design*, 2009, pp. 595–600.

[2] S.-Y. Chen and Y.-W. Chang, "Native-conflict-aware wire perturbation for double patterning technology," in *IEEE Intl. Conf. on Computer-Aided Design*, 2009, pp. 556–561.

[3] K. Yuan and D. Z. Pan, "WISDOM: wire spreading enhanced decomposition of masks in double patterning lithography," in *IEEE Intl. Conf. on Computer-Aided Design*, 2009, pp. 32–38.

[4] R. S. Ghaida, K. B. Agarwal, S. R. Nassif, X. Yuan, L. W. Liebmann, and P. Gupta, "A framework for double patterning-enabled design," in *Intl. Conf. on Computer-Aided Design*, Nov. 2011, pp. 14–20.

[5] L. Liebmann, D. Pietromonaco, and M. Graf, "Decomposition-aware standard cell design flows to enable double-patterning technology," in *SPIE*, vol. 7974, 2011, p. 79740K.

[6] Y. Ma *et al.*, "Double patterning compliant logic design," in *SPIE*, vol. 7974, 2011, p. 79740D.

[7] Y. Deng *et al.*, "Dpt restricted design rules for advanced logic applications," in *SPIE*, vol. 7973, 2011, p. 79730H.

[8] UCLA_DRE v1.4. [Online]. Available: http://nanocad.ee.ucla.edu/Main/DownloadForm

[9] R. S. Ghaida and P. Gupta, "A framework for early and systematic evaluation of design rules," in *Intl. Conf. on Computer-Aided Design*, Nov 2009, pp. 615–622.

[10] J. Lou, S. Thakur, S. Krishnamoorthy, and H. Sheng, "Estimating routing congestion using probabilistic analysis," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 21, no. 1, pp. 32 –41, jan 2002.

[11] J. Westra, C. Bartels, and P. Groeneveld, "Probabilistic congestion prediction," in *Proceedings of the 2004 international symposium on Physical design*, ser. ISPD '04. New York, NY, USA: ACM, 2004, pp. 204–209. [Online]. Available: http://doi.acm.org/10.1145/981066.981110

[12] Z. Li, C. Alpert, S. Quay, S. Sapatnekar, and W. Shi, "Probabilistic congestion prediction with partial blockages," in *Quality Electronic Design, 2007. ISQED '07. 8th International Symposium on*, march 2007, pp. 841 –846.

[13] A. B. Kahng, "Accurate pseudo-constructive wirelength and congestion estimation," in *Proc. ACM Int. Workshop on System-Level Interconnect Prediction*, 2003, pp. 61–68.

[14] Q. Liu and M. Marek-Sadowska, "Pre-layout wire length and congestion estimation," in *Design Automation Conference, 2004. Proceedings. 41st*, july 2004, pp. 582 –587.

[15] R. S. Shelar, S. S. Sapatnekar, P. Saxena, and X. Wang, "A predictive distributed congestion metric with application to technology mapping," *IEEE Trans. CAD*, vol. 24, pp. 696–710, 2005.

[16] J. Soukup, "Circuit layout," *Proceedings of the IEEE*, vol. 69, no. 10, pp. 1281 – 1304, oct. 1981.

[17] H. Chen, C. Qiao, F. Zhou, and C.-K. Cheng, "Refined single trunk tree: A rectilinear steiner tree generator for interconnect prediction," in *INTL. WORKSHOP ON SYSTEM-LEVEL INTERCONNECT PREDICTION (SLIP)*, 1992, pp. 85–89.

[18] A. Chao, E. Nequist, and T. Vuong, "Direct solution of performance constraints during placement," in *Custom Integrated Circuits Conference, 1990., Proceedings of the IEEE 1990*, may 1990, pp. 27.2/1 –27.2/4.

[19] A. Srinivasan, K. Chaudhary, and E. Kuh, "Ritual: a performance driven placement algorithm for small cell ics," in *Computer-Aided Design, 1991. ICCAD-91. Digest of Technical Papers., 1991 IEEE International Conference on*, nov 1991, pp. 48 –51.

[20] Nangate open cell library v1.3. 2009. [Online]. Available: http://www.si2.org/openeda.si2.org/projects/nangatelib

[21] FreePDK. [Online]. Available: http://www.eda.ncsu.edu/wiki/FreePDK

[22] C. Chu and Y.-C. Wong, "FLUTE: Fast lookup table based rectilinear steiner minimal tree algorithm for vlsi design," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 27, no. 1, pp. 70 –83, jan. 2008.

[23] C. Mair *et al.*, "An investigation of machine learning based prediction systems," *Journal of Systems and Software*, vol. 53, no. 1, pp. 23–29, 2000.

[24] Calibre standard verification rule format manual v2009. [Online]. Available: http://www.mentor.com/

[25] "MATLAB v. 7.13.0," The MathWorks Inc., 2011.

[26] K. Levenberg, "A method for the solution of certain non-linear problems in least squares," *Quarterly Journal of Applied Mathmatics*, vol. 2, no. 2, pp. 164–168, 1944.

[27] D. W. Marquardt, "An algorithm for least-squares estimation of nonlinear parameters," *SIAM Journal on Applied Mathematics*, vol. 11, no. 2, pp. 431–441, 1963.

[28] R. Zayani, R. Bouallegue, and D. Roviras, "Levenberg-marquardt learning neural network for adaptive predistortion for time-varying HPA with memory in OFDM systems," in *Proc. of 16th European Signal Processing Conference*, 2008.

[29] [Online]. Available: http://www.opencores.org/