BEOL Stack-Aware Routability Prediction from Placement Using Data Mining Techniques

Wei-Ting J. Chan[‡], Yang Du[⋆], Andrew B. Kahng[†‡], Siddhartha Nath[†] and Kambiz Samadi[⋆]
[†]CSE and [‡]ECE Departments, UC San Diego, La Jolla, CA 92093
[⋆]Qualcomm Research, San Diego, CA
{wechan, abk, sinath}@ucsd.edu, {ydu, ksamadi}@qti.qualcomm.com

***Abstract*—In advanced technology nodes, physical design engineers must estimate whether a standard-cell placement is routable (before invoking the router) in order to maintain acceptable design turnaround time. Modern SoC designs consume multiple compute servers, memory, tool licenses and other resources for several days to complete routing. When the design is unroutable, resources are wasted, which increases the design cost. In this work, we develop machine learning-based models that predict whether a placement solution is routable without conducting trial or early global routing. We also use our models to accurately predict iso-performance Pareto frontiers of utilization, aspect ratio and number of layers in the back-end-of-line (BEOL) stack. Furthermore, using data mining and machine learning techniques, we develop new methodologies to generate training examples given very few placements. We conduct validation experiments in three foundry technologies (28nm FDSOI, 28nm LP and 45nm GS), and demonstrate accuracy $\geq 85.9\%$ in predicting routability of a placement. Our predictions of Pareto frontiers in the three technologies are pessimistic by at most 2% with respect to the maximum achievable utilization for a given design in a given BEOL stack.**

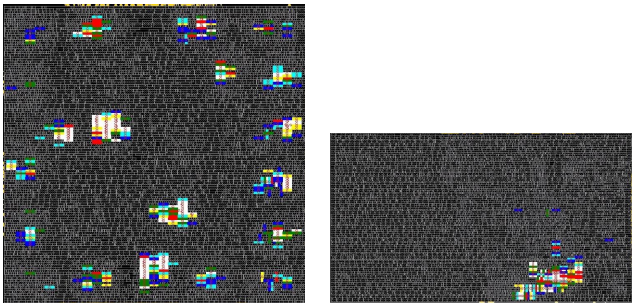
I. Introduction

Physical design of digital integrated circuits in advanced technology nodes is complicated by multiple design rules that must be satisfied before tapeout. Design rule violations are reported by commercial place-and-route (P&R) tools after the routing stage. However, discovering many design rule violations post-routing is costly: at best, it consumes engineer resources to fix all the violations and increases design turnaround time. Sometimes, the number of design rule violations is so large as to be unfixable; this scenario leads to disruptive changes to the placement, layout contexts and constraints. Early prediction of routability in the physical design flow is therefore critical to reduce design turnaround time and cost. Multiple factors affect routability such as timing constraints, utilization, aspect ratio, the number of metal layers, etc. However, to our best knowledge, no routability models exist today that enable IC physical design engineers to perform fast and accurate design-space exploration of timing constraints, utilization, aspect ratio and the number of metal layers in the back-end-of-line (BEOL) stack. Today, designers use congestion maps from P&R tools at the placement stage to predict routability. Congestion maps may be insufficient (and even misleading) for prediction of routability as measured by the number of design rule check (DRC) violations.[1] This is because routing tools are themselves highly unpredictable, and because congestion maps may not comprehend design rules or other factors that affect local routability such as pin density or timing criticality.

A. Motivation

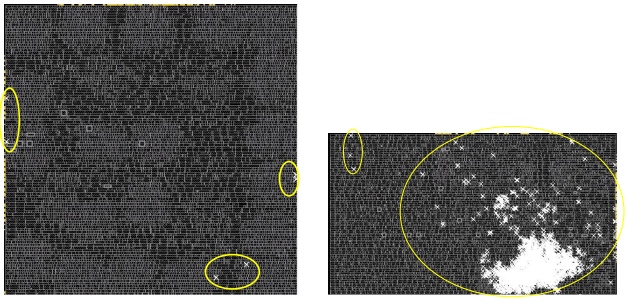
Physical design engineers typically use congestion maps from trial routing at the placement stage to determine whether a given placement is likely to be routable. However, this is still largely an "art", akin to reading tea leaves, as congestion maps are not straightforward indicators of design rule violations in detailed routing.

As a motivating illustration, we consider implementations of *aes_cipher_top* [39] and ARM *Cortex M0* designs in 28nm FDSOI with eight-track cells and a five-layer BEOL stack, obtained using a commercial P&R tool. We implement *aes_cipher_top* with aspect ratio 1.0 and *Cortex M0* with aspect ratio 2.0. Figures 1(a) and (b) show layouts and congestion maps of *aes_cipher_top* and *Cortex M0*, respectively. Red regions indicate congestion overflow (i.e., difference in supply and demand of routing resources) < -5 and ≥ -7; white regions indicate overflow < -7. From these maps, a user might surmise that the *aes_cipher_top* placement is unroutable due to many regions of high congestion, while the *Cortex M0* placement is routable (or, routable with a manually-fixable number of DRCs) since it has only a few hotspots in the right-bottom region. However, Figure 2(a) shows that the ∼6 post-routing DRC violations for *aes_cipher_top* do not occur in the highly congested regions. Figure 2(b) shows that post-routing DRC violations for *Cortex_M0* occur in both the congested and uncongested regions, and that the placement is unroutable. The reasons for these DRC violations are not intuitive from examination of the congestion maps. We demonstrate below that by using relevant placement-derived parameters beyond congestion map information, we can train models that accurately predict routability of a given BEOL stack-specific placement.[2]



(a) (b)

Fig. 1. Congestion maps at placement stage in 28nm FDSOI foundry technology, with 8T cells, of (a) *aes_cipher_top* implementation with 77% utilization, aspect ratio 1.0 and five-layer BEOL stack and (b) ARM *Cortex M0* implementation with 77% utilization, aspect ratio 2.0 and five-layer BEOL stack. Red and white regions indicate large congestion with overflow < -5.



