Routability Optimization for Industrial Designs at Sub-14nm Process Nodes Using Machine Learning

Wei-Ting J. Chan[2], Pei-Hsin Ho[3], Andrew B. Kahng[1,2] and Prashant Saxena[3]
[1]CSE and [2]ECE Departments, UC San Diego, La Jolla, CA 92093
[3]Synopsys Inc., Hillsboro, OR 97124
{wechan, abk}@ucsd.edu, {pei-hsin.ho, prashant.saxena}@synopsys.com

ABSTRACT

Design rule check (DRC) violations after detailed routing prevent a design from being taped out. To solve this problem, state-of-the-art commercial EDA tools global-route the design to produce a global-route congestion map; this map is used by the placer to optimize the placement of the design to reduce detailed-route DRC violations. However, in sub-14nm processes and beyond, DRCs arising from multiple patterning and pin-access constraints drastically weaken the correlation between global-route congestion and detailed-route DRC violations. Hence, the placer—based on the global-route congestion map—may leave too many detailed-route DRC violations to be fixed manually by designers. In this paper, we present a method that employs (1) machine-learning techniques to effectively predict detailed-route DRC violations after global routing and (2) detailed placement techniques to effectively reduce detailed-route DRC violations. We demonstrate on several layouts of a sub-14nm industrial design that this method predicts the locations of 74% of the detailed-route DRCs (with false positive prediction rate below 0.2%) and automatically reduces the number of detailed-route DRC violations by up to 5×. Whereas previous works on machine learning for routability [30] [4] have focused on routability prediction at the floorplanning and placement stages, ours is the first paper that not only predicts the actual locations of detailed-route DRC violations but furthermore optimizes the design to significantly reduce such violations.

1. INTRODUCTION

As semiconductor technology advances, the EDA and design communities have seen increasing unpredictability in the IC implementation flow. In particular, design tapeouts are increasingly placed at risk by the inability of the router to complete the routing successfully. Historically, any risk of unroutability could be identified prior to the runtime-intensive detailed route (DR) stage, based on congestion maps generated using global routing (GR). However, the miscorrelation between these congestion maps and the actual routing design rule check (DRC) violation maps has increased significantly at current process nodes due to the many new, complicated design rules defined at these nodes. This unpredictability causes added iterations (and consequent schedule slippage) during design implementation, sometimes endangering the design tapeout itself.

*This work was performed while W.-T. J. Chan was with Synopsys, Inc., Hillsboro, OR.

ISPD '17, March 19–22, 2017, Portland, OR, USA.

 ISBN 978-1-4503-4696-2/17/03…$15.00
DOI: http://dx.doi.org/10.1145/3036669.3036681

1.1 Motivation

At advanced process nodes, GR-based congestion maps do not correlate well with DRC violation maps obtained at the end of detailed routing. This is a consequence of the numerous complicated design rules imposed upon design layouts to ensure viable fabrication; these DRCs, most of which are not visible in the GR routing model, constrain the detailed router significantly. (The study of Han et al. [6] quantifies wirelength overheads due to increasing numbers of design rules.) As a result, GR-based congestion maps are no longer good predictors either for evaluating overall design routability or for identifying potential DRC violation hotspots prior to detailed routing. Therefore, they can easily mislead any routability optimization engines that rely on these maps, resulting in poor effectiveness at resolving routability problems, even as they sacrifice timing and area metrics to ameliorate spurious congestion problems. Figure 1 shows an example of such a miscorrelation on a sub-14nm design. The figure compares a map of actual DRC violations with a map of congestion hotspots obtained by running a state-of-the-art industrial global router on the same layout; an overlay of these two maps is also shown. The GR-based congestion map is thresholded so that both maps display the same number of violating grid cells.[1]

The miscorrelation between the congestion map and the actual DRC hotspot map demonstrates why the routability improvement techniques traditionally employed during physical synthesis are not very effective at advanced process nodes (since they are driven by GR-based congestion maps). At the same time, it is critical to model and optimize routability during the physical synthesis stage, since the netlist and layout transforms that are permissible during routing-based optimization—when the DRC violations actually manifest themselves—are very limited in scope. This motivates the study of better DRC hotspot prediction tech-

[1]We use the GR-based congestion map as the reference because this is still the most common way to estimate routability in industrial physical implementation tools and flows. We generate the congestion maps by summing up numbers of overflows on each metal layer.