(a) (b)

Fig. 2. DRC violations after routing in 28nm FDSOI foundry technology, with 8T cells, of (a) *aes_cipher_top* implementation with 77% utilization, aspect ratio 1.0 and five-layer BEOL stack and (b) ARM *Cortex M0* implementation with 77% utilization, aspect ratio 2.0 and five-layer BEOL stack. The white crosses show the DRC violations and the yellow ovals further highlight the violations.

[1]In the following, we use #DRCs to denote the number of design rule violations (after the routing tool has run).

[2]We use the term "BEOL stack-specific placement" to acknowledge that the placement tool will output different placement solutions for the same netlist and site map, according to the specific metal layer stack.

B. Our work

We define a placement to be *routable* when the #DRCs is $<$ *threshold* after the routing stage; conversely, a placement is *unroutable* when the #DRCs is $\geq$ *threshold* after the routing stage.[3] In this work, we develop models using data mining or machine learning techniques to accurately predict whether a BEOL stack-specific placement is routable. We study and propose placement-derived parameters that enable us to achieve high prediction accuracy. Applications of our models include the following use cases.

- Given a netlist, clock period, utilization, aspect ratio and BEOL stack-specific placement, our models predict whether the placement will be routable. In existing implementation flows, our models will evaluate solutions at the placement stage and guide designers to either proceed with the solution or modify it to make it routable.
- Our models predict iso-performance Pareto frontiers of utilization, number of metal layers and aspect ratio based on very few (≤ 20) placements (and, no routing or trial routing) of a design. Using these Pareto frontiers, a designer can determine the minimum number of metal layers[4] or the maximum achievable utilization of a block. In existing implementation flows, our predictions of Pareto frontiers can guide designers to perform better implementation-space exploration of BEOL stack options and layout contexts at the floorplan stage.

To the best of our knowledge, the above use cases are not well-supported in current design methodologies and flows. As noted above, physical design engineers estimate routability based on congestion maps from commercial P&R tools, but such maps can be quite misleading. Currently, to our knowledge, no tool exists that predicts the Pareto frontiers of utilization, number of metal layers and aspect ratio based on very few placements.

The key contributions of our work are as follows.

1) We demonstrate that congestion maps from commercial tools are likely insufficient to predict routability. E.g., we obtain a classification error of 33.6% in 45nm GS technology when only values from congestion maps are used to predict routability.
2) We describe a methodology based on machine learning to predict whether a placement will be routable, given a netlist, clock period, utilization, aspect ratio and BEOL stack.
3) We describe new parameters that we identify – related to congestion distribution, critical timing path distribution, and available routing resources [13] [14] – that guide our learning-based models to accurately predict routability of a placement, given a netlist, clock period, utilization, aspect ratio and BEOL stack. In doing this, we do not use any information from trial routing or early global routing from P&R tools. The worst-case classification error in our models is 14.1% in 45nm GS. In 28nm FDSOI, the classification error of our model is no more than 13%.
4) Our models also enable accurate prediction of *iso-performance Pareto frontiers* of utilization, number of metal layers and aspect ratio, based on very few placements.

The remainder of this paper is organized as follows. In Section II, we review related works. Section III describes our methodology to identify new and relevant parameters, interpolation / extrapolation methods of parameter values and modeling methodology. In Section IV, we describe our design of experiments for training and testing and present results in three different foundry technologies. In Section V, we provide conclusions and outline ongoing work.

[3]In this work, we set the *threshold* to 50, i.e., we assume (rather conservatively) that 50 DRC violations remaining after detailed routing can be manually fixed.

[4]In advanced nodes, due to complexity of lithography (e.g., double-patterning, triple-patterning, etc.) and process options such as airgaps in intralayer dielectrics, each metal layer can account for a sizeable percentage of wafer cost.

II. Related Work

We review related work that addresses (i) requirements for and optimizations of the number of metal layers needed for routing, and (ii) early placement-based estimation of congestion and routability.

Number of metal layers. Dong et al. [4] show that only five metal layers are required for designs with up to 5M gates for signal and clock routing only (i.e., no PDN). Thereafter, the number of layers scales linearly, e.g., 50M gates require eight layers and 100M gates require 10 layers. The study in [4] uses analytical models and gate areas from a 65nm process, but does not specify metal layer pitches relative to the pitch of the minimum-width layer. Andreev et al. [1] have patented a dynamic programming technique that assigns segments of signal nets from M1 through to the top metal layer, minimizing the amount of metal layer area consumed by vias. The patent assumes that all layers have the same pitch. It does not describe a way to achieve a minimum number of metal layers or a maximum achievable utilization consistent with routability. The patent of Lin [17] describes a method to choose widths of various metal layers in the stack to minimize IR drop and RC delay of nets. The patent also describes material choices for pads and dielectric to achieve minimum RC delay.

Early estimation of congestion. Congestion estimation has attracted much attention in the research community. To address routability issues prior to the routing stage and to minimize turnaround time, modern placers are equipped with congestion estimators to guide the placement to achieve router-friendly placement solutions. Brenner et al. [2] and Jiang et al. [10] propose approaches to estimate congestion at the global routing stage for congestion-driven placement. Caldwell et al. [3] accurately estimate routed wirelength by comprehending floorplan aspect ratios for routability-driven placement. Roy et al. [24] propose a congestion-driven whitespace allocation algorithm during placement and they apply their algorithm in their placement tool ROOSTER. Wang et al. [32] and Zhong et al. [36] propose approaches to cure hotspots at the global routing stage. To predict congestion, Tsota et al. [30] use density of nets within a bounding box; He et al. [6] apply decomposition of multi-pin nets into two-pin nets; Spindler and Johannes [27] exploit the ratio of a net's wire area (obtained by enumerating all possible RSMTs) to the area of a net's bounding box; and Hsu et al. [8] use pin density and routing resources. The authors of [8] use the predicted congestion to modify their global placer's cost function and reduce congestion. Westra et al. [34] extract routing patterns (L/Z-shapes) to predict congestion. Kahng and Xu [15] propose a statistical model for congestion that comprehends effects of blockages and routing bends. He et al. [7], Liu et al. [18] [19] [20], Pan and Chu [22], and Kim et al. [16] use global routers to predict congestion. Westra et al. propose a constructive approach in [35] for placement. Furthermore, Taghavi et al. [29] propose *MILOR* to avoid routing infeasibility due to local congestion at the placement stage. Shojaei et al. [25] [26] propose a congestion-estimation framework with integer linear programming at the global routing stage. Wei et al. [33] propose *Glare* for local and global congestion estimation. Qi et al. [23] use multivariate adaptive regression splines (MARS) to develop predictors of routing congestion using pin density and congestion maps from global routing. The authors report 13% reduction in the number of design rule violations when their predictor is used, as compared to using analytical models of routing congestion. Zhou et al. [37] propose a learning-based congestion model for detailed routing. The authors use parameters from global routing and achieve accuracy of $\sim$80% using MARS.

III. Our Approach

We now describe our modeling parameters, how we have identified them, and our modeling methodology.

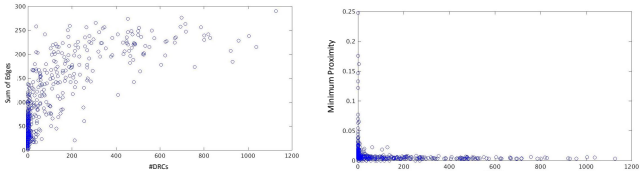
A. List of Parameters

We divide the placement region into grids whose height and width are multiples of the P&R tool's global routing cell (gcell) height

and width.[5] We extract modeling parameters from these grids that intuitively affect local routing of net segments on various metal layers. The placement-derived parameters that we obtain from each grid are:

- pin density;
- minimum proximity of any pair of pins;
- number of complex cells (i.e., *AOI*, *OAI*, three-input *XOR* and *XNOR*, and *MUX*);
- sum of incoming and outgoing hyperedges (signal nets with pins both inside and outside the grid);
- number of buried nets, that is, the number of nets that have all of their pins within the grid;
- arithmetic and geometric mean values of placement-based Rent parameter;
- the worst signal transition time of all pins at the worst corner; and
- the smallest values of the worst negative slack (WNS) of setup time of any pin within the grid.

Figures 3(a) and (b) show correlation of #DRCs (our routability metric) to the sum of incoming and outgoing hyperedges[6] and minimum proximity of pins, respectively. When a grid has small values of minimum proximity of pins, it indicates that pins of adjacent cells within a grid are placed very closely and can lead to spacing-related DRC violations at the routing stage. When a grid has pins with large transition times or small WNS, it indicates that cells can be sized up and buffers can be inserted that can worsen local routability and increase the number of DRC violations.



(a) (b)

Fig. 3. Correlations of #DRCs with (a) sum of incoming and outgoing hyperedges and (b) minimum proximity of pins within a grid.

From the above parameters, we compute the coefficient of variation (i.e., the ratio of standard deviation to mean) of pin density, minimum proximity, number of complex cells, sum of incoming and outgoing hyperedges, number of buried nets, arithmetic and geometric mean values of placement-based Rent parameter, worst transition time and worst WNS. We use the following as our modeling parameters.[7]

1) Coefficient of variation of pin density, minimum proximity, number of complex cells, sum of incoming and outgoing edges, number of buried nets, arithmetic and geometric mean values of Rent parameter, worst transition time and worst WNS.
2) Maximum values (across all grids) of pin density, number of complex cells, sum of incoming and outgoing edges, number of buried nets, arithmetic and geometric mean values of Rent parameter and worst transition time.
3) Minimum values (across all grids) of minimum proximity and worst WNS.
4) Utilization of standard cells.
5) Clock period of design used for P&R.
6) Aspect ratio of the floorplan.
7) Numbers of horizontal and vertical tracks, which we calculate from the height and width of the placement region and pitches of all horizontal and vertical layers of the BEOL stack. For example, if there are three horizontal layers, each with pitch p_h, and core height H, then the number of horizontal tracks is $3 \cdot H/p_h$.

The parameters listed in (1)–(3) above are an indication of the quality of a BEOL stack-specific placement and how it spreads across multiple grids. If the spread (indicated by coefficient of variation) is large, it suggests that grids may have local congestion. The timing parameters capture how critical paths and critical pins are distributed, and the extent of violation. Large violations at the placement stage indicate that buffers can be inserted during routing, which can increase local congestion and violate design rules. The parameters listed in (4)–(6) above describe the layout context and timing constraints, and the parameters listed in (7) above capture details of the BEOL stack and the amount of routing resources available for the design.[8]

B. Parameter value interpolation and extrapolation

Given a few (≤ 20) placement solutions of a design that span some values of clock period, utilization, aspect ratio and BEOL stack, we need to generate additional values of parameter to train our models. We propose the following methodology, partially adapted from [21], to interpolate and extrapolate values of parameters (e.g., {maximum, coefficient of variation, ...} $\times$ {pin density, sum of edges, ...}) from Section III-A across multiple values of clock period, utilization, aspect ratio and BEOL stack. We train models for each parameter as a function of clock period, utilization, aspect ratio, number of horizontal (#H) and vertical (#V) tracks and known values of the parameter extracted from the given placements. We train models for each parameter that achieves a given error bound UB_{error} as follows.