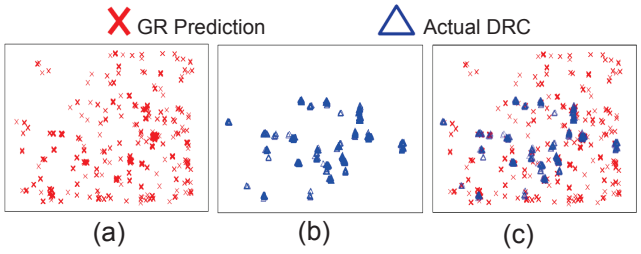

Figure 1: Comparison between the GR-based congestion map and the actual DRC violations (gcells with DRCs) on a sub-14nm design. (a) Overflows extracted from GR-based congestion map (per gcell). (b) DRC violations after detailed routing. (c) An overlay of the GR-predicted overflows and actual DRCs, highlighting the numerous false positive and false negative predictions. The placer and global router are from a state-of-the-art industrial physical design platform.

niques and their use for routability improvement without hurting the timing convergence of the design. In this paper, we describe a new algorithm that employs machine learning to accurately identify and optimize routability hotspots during physical synthesis without any timing or area overhead; furthermore, we demonstrate the effectiveness of our approach on industrial benchmarks in a sub-14nm process node from a leading foundry.

1.2 Related Work

Ensuring the routability of designs has always been a central challenge in IC implementation. Attempts to address this problem are usually most effective at the physical synthesis stage. In parallel, the problem of routability misprediction has also attracted interest in the EDA research community. We categorize and summarize previous works in these domains as follows.

Congestion predictors. Taghavi et al. [24] propose *MILOR* to identify local routing hotspots by examining pin-shape layouts and their densities or proximities within the placed design. Chan et al. [4] propose a learning-based methodology to predict the overall routability of a design by using placement information. Zhou et al. [30] use MARS to model DR congestion.

Routability-aware global routing. Qi et al. [20] improve the DR model from [30] to guide the global router. Wang et al. [26] and Zhong et al. [29] also propose the use of DR model-guided global routers. However, such works typically apply only to the routing stage, during which the allowed netlist and layout transforms are limited.

Congestion-aware placement. There has been significant work on congestion-aware placement in both industrial and academic coarse placers. For example, [12] [10] [8] [15] [16] [17] [19] [14] use global routers to predict congestion and feed the information back to the placer in order to improve routability. The works [3] [25] [9] [22] [13] [21] [27] use spreading regions, densities of nets, or routing patterns to estimate congestion and guide placement. However, such works are typically limited by their reliance on GR-based congestion maps.

White space optimization. When the pin accessibility problem dominates the DRCs, it is effective to use cell inflation during placement to improve the routability. Sadakane et al. [23] first addressed pin accessibility in the context of metal gate arrays, using a simulated annealing approach. [11] and [2] incorporate empirical heuristics to guide cell inflation in two academic placers to improve the routability. Instead of considering added space as attachments to (bloated) cells, [28] [18] [5] [1] handle the white space as separate components in placement, which enables more proactive control of white space to improve routability. However, the introduction of white space typically involves a timing and area overhead, especially when it is conservatively driven by poor routability predictor metrics such as GR-based congestion maps.

Our work is different from the previous works in several significant ways. (1) Rather than merely predicting routability, we show how to use machine learning to automatically *improve* the routability of the design. (2) Our engine focuses on improving route completion at the detailed routing stage, rather than optimizing GR congestion (and, does so without hurting the timing closure of the design). (3) Rather than merely predicting whether the overall design is routable or not, we use learning to predict the actual locations of the DRC hotspots. (4) Our predictor comprehends global routing, netlist structure, and cell-layout level information to capture routability risks due to both routing resource shortage and complicated design rules.

1.3 Overview of Our Work

In order to overcome the miscorrelation between the GR-based congestion map and the actual DRC map, we use machine learning to improve the accuracy of our identification of potential DRC hotspots. We model this prediction problem as a supervised classification problem. We label DRC-violating gcells in our training set of IC layouts with true labels, and cleanly-routed gcells with false labels. We then extract various parameters from the training netlists and layouts and use them to build an accurate predictor. Using this predictor, we propose an engine that minimally perturbs the converged physical synthesis netlist in a way that surgically redistributes the white space in the vicinity of the predicted DRC hotspots so as to ameliorate those hotspots without hurting timing, area or wirelength. We demonstrate the effectiveness of this approach on several layouts at a sub-14nm process node from a leading foundry. Our results show that we can reduce DRC violations by up to 76.8% and by an average of 20.6%, without hurting design convergence.

The key contributions of our work are as follows.