Algorithm 1 Interpolation / extrapolation of parameter values.

Procedure $genParamModel$
Input : $\mathbb{P}$ ($3 < |\mathbb{P}| \leq 20$), UB_{error}
Output: Model f_m for parameter $m \in$ {max pin density, max # edges, etc.}.

1: $\mathbb{P}_{tr} \leftarrow \{p_i, p_j, p_k\} \in \mathbb{P}$, $i, j, k \leq |\mathbb{P}|$
2: $\mathbb{P} \leftarrow \mathbb{P} \setminus \{p_i, p_j, p_k\}$ // remove p_i, p_j, p_k from $\mathbb{P}$
3: $p_f \leftarrow \{p_l\} \in \mathbb{P}$, $l \leq |\mathbb{P}|$
4: $\mathbb{P} \leftarrow \mathbb{P} \setminus \{p_l\}$
5: $\mathbf{X}_{tr} \leftarrow \mathbb{P}_{tr} \cup \{p_f\}$ // extract inputs to f_m, e.g., ratio of util, clk period, ...
6: $y_{tr} \leftarrow p_f$ // extract value of parameter m in p_f
7: $\hat{y}_{tr} \leftarrow f_m(\mathbf{X}_{tr})$ // f_m is trained using MARS, SVM, etc.
8: $e \leftarrow 2\times \text{UB}_{error}$
9: **while** $|\mathbb{P}| > 0$ && $|e| > \text{UB}_{error}$ **do**
10: **for all** $p \in \mathbb{P}$ **do**
11: $y \leftarrow p$
12: $\mathbf{X}_{test} \leftarrow \mathbb{P}_{tr} \cup \{p\}$
13: $e \leftarrow f(\mathbf{X}_{test}) - y$
14: **if** $|e| > \text{UB}_{error}$ **then**
15: $\mathbb{P}_{tr} \leftarrow \mathbb{P}_{tr} \cup \{p\}$
16: $\mathbb{P} \leftarrow \mathbb{P} \setminus \{p\}$
17: $\mathbf{X}_{tr} \leftarrow \mathbb{P}_{tr} \cup \{p_f\}$
18: $y_{tr} \leftarrow p_f$
19: $\hat{y}_{tr} \leftarrow f_m(\mathbf{X}_{tr})$ // retrain model
20: **end if**
21: **end for**
22: **end while**

In procedure *genParamModel* of Algorithm 1, we assume that we are given a set $\mathbb{P}$ of placements of a design and an error upper bound UB_{error}. We expect a minimum of four and a maximum of 20 placements, that is $3 < |\mathbb{P}| \leq 20$.[9] In Lines 1–4, we choose a subset of three placements $\mathbb{P}_{tr}$ from $\mathbb{P}$ for training and one placement p_f for model fitting. In Line 5, we extract values of the parameter (e.g., max pin density, etc.) as well as other inputs used to fit a model. These inputs are the ratio of clock periods, utilization, aspect ratio, #H and #V tracks of p_f to those in $\mathbb{P}_{tr}$, as well as the values of the same parameter from the placements in $\mathbb{P}_{tr}$. In Line 6, we obtain

[5]Typical grid size in a commercial P&R tool is 15 tracks $\times$ 15 tracks [38], where a track is equal to the M2 pitch value.

[6]We use the term edges below to denote hyperedges.

[7]We greedily select parameters by incrementally adding each parameter one by one, creating models using our training dataset and checking accuracy of models using the test dataset. The next parameter to be selected is the one which improves model accuracy the most. We list the parameters that achieve the highest accuracy in our experiments.

[8]Our models are sensitive to the implementation flow and tools used to generate training data, and do not generalize across flows and tools. Therefore, our models must be re-trained when the flow or tools change.

[9]Given placement solutions of a new design, we do not know how each parameter varies with the inputs. We use at least three data points to capture parameters that can be polynomial (with polynomial degree ≥ 2) with respect to the input parameters.

the value of the parameter from p_f that we use to fit, and in Line 7 we train a model f_m. For example, to train a model for max pin density, we use the max pin density values of placements in $\mathbb{P}_{tr}$ and the ratio of clock periods, utilization, aspect ratio, #H and #V tracks of p_f to those in $\mathbb{P}_{tr}$. In Line 8, we initialize the error e to $2\times$ UB$_{error}$. In Lines 9–22, we refine f_m by retraining to achieve $e \leq$ UB$_{error}$. In Lines 9–13, we choose the remaining placements in $\mathbb{P}$ for testing, extract the fitting parameters and the actual values of the parameters. We then use f_m to test the model. In Lines 14–20, we check if the error $e >$ UB$_{error}$ in the placements used for testing, we add these placements to our set of placements used for training $\mathbb{P}_{tr}$, and remove these from $\mathbb{P}$. The while loop exits when either all the remaining placements in $\mathbb{P}$ have been added to $\mathbb{P}_{tr}$ or when f_m achieves $e \leq$ UB$_{error}$.[10] Using f_m, we can now interpolate or extrapolate values of the parameter. That is, for any new value of clock period, utilization, aspect ratio and BEOL stack, we calculate the ratios of clock periods, utilization, aspect ratio, #H and #V tracks, etc. and use f_m to estimate the values of the parameters. We train one model f_m for each parameter described in Section III-A.