1. We present the first application of the machine learning paradigm to actually *optimize* design routability (in contrast to *predicting* routability, as in [4]).
2. We quantify the miscorrelation between a GR-based congestion map and the actual DRC map in designs at a sub-14nm process node.
3. We use machine learning to predict actual DRC locations in a design layout and use the prediction to improve post-placement routability. This is a significantly more difficult problem than the binary predic-

tion made in [4] on whether the overall design would be routable or not.

4. We develop an engine that employs our new learning-based predictor of DRC hotspots to ameliorate these hotspots without hurting timing, area or wirelength.

2. OUR APPROACH

We use machine learning to generate a model to close the gap between DRC hotspot prediction using congestion map and actual DRC violations located after detail routing. To enable accurate predictions, our model incorporates diverse parameters that we describe in Section 2.1.

With the help of this robust and accurate predictor, we are able to guide the optimization effectively to improve design routability. We design an algorithm that can leverage this predictor to ameliorate the DRC hotspots with minimal layout perturbation, thus avoiding timing or area penalties. This algorithm is described in more detail in Section 2.2.

2.1 Predictor Design

We use *hotspots to* refer to gcells with DRCs. The objective of our predictor is to separate hotspot gcells from non-hotspot gcells. In order to analyze the root causes of routability problems, we first partition training layouts into small grids on top of gcells (we use the term *local windows*[2] for these grids) to generate training data. We extract numerous netlist and layout parameters for each local window. These parameters include:

- **Density parameters** such as local pin density and local cell density;
- **GR parameters** obtained from a global routing invocation, such as local overflow, demand and capacity of each metal layer and via layer;
- **Pin proximity** measuring the average and minimum spacings between pins in each local window (proposed in [4]);
- **"Unfriendly" cells**, which are library cells that occur in the DRC hotspots (local windows with more than one DRC) at a rate significantly higher than their overall rate of incidence in the netlist;
- **Multi-height and sequential cells** and parameters relating to their fanins, fanouts, and occurrence frequencies in local windows;
- **Connectivity parameters** such as the number of buried nets completely enclosed inside the local windows, the number of non-buried nets crossing these window boundaries [4], and the number of connected pins lying outside the windows; and
- **Structural parameters** such as the number and depth of fanin and fanout logic stages in paths crossing local windows.

[2]We use two sizes of local windows, including 1×1 (gcell itself) and 3×3 windows (including a central gcell and the surrounding gcells). The 3×3 local windows have overlapping regions. Overlapping regions capture the influence among gcells in the DRC prediction.

Multi-height cells are cells with heights greater than that of a single cell row, which are typically sequential cells in this technology. Fanin or fanout numbers refer to the numbers of sequential cells transitively connected (incident or outgoing, respectively) to the cells within the current local window. The intuition of "structural parameters" is to evaluate the likelihood of cells to be placed around the sequential cells. The sequential cells are especially of interest because they have lower pin densities and different track heights.

We then go through an iterative process of training the predictor model using our parameter list. The iterative process contains both layout observation and evaluation of statistical significance.[3] In each iteration, we measure the statistical significance of the various parameters, and analyze the locations of both false-positive and false-negative predictions, in order to refine the parameter list and identify additional predictive features of the design. This process is repeated with several mathematical models of machine learning, *viz.*, linear regression, logistic regression, and support vector machines (SVM) [7] with various kernel choices.[4] As an illustration of the physical analysis involved in this iterative process, consider the following example.

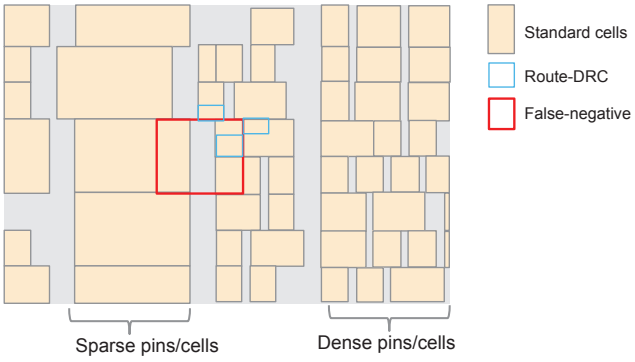

Figure 2: An example of the analysis of a false negative prediction.

Figure 2 shows an example of a DRC hotspot that was predicted to be DRC-clean in one of our earlier predictor models.[5] In our manual analysis of this hotspot, we find that the red false-negative region itself has low pin and cell density but is located close to a region with higher pin and cell density. We capture this anomaly (caused by the tendency of the router to introduce small detours around

[3]For the layout observation, we check the DRC locations and the cell placements. We also examine the false-positive and false-negative rates and the p-values (derived from the confusion matrices) after adding the parameters. We keep parameters that contribute to accuracy improvement in the process.