To train each f_m, we use MARS [5] and Support Vector Machine (SVM) [5] using a Radial Basis Function (RBF) kernel [5] and combine their responses using weights determined by least-squares regression, and train a model for each parameter. Once we have obtained a set of estimated values of parameters using the above methodology, we will use these as our modeling parameters to predict routability as described in Section III-A, and predict Pareto frontiers as described in Section IV-C.

C. Modeling methodology

The goal of modeling is to predict whether a BEOL stack-specific placement is routable (for a given router). We classify an implementation to be unroutable when the number of design rule violations at the post-route stage (or, the number of DRCs) obtained from commercial place-and-route (P&R) tools is ≥ 50. When the number of violations is < 50, these violations are typically fixed by designers manually. Our model for each technology is a binary classifier developed using the SVM algorithm using a RBF kernel. We use the label "+1" when a design is routable and the label "-1" when an implementation is unroutable. We use five-fold cross-validation to generalize our models.[11]

IV. Experimental Setup and Results

We now describe our experimental setup, design of experiments (DoE), and present our results. In Section IV-A, we describe our DoE for three foundry technologies and designs that we use to train our models. To test our models, we use new designs that the training data has not seen. We describe the DoE of our test dataset and its applications to our models in Sections IV-B and IV-C.

A. Model construction

We conduct our experiments on multiple designs: *aes_cipher_top* and *vga_enh_top* from OpenCores [39]; *aes_x2* and *aes_x3* created by instantiating and stitching two and three *aes_cipher_top* designs respectively; and ARM *Cortex M0* core, *leon3mp* core, *DW_dct* and *jpeg_x5* created by instantiating and stitching five *jpeg_encoder* designs [39]. We synthesize these designs using Synopsys *Design Compiler vI-2013.12-SP3* [40]. Table I shows the DoE used to obtain ground truth for modeling of designs with 28nm FDSOI eight-track (8T), 28nm LP 12-track (12T) and 45nm GS nine-track (9T) technology libraries. We use these data points to train our models (one model for each technology). We create custom LEF files with eight, seven, six, five and four metal layers, all having $1\times$ pitch as the Mx layer. We run P&R using Cadence *Innovus v15.2* [38]. We perform denoising [9] [11] by executing P&R for each point in our DoE six times, i.e., by perturbing each P&R clock periods {-5, +0, +5}ps with an utilization value, and by perturbing each utilization value by {-0.05, +0.00, +0.05}% with a clock period.[12] We classify a placement as unroutable when all the six runs indicate that the #DRCs is ≥ 50. We then use custom scripts in *Tcl* to extract the parameters described in Section III-A as follows.

- We use grids to divide the entire layout. Note that the number of grids varies with designs and utilizations.
- To obtain pin density per grid, we count the number of pins in each grid and divide it by the grid area.
- To obtain the minimum proximity of pins, we calculate half-perimeter wirelength (HPWL) of all pairs of pins within a grid. We then use the minimum HPWL of these values as minimum proximity.
- To obtain the number of complex cells, we obtain all cells within the bounding box of a grid and their cell masters. We then count the cells whose master names are either *AOI*, *OAI*, three-input *XOR* and *XNOR*, or *MUX*.
- To obtain the number of buried nets, we count nets that have all of their pins within a grid.
- To obtain the number of edges, we count the number of incoming incident edges to pins within a grid and the number of outgoing edges from pins within a grid. We then add the number of incoming and outgoing edges.
- To obtain the placement-based Rent parameter, we use the *RentCon* tool [41] with 15 tracks $\times$ 15 tracks grid size, and shifting of evaluation windows by $\frac{1}{4}\times$ the size of the grid. Thus, 16 grids over the layout region are used for evaluation of the placement-based Rent parameter.
- To obtain the worst signal transition time of all pins within a grid, we obtain all pins over all critical paths that are within a grid in the worst corner. We then take the worst signal transition time of all these pins using the *get_property* command.
- To obtain the worst setup WNS within a grid, we check all pins within a grid and obtain WNS of the paths to which these pins belong. We then take the worst (i.e., minimum value) WNS as our parameter.
- We obtain statistical information, i.e., max and min values by considering the maximum and minimum values of our parameters across all grids. We calculate the coefficient of variation by dividing the standard deviation by the mean value of parameters (across all grids).

We now explain our choice of grid sizes. We use grid sizes of 45 tracks $\times$ 45 tracks because these are multiples of gcell sizes (15 tracks $\times$ 15 tracks) used by our P&R tool. We have tried grid sizes of 15 tracks $\times$ 15 tracks, 30 tracks $\times$ 30 tracks and 90 tracks $\times$ 90 tracks as well. Small grid sizes hide the correlation between #DRCs and our parameters because multiple neighboring gcells can together cause DRC violations. Large grid sizes blur the differences between various utilizations, i.e., do not capture local hotspots. We use 45 tracks $\times$ 45 tracks because these grids show correlation with #DRCs and at the same time does not blur differences across utilizations. Figures 4(a) and (b) show that by using grid sizes of 15 tracks $\times$ 15 tracks the correlation between pin density and #DRCs is not apparent in 28nm FDSOI. By using grid sizes of 45 tracks $\times$ 45 tracks, the correlations between #DRCs and pin density become more apparent. We compare the coefficient of determination R^2 in both figures and observe that Figure 4(b) has larger R^2 value than Figure 4(a). Similarly, Figures

[10]Note that it is possible that we use all the placement solutions to train a model for a parameter, but error is $>$ UB$_{error}$.

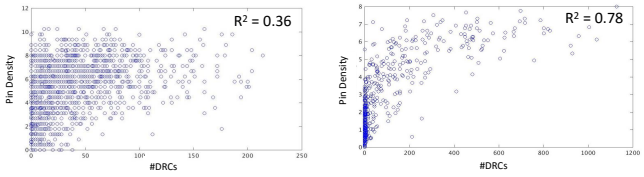
[11]Interested readers can find further details of kernel-based SVM classification in [5].

[12]On a 2.6GHz Intel Xeon E5-2690 processor, placement takes around 1.5 hours (on average) for *aes_cipher_top*, 2.4 hours (on average) for *aes_x2* and 3.3 hours (on average) for *aes_x3* with two cores. On average, routing executes in 2.5, 3.1 and 5 hours, respectively for these designs when the placement is routable, and takes around 3.5, 5.1 and 6.3 hours, respectively when the placement is unroutable. We choose clock periods so that the implementations meet timing at the post-route stage. Note that out of six runs, one of the runs that uses the utilization and clock period values from the DoE is duplicated.

TABLE I
DESIGN OF EXPERIMENTS USED TO OBTAIN GROUND TRUTH FOR TRAINING OUR MODELS.

		aes_cipher_top	*aes_x2*	*aes_x3*
Syn Clk Period (ns)	28FDSOI, 8T	0.6	0.6	0.6
	28LP, 12T	0.6	0.6	–
	45GS, 9T	1.0	1.0	–
#Instances	28FDSOI, 8T	11461	22770	34834
	28LP, 12T	11783	24744	–
	45GS, 9T	13601	28562	–
#FFs	28FDSOI, 8T	530	1060	1590
	28LP, 12T	530	1060	–
	45GS, 9T	530	1060	–
P&R Clk Period (ns)	28FDSOI, 8T	{0.6, 0.9, 1.1}	{0.6, 0.9, 1.1}	{0.6, 0.9, 1.1}
	28LP, 12T	{0.6, 0.9, 1.1}	{0.6, 0.9, 1.1}	–
	45GS, 9T	{1.0, 1.2, 1.8}	{1.0, 1.2, 1.8}	–
Util (%)	28FDSOI, 8T	{70, . . ., 86}	{70, . . ., 86}	{70, . . ., 86}
	28LP, 12T	{70, . . ., 86}	{70, . . ., 86}	–
	45GS, 9T	{70, . . ., 86}	{70, . . ., 86}	–
Aspect Ratio	28FDSOI, 8T	{1.0, 1.3, 1.8}	{1.0, 1.3, 1.8}	{1.0, 1.3, 1.8}
	28LP, 12T	{1.0, 1.3, 1.8}	{1.0, 1.3, 1.8}	–
	45GS, 9T	{1.0, 1.3, 1.8}	{1.0, 1.3, 1.8}	–
#Metal Layers	28FDSOI, 8T	{6, 5, 4}	{6, 5, 4}	{6, 5, 4}
	28LP, 12T	{6, 5, 4}	{6, 5, 4}	–
	45GS, 9T	{6, 5, 4}	{6, 5, 4}	–
Grid Size (#tracks)	28FDSOI, 8T	45 × 45	45 × 45	45 × 45
	28LP, 12T	45 × 45	45 × 45	–
	45GS, 9T	45 × 45	45 × 45	–
#Routable	28FDSOI, 8T	306	306	294
	28LP, 12T	437	424	–
	45GS, 9T	317	311	–
#Unroutable	28FDSOI, 8T	153	153	165
	28LP, 12T	22	35	–
	45GS, 9T	142	148	–

5(a) and (b) show that by using grid sizes of 15 tracks × 15 tracks, the correlation between #DRCs and the number of complex cells is not apparent in 28nm FDSOI. By using grid sizes of 45 tracks × 45 tracks, the correlation between #DRCs and the number of complex cells becomes more apparent. We compare the coefficient of determination R^2 in both figures and observe that Figure 5(b) has larger R^2 value than Figure 5(a).[13]



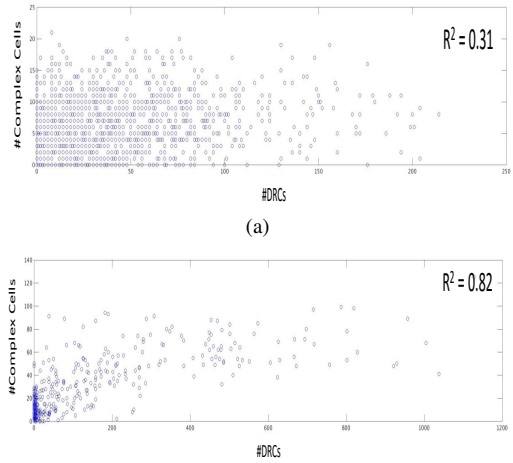
(a) (b)

Fig. 4. Correlations of #DRCs with pin density at 28nm FDSOI. The size of grids is set to (a) 15 tracks × 15 tracks, and (b) 45 tracks × 45 tracks.

In total, we use 1377 data points for training our model in 28nm FDSOI, out of which 906 data points are from routable implementations and the remaining 471 are from unroutable implementations. In 28nm LP, we use a total of 918 data points for training, out of which 861 are from routable and 25 are from unroutable implementations. In 45nm GS, we use a total of 918 data points for training, out of which 618 are from routable and 290 are from unroutable implementations. We train our models using *Matlab* scripts. We conduct two experiments to demonstrate applications of our models, as follows.

- **Experiment 1.** To determine whether a placement is routable using our models on "unseen" data points from new designs across various technologies.
- **Experiment 2.** To predict Pareto frontiers of utilization, aspect ratio and number of metal layers at iso-performance.