[4]We test among linear, polynomial, and radial basis function (RBF) kernels. We apply different weights to the DRC-violating gcells during model training and evaluate the model accuracy with testing gcells. RBF shows the best true-positive rate with similar false-positive rate with the first few parameters (density and GR). We choose RBF for our main experiments reported here.

[5]This reference predictor model has only basic parameters (density, GR overflow, etc.) and uses a single size (1×1) of local windows. The mathematical model for prediction is SVM.

DRC hotspots) in our predictor model by using larger local windows for such parameters, and by distinguishing between the parameter values inside a gcell, and the parameter values in a larger window that is centered on the gcell.

Besides incorporating different parameters to improve the prediction accuracy, we apply the following approaches to improve the accuracy and robustness of our local hotspot predictor. (1) Since the majority of the gcells in a typical layout do not have DRC violations, the training of the predictor is easily misguided by the biased distribution of the few DRC-violating gcells and the many DRC-clean gcells. We address this problem by emphasizing the DRC-violating gcells, increasing their weights[6] during the training stage. (2) Given that there are very few real-world sub-14nm designs and layouts available at this time, it becomes important to choose the training methodology carefully so as to avoid overfitting. We do this by randomly choosing 20% of the gcells from the layout to be the training data set and use the remaining 80% for testing. We repeat this randomized 20%-80% evaluation 12 times and use the average and distribution of the prediction accuracy from these 12 runs[7] to draw any conclusions about the tested parameter set and mathematical prediction model (*viz.*, linear regression, logistic regression, or SVM). (3) In order to take the neighborhood effect into consideration, we use both small and large local windows to annotate the central gcells. In addition to the multiple local window sizes, we also annotate the central gcell of each window with the extremal (i.e., maximum and minimum) values of selected parameters within the *expanded observation windows*.[8]

We use the *R* [31] statistical analysis package to prototype our predictor. As an illustration, the average true-positive and false-negative rates (over the 12 evaluations) for a series of evolving parameter sets are reported in Figure 3. We incrementally update the parameter set and mathematical model from predictor P1 through to predictor P9 based on our physical and statistical analysis of each of these predictors.

We use true-positive rate and false-negative rate to evaluate the statistical significance of prediction results. We compare the true-positive rates among combinations of different weighting[9] and the 12 evaluations for P{i} with or without a new set of parameters (unfriendly cells, sequential cells, etc.) If improvement is observed in the true-positive rate with the same false-positive rate constraint (typically, 0.5%), we accept the P{i} with new parameters as the P{i+1} predictor.

In Figure 3, P1 is the baseline set of predictors with 1×1 local windows, including pin density, cell density, and per-layer (metals and vias) GR capacity, demand, and overflow. P2 to P3 are variations of P1 with 3×3 windows and combinations of 1×1 and 3×3 windows.[10] P4 to P8 are the baseline predictors, combined with pin proximity, number of multi-height cells, number of unfriendly cells, connectivity parameters, and structural parameters. P9 uses the *Leaps* [32] package in *R* to pick predictors with high correlation to DRCs.[11]

Figure 3: Accuracy comparisons between different parameter sets and mathematical models. (a) Linear regression; (b) logistic regression; (c) SVM classifier.

We observe significant improvements in the prediction accuracy, especially in the false-positive rate, across this series. These plots also show that SVM (with a RBF kernel) can provide better separation between DRC gcells and non-DRC gcells than linear or logistic regression models. We also plot the predicted DRC hotspots (red squares) and actual DRC hotspots (blue squares) in Figure 4. The contrast with the corresponding figure (Figure 1) for GR-based predictions is readily apparent.[12] Our learning-based predictor provides a significantly more accurate prediction of the DRC hotspots, achieving 74% true-positive rate with a false-positive rate less than 0.2% (see Table 1). This is in contrast to a true-positive rate of 24% and a false-positive rate of 0.5% obtained from the GR-based predictor (Figure 1), as shown in Table 2.

2.2 Predictor-guided Routability Optimization

We present an optimization engine to improve the routability of a converged netlist with minimal perturbation of the layout. This algorithm is summarized in Figure 5. Given

[6]The DRC-clean cells are always weighted with one. We swept the weight of DRC-violated gcells with {2, 3, 4, 5,.... 10, 20, 30, 40, 50}.

[7]We pick 12 evaluations (larger than two times of 100%/20%) to avoid overfitting on specific training sets since we can only access one design in this technology.

[8]Two types of windows are mentioned in our discussion. The first type is the local window with two sizes (1×1 and 3×3 gcells) to extract per-layer GR information and other parameters. The second type is the expanded observation window with four sizes (3×3, 5×5, 7×7, 9×9) to keep track of max/min values within a certain range.

[9]We then choose the weight according to the true-positive rate for those runs with false-positive rates lower than a certain threshold (typically 0.5%).

[10]P1 to P3 are the same predictors with different local window sizes. We observe significant false-positive rate improvement when we compare P2 and P3 with P1 ($\sim$ 1% vs. $>$ 6%), due to different local window sizes.

[11]Note that P1 essentially uses only cell and pin density parameters from global routing and thus has a high false-positive rate even if SVM is used ($\sim$6%). This again indicates that global routing is not sufficient to predict DRC hotspots.

[12]Note that the learning-based model has an advantage over GR-based congestion map because it generates proper thresholds (i.e., support vectors) in the training stage.

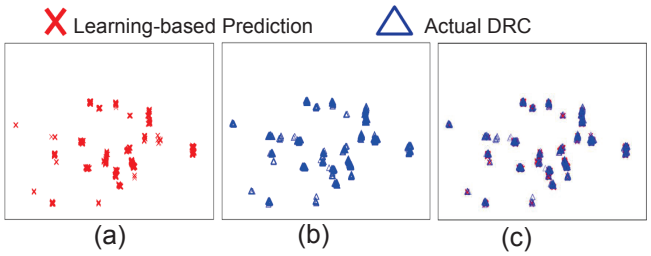

Figure 4: Comparison between our learning-based DRC hotspot map and the actual DRC violations (gcells with DRCs) on a sub-14nm design. (a) DRC hotspots predicted by our learning-based model. (b) Actual DRC violations after detailed routing (note that this is the same as Figure 1(b) presented earlier, and is reproduced here merely for comparison with the predicted map in (a)). (c) An overlay of the predicted and actual DRCs.

		Actual	
		FALSE	TRUE
Prediction	FALSE	98571	117
	TRUE	170	344

Table 1: Learning-based prediction: confusion matrix of our learning-based predictor. True-positive rate = 74% and false-positive rate = 0.2%.

		Actual	
		FALSE	TRUE
Prediction	FALSE	98260	350
	TRUE	481	111

Table 2: GR-based prediction: confusion matrix of prediction by GR shown in Figure 1. True-positive rate = 24% and false-positive rate = 0.5%.

the placement of a timing-converged netlist, we first use our DRC prediction model to identify the potential DRC hotspots in that layout. We have developed a local white space optimization engine that redistributes the white space already present in the neighborhood of the predicted DRC hotspots in a way that improves the routability of these hotspots. Real-world physical implementation flows are almost invariably run under some form of local cell density constraints in order to improve the flow convergence. Such constraints ensure the existence of white space everywhere in the layout when measured at some level of spatial granularity. However, this granularity is typically much larger than that of individual cells; therefore, this default white space distribution is not always able to resolve detailed routing DRC problems. This shortcoming is addressed effectively in our work by introducing a new, detailed white space redistribution stage that relies on our hotspot prediction model to minimize the perturbation to the layout while maximizing the routability impact of the redistribution.

The spreading is constructed based on the legalizer. First, the available whitespace size is collected in a given window. Then, we distribute a certain fraction of the available whitespace by adding small temporary keepout regions adjacent to the cells in the window, and run the legalizer to increase the cell spacing. This step incrementally moves the cells from the initial locations obtained from the original converged layout. The legalizer tends to abut the cells in order to

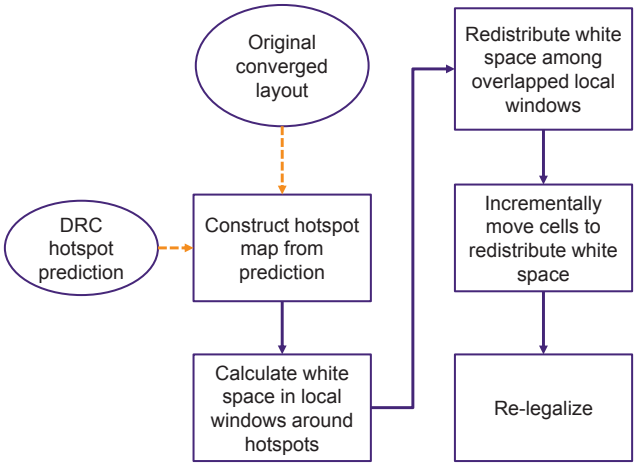

Figure 5: The predictor-guided routability optimization algorithm.

minimize the routed wirelength; the temporary keepouts help counter this behavior by introducing porosity in order to improve routability.