[13]Commercial P&R tools use pitch of the M2 layer as track size. In our technology libraries, 28nm FDSOI and LP technologies have M2 pitch of 0.1μm, and 45nm GS technology has M2 pitch of 0.14μm. Therefore, our grid sizes are 4.5μm × 4.5μm for 28nm FDSOI and LP, and 6.3μm × 6.3μm for 45nm GS.



(a)

(b)

Fig. 5. Correlations of #DRCs with the number of complex cells at 28nm FDSOI. The size of grids is set to (a) 15 tracks × 15 tracks, and (b) 45 tracks × 45 tracks.

B. Results of Experiment 1

Experiment 1 uses our models to predict whether a placement is routable for a fixed utilization, aspect ratio, clock period and BEOL stack. We test our models on completely unseen data points from new designs. Table II shows the DoE used to obtain ground truth for testing our models in 28nm FDSOI, 28nm LP and 45nm GS technologies. We use five *new* designs such as *Cortex M0* (*Cortex*), *DW_dct* (*dct*), *vga_enh_top* (*vga*), *jpeg_x5* and *leon3mp*,[14] as well as eight- and seven-layer BEOL stacks that we did not use for training. These make the classification problem more difficult and help assess whether our models are generalizable and scalable. We eliminate tool noise in the same manner as described for the training dataset, i.e., we execute six P&R runs for each point in the DoE by perturbing the clock period (by keeping utilization fixed) and utilization (by keeping clock period fixed) values.[15]

We use standard classification metrics to assess our training and test classifications such as accuracy, precision, recall and negative predictive value (NPV). We use confusion matrices to illustrate classification performed by our models on training and test data points. Accuracy is defined as the ratio of the sum of true positives (TPs) and true negatives (TNs) to the sum of all data points used for classification (either for training or testing). Precision is defined as the ratio of TPs to the sum of TPs and false positives (FPs); recall is defined as the ratio of TPs to the sum of TPs and false negatives (FNs); and NPV is defined as the ratio of TNs to the sum of TNs and FNs.

Table III shows the confusion matrices of our predictions in 28nm FDSOI, 28nm LP and 45nm GS by using parameters listed in Section III-A. For each technology, we use the data points from Table I for training, and the data points from Table II for testing. "True" refers to a placement being routable, that is, having a label "+1", and "False" refers to a placement being unroutable, that is, having a label "-1".

Table IV shows error metrics (i.e., accuracy, precision, recall and NPV) for training and test datasets in 28nm FDSOI, 28nm LP and 45nm GS. We observe that the accuracy values are $\geq$ 85.9% in the test dataset, that is, our models are able to accurately classify placements as routable and generalizes to unseen data points. The accuracy is 90% for 28nm LP because the prediction problem is less difficult than the other two technologies as the number of unroutable

[14]We use a variety of open-source designs to represent blocks in complex real-world SoCs since industrial designs were not available to us.

[15]Note that we do not use all six runs for training or testing. These runs are for denoising and generate only one data point. The data point is unroutable if each of the six runs contains #DRCs $\geq$ 50; otherwise, it is routable. A designer can choose to skip the denoising step and execute only one run.

TABLE II
DESIGN OF EXPERIMENTS USED TO OBTAIN GROUND TRUTH FOR TESTING OUR MODELS.

		Cortex	*jpeg_x5*	*leon3mp*	*dct*	*vga*
Syn Clk Period (ns)	28FDSOI, 8T	0.8	1.0	1.0	0.8	0.6
	28LP, 12T	0.8	1.0	–	1.0	0.7
	45GS, 9T	1.2	1.5	–	–	–
#Instances	28FDSOI, 8T	9282	111342	442734	11987	71080
	28LP, 12T	9380	111463	–	12015	71096
	45GS, 9T	13601	168310	–	–	–
#FFs	28FDSOI, 8T	840	23560	108817	798	17057
	28LP, 12T	840	23560	–	798	17057
	45GS, 9T	840	23560	–	–	–
P&R Clk Period (ns)	28FDSOI, 8T	{0.8, 1.0, 1.5, 2.0}	{1.3, 1.5}	{1.5, 2.0}	{0.9, 1.2}	{0.7, 1.2}
	28LP, 12T	{0.8, 1.0, 1.5, 2.0}	{1.3, 1.5}	–	{1.0, 1.2}	{0.8, 1.3}
	45GS, 9T	{1.5, 2.0, 2.2}	{1.5, 2.0}	–	–	–
Util (%)	28FDSOI, 8T	76, . . ., 90	76, . . ., 90	76, . . ., 90	76, . . ., 90	76, . . ., 90
	28LP, 12T	76, . . ., 90	76, . . ., 90	–	76, . . ., 90	76, . . ., 90
	45GS, 9T	76, . . ., 90	76, . . ., 90	–	–	–
Aspect Ratio	28FDSOI, 8T	{1.0, 1.8, 2.0, 2.2}	{1.0, 1.2, 1.5, 2.1}	{1.0, 1.2}	{1.0, 1.5, 2.0}	{1.0, 1.5, 2.0}
	28LP, 12T	{1.0, 1.5, 1.7}	{1.0, 1.1, 1.6}	–	{1.0, 1.5, 2.0}	{1.0, 1.5, 2.0}
	45GS, 9T	{1.0, 1.5, 1.7}	{1.0, 1.3, 2.0}	–	–	–
#Metal Layers	28FDSOI, 8T	{8, . . ., 4}	{8, . . ., 4}	{8, . . ., 4}	{6, 5, 4}	{6, 5, 4}
	28LP, 12T	{6, 5, 4}	{6, 5, 4}	–	{6, 5, 4}	{6, 5, 4}
	45GS, 9T	{6, 5, 4}	{6, 5, 4}	–	–	–
Grid Size (#tracks)	28FDSOI, 8T	45 × 45	45 × 45	45 × 45	45 × 45	45 × 45
	28LP, 12T	45 × 45	45 × 45	–	45 × 45	45 × 45
	45GS, 9T	45 × 45	45 × 45	–	–	–
#Routable	28FDSOI, 8T	900	502	195	180	177
	28LP, 12T	508	246	–	240	235
	45GS, 9T	277	195	–	–	–
#Unroutable	28FDSOI, 8T	300	98	105	90	93
	28LP, 12T	32	24	–	30	35
	45GS, 9T	128	75	–	–	–