During the whitespace redistribution process, we first mark out local windows around each hotspot, and calculate the amount of white space available in these windows in order to compute a local white space budget (splitting overlapping windows appropriately during this process). We then incrementally distribute this budget among the cells in a row-major scanning sequence by gradually adding the aforementioned keepout region instances within the windows until the space budget is used up.

Subsequent to this white space redistribution step, the detailed routing engine can use the newly introduced space between problematic cell instances to handle the complicated design rules. This approach of applying redistribution to only the potential DRC hotspots and avoiding the introduction of any new white space there, while leaving the rest of the layout untouched, minimizes the perturbation to the already-converged layout, which in turn minimizes the likelihood of incurring any significant timing penalty. Indeed, our experimental results (discussed in Section 3) demonstrate large routability improvements from the application of this algorithm, without hurting timing.

3. EXPERIMENTAL RESULTS

In this section, we describe the experimental methodology that we have used to evaluate the effectiveness of our predictor-guided routability optimization algorithm. We then present the results of this evaluation.

Our experimental methodology is shown in Figure 6. Our routability optimization engine is implemented in *C++* as part of a state-of-the-art industrial physical implementation platform. The predictor model generation code is implemented using scriptware in the *R* [31] statistical analysis package.[13] We evaluate our algorithm using an industrial benchmark design at a sub-14nm process node from a leading foundry. Given the difficulty of obtaining additional real-world sub-14nm benchmarks at this time, we gener-

[13]The extraction and training scriptware is split across several machines to reduce turnaround time.

Table 3: Comparisons between the default and the learning-optimized layouts. There are under one million instances in this design.

	#DRCs			Wirelength			TNS (ns)			#FEPs		
	Base	**Test**	**%Change**	**Base**	**Test**	**%Change**	**Base**	**Test**	**%Change**	**Base**	**Test**	**%Change**
eg1	8478	1964	-76.8%	1742804	1747685	0.3%	-153.43	-158.4	3.2%	7289	7352	0.86%
eg2	1502	927	-38.3%	1750698	1753047	0.1%	-168.23	-163.5	-2.8%	7406	7374	-0.43%
eg3	2017	1819	-9.8%	1772889	1773701	0.0%	-215.75	-213.6	-1.0%	7817	7751	-0.84%
eg4	2026	1780	-12.1%	1735185	1735227	0.0%	-151.36	-149.6	-1.2%	7195	7143	-0.72%
eg5	4252	4255	0.1%	1831492	1836060	0.2%	-264.34	-275.6	4.3%	7865	7975	1.40%
eg6	3440	3891	13.1%	1790059	1794184	0.2%	-195.65	-203.5	4.0%	7587	7562	-0.33%
		Avg	-20.6%		Avg	0.2%		Avg	1.1%		Avg	-0.01%
		Max	13.1%		Max	0.3%		Max	4.3%		Max	1.40%
		Min	-76.8%		Min	0.0%		Min	-2.8%		Min	-0.84%

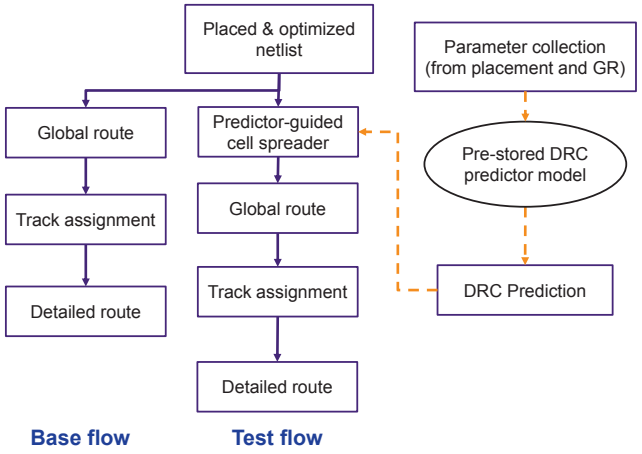

Figure 6: The experimental methodology used to evaluate the proposed predictor-guided routability optimization algorithm.

ate multiple widely-differing netlists and layouts from our benchmark design by running the design through an industrial congestion-aware physical synthesis flow with different (but still realistic) placement and optimization settings.

The input to our evaluation is an optimized netlist and a legalized placed layout obtained from the physical synthesis flow as described above. The layout has already been optimized using traditional congestion alleviation techniques during physical synthesis. Our base flow is the typical physical implementation flow that takes netlist and layout through standard global routing, track assignment and detailed routing using the state-of-the-art router embedded in our industrial physical implementation platform. For our test flow, we first use the pre-stored predictor to predict DRC hotspots in our starting layout. This prediction is then used by our routability optimization engine for localized white space redistribution, followed by legalization. The resulting layout is fed to the same router as in the base flow.

We report the number of DRC violations, total negative timing slack (TNS), wirelength, and number of failing timing endpoints (#FEP) at the end of detailed routing in the base and test flows. The results are shown in Table 3.[14] In this table, negative values in the "%Change" columns refer to improvements achieved in the test flow (relative to the base flow), and positive values in these columns refer to degradations. As is evident from this data, we achieve significant reduction of the DRC count in the test flow. Moreover, these routability improvements are obtained without any significant impact on design closure: the timing and wirelength impacts are neutral, and the design area is unchanged. More specifically, we see that the number of DRC violations reduces by an average of 20.6% and a maximum of 76.8%, with TNS degrading by an average of 1.1%, and the number of failing timing endpoints improving by an average of 0.01%, and wirelength degrading by an average of 0.2%.

[14]In our experiments, we do not retrain the model for these regenerated layouts.

4. CONCLUSIONS

In this paper, we have addressed the route completion problem for designs at advanced process nodes. We make the case that traditional routability amelioration approaches that rely on GR-based congestion maps during physical synthesis are no longer effective at these advanced nodes due to the complexity of the design rules. We quantify this difficulty by measuring the accuracy of a GR-based prediction of routability hotspots. We then propose a machine learning based algorithm to automatically improve the routability of these designs without hurting timing convergence. We evaluate the effectiveness of this algorithm on design layouts at a sub-14nm process node from a leading foundry, using an industrial physical implementation platform. Our experiments show that we are able to reduce the number of DRC violations by an average of 20.6% and a maximum of 76.8%, with no adverse impact on design closure. Our future works include (1) improving the prediction accuracy and spreading results, (2) guiding the routability optimization by applying the machine learning model to coarse placement, and (3) applying our methodology to other advanced node technologies.

5. ACKNOWLEDGMENTS

We would like to express our gratitude to T. Andersen, B. Gregory, S. Nath, W. Naylor and J. Wong for a number of valuable discussions, and to J. Wong for helping us set up the experimental framework.

6. REFERENCES

[1] S. N. Adya, I. L. Markov and P. G. Villarrubia, "On Whitespace and Stability in Mixed-Size Placement

and Physical Synthesis" *Integration, the VLSI Journal* 39(4) (2008), pp. 340-362.

[2] U. Brenner and A. Rohe, "An Effective Congestion Driven Placement Framework", *Proc. ISPD*, 2002, pp. 6-11.

[3] A. E. Caldwell, A. B. Kahng, S. Mantik, I. L. Markov and A. Zelikovsky, "On Wirelength Estimations for Row-based Placement" *IEEE Trans. on CAD* 18(9) (1999), pp. 1265-1278.

[4] W.-T. J. Chan, Y. Du, A. B. Kahng, S. Nath and K. Samadi, "BEOL Stack-Aware Routability Prediction from Placement Using Data Mining Techniques", *Proc. ICCD*, 2016, pp. 41-48.

[5] A. E. Caldwell, A. B. Kahng and I. L. Markov, "Hierarchical Whitespace Allocation in Top-down Placement" *IEEE Trans. on CAD* 22(11) (2003), pp. 1550-1556.

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

[7] T. Hastie, R. Tibshirani and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, Springer, 2009.

[8] X. He, T. Huang, W.-K. Chow, J. Kuang, K.-C. Lam, W. Cai and E. F. Y. Young, "Ripple 2.0: High Quality Routability-Driven Placement via Global Router Integration", *Proc. DAC*, 2013, pp. 1-6.

[9] X. He, T. Huang, L. Xiao, H. Tian, G. Cui and E. F. Young, "Ripple: An Effective Routability-Driven Placer by Iterative Cell Movement", *Proc. ICCAD*, 2011, pp. 74-79.

[10] M.-K. Hsu, S. Chou, T.-H. Lin and Y.-W. Chang, "Routability-Driven Analytical Placement for Mixed-Size Circuit Designs", *Proc. ICCAD*, 2011, pp. 80-84.

[11] W. Hou, H. Yu, X. Hong, Y. Cai, W. Wu, J. Gu and W. H. Kao, "A New Congestion-Driven Placement Algorithm Based on Cell Inflation", *Proc. ASP-DAC*, 2001, pp. 606-608.

[12] Z.-W. Jiang, B.-Y. Su and Y.-W. Chang, "Routability-Driven Analytical Placement by Net Overlapping Removal for Large-scale Mixed-Size Designs", *Proc. DAC*, 2008, pp. 167-172.

[13] A. B. Kahng and X. Xu, "Accurate Pseudo-Constructive Wirelength and Congestion Estimation", *Proc. SLIP*, 2003, pp. 61-68.

[14] M.-C. Kim, J. Hu, D.-J. Lee and I. L. Markov, "A SimPLR Method for Routability-Driven Placement", *Proc. ICCAD*, 2011, pp. 67-73.

[15] W.-H. Liu, T.-K. Chien and T.-C. Wang, "A Study on Unroutable Placement Recognition", *Proc. ISPD*, 2014, pp. 19-26.

[16] W.-H. Liu, T.-K. Chien and T.-C. Wang, "Region-Based and Panel-Based Algorithms for Unroutable Placement Recognition" *IEEE Trans. on CAD* 34(4) (2015), pp. 502-514.

[17] W.-H. Liu, Y.-L. Li and C.-K. Koh, "A Fast Maze-Free Routing Congestion Estimator with Hybrid Unilateral Monotonic Routing", *Proc. ICCAD*, 2012, pp. 713-719.

[18] C. Li, M. Xie, C.-K. Koh, J. Cong and P. H. Madden, "Routability-Driven Placement and White Space Allocation", *Proc. ICCAD*, 2004, pp. 394-401.

[19] M. Pan and C. Chu, "IPR: An Integrated Placement and Routing Algorithm", *Proc. DAC*, 2007, pp. 59-62.

[20] Z. Qi, Y. Cai and Q. Zhou, "Accurate Prediction of Detailed Routing Congestion using Supervised Data Learning", *Proc. ICCD*, 2014, pp. 97-103.

[21] J. A. Roy and I. L. Markov, "Seeing the Forest and the Trees: Steiner Wirelength Optimization in Placement" *IEEE Trans. on CAD* 23(4) (2007), pp. 632-644.

[22] P. Spindler and F. M. Johannes, "Fast and Accurate Routing Demand Estimation for Efficient Routability-Driven Placement", *Proc. DATE*, 2007, pp. 1226-1231.

[23] T. Sadakane, H. Shirota, K. Takahashi, M. Terai and K. Okazaki, "A Congestion-Driven Placement Improvement Algorithm for Large Scale Sea-of-gates Arrays", *Proc. CICC*, 1997, pp. 573-576.

[24] T. Taghavi, C. J. Alpert, A. Huber, Z. Li, G.-J. Nam and S. Ramji, "New Placement Prediction and Mitigation Techniques for Local Routing Congestion", *Proc. ICCAD*, 2010, pp. 621-624.

[25] K. Tsota, C.-K. Koh and V. Balakrishnan, "Guiding Global Placement with Wire Density", *Proc. ICCAD*, 2008, pp. 212-217.

[26] M. Wang, X. Yang, K. Eguro and M. Sarrafzadeh, "Multicenter Congestion Estimation and Minimization during Placement", *Proc. ISPD*, 2000, pp. 147-152.

[27] J. Westra, C. Bartels and P. Groeneveld, "Probabilistic Congestion Prediction", *Proc. ISPD*, 2004, pp. 204-209.

[28] X. Yang, B.-K. Choi and M. Sarrafzadeh, "Routability-Driven White Space Allocation for Fixed-Die Standard-Cell Placement" *IEEE Trans. on CAD* 22(4) (2003), pp. 410-419.

[29] K. Zhong and S. Dutt, "Algorithms for Simultaneous Satisfaction of Multiple Constraints and Objective Optimization in a Placement Flow with Application to Congestion Control", *Proc. DAC*, 2002, pp. 854-859.

[30] Q. Zhou, X. Wang, Z. Qi, Z. Chen, Q. Zhou and Y. Cai, "An Accurate Detailed Routing Routability Prediction Model in Placement", *Proc. ASQED*, 2015, pp. 119-122.

[31] The R Project for Statistical Computing, https://www.r-project.org/

[32] Leaps package, https://cran.r-project.org/web/packages/leaps/