placements are few. Across all technologies and designs, our precision is $\geq 91\%$ and recall is $\geq 86\%$. That is, there are few false positives and few false negatives in the classification results of the test dataset. In 28nm LP, only 10% of the data points are unroutable, that is, the training data is biased towards the routable label of "+1". Therefore, the modeling problem is relatively easy. However, our 28nm LP model does not overfit the routable data points and is able to identify 23 out of 121 unroutable placements correctly. Even though the sizes of our test datasets are up to $\sim 1.9\times$ the sizes of our training datasets, our classifications are accurate across technologies. For example, in 45nm GS, accuracy degrades from 97.0% in the training dataset to 85.9% in the test dataset.[16]

Tables V and VI show the confusion matrices and error metrics by using only congestion maps from the placement that are typically used by physical design engineers to predict routability. NPV is a measure of how accurately a model can predict unroutable placements. In other words, NPV measures the ratio of placements that are truly unroutable to the placements that are predicted to be unroutable.[17] The overhead of incorrectly classifying a routable placement as unroutable is high, as design turnaround time increases and quality of results worsens. In Section I-A, we illustrated the poor correlation of #DRCs with placement congestion maps; now we quantify the error across technologies and designs. We observe that accuracy for 28nm LP placements is 77.3% with NPV of 20.8% in the test dataset, whereas by using our new parameters, the accuracy is 90.5% and NPV is 48.2% for the same test dataset.[18]

[16]To test robustness of our conclusions, we performed modeling by interchanging the training and test datasets. We train our models using DoE data points from Table II and test the models on data points from Table I. In the training dataset, the worst-case differences in accuracy is 2.4%, precision is 1.1%, recall is 1.6% and NPV is 4.4%, across all technologies. In the test dataset, the worst-case differences in accuracy is 3.3%, precision is 1.6%, recall is 2.4% and NPV is 9.7%.

[17]For example, if every placement is always predicted to be routable, then NPV will be zero.

[18]We experimentally demonstrate that it is important to include #H, and #V tracks as parameters (to represent BEOL stacks) in our models. We have conducted experiments in 28nm FDSOI by removing these parameters and observe that the modeling accuracy in the test dataset reduces from 87.2% to 77.5%.

TABLE III
CONFUSION MATRICES FOR ROUTABILITY PREDICTION FOR TRAINING AND TEST DATASETS.

			Training			Testing		
			Actual				Actual	
			True	False			True	False
28nm FDSOI	Pred	True	869	11		True	1790	173
		False	37	460		False	164	513
28nm LP	Pred	True	829	1		True	1124	23
		False	32	56		False	105	98
45nm GS	Pred	True	612	11		True	406	29
		False	16	279		False	66	174

TABLE IV
CLASSIFICATION ERROR METRICS FOR TRAINING AND TEST DATASETS.

	Dataset	Accuracy (%)	Precision (%)	Recall (%)	NPV (%)
28nm FDSOI	Training	96.5	98.8	95.9	92.5
	Testing	87.2	91.9	91.6	75.8
28nm LP	Training	96.4	99.8	96.3	63.6
	Testing	90.5	98.0	91.5	48.2
45nm GS	Training	97.0	98.2	97.4	94.6
	Testing	85.9	93.3	86.0	72.5

TABLE V
CONFUSION MATRICES FOR ROUTABILITY PREDICTION BY USING CONGESTION MAP ONLY FOR TRAINING AND TEST DATASETS.

			Training			Testing		
			Actual				Actual	
			True	False			True	False
28nm FDSOI	Pred	True	833	66		True	1427	298
		False	73	405		False	527	388
28nm LP	Pred	True	790	11		True	978	55
		False	71	46		False	251	66
45nm GS	Pred	True	586	47		True	334	89
		False	42	243		False	138	114

TABLE VI
CLASSIFICATION ERROR METRICS BY USING CONGESTION MAP ONLY FOR TRAINING AND TEST DATASETS.

	Dataset	Accuracy (%)	Precision (%)	Recall (%)	NPV (%)
28nm FDSOI	Training	89.9	92.6	91.9	84.7
	Testing	68.8	82.7	73.0	42.4
28nm LP	Training	91.1	98.6	91.7	39.3
	Testing	77.3	94.7	79.6	20.8
45nm GS	Training	90.3	92.6	93.3	85.3
	Testing	66.3	78.9	70.8	45.2

C. Results of Experiment 2

In this experiment, our goal is to determine the iso-performance Pareto frontiers of utilization, aspect ratio and number of metal layers for various designs using our models. Our models can predict "iso-performance" because we comprehend clock period and timing-related parameters in our modeling, and our results are actually different from "performance-oblivious" models. We are given only a few placements, so we must interpolate and extrapolate our modeling parameters from these placements to predict the Pareto frontiers. This is very challenging because the metrics do not scale in a known manner (e.g., unimodal, linear, etc.) when utilization, aspect ratio and the BEOL stack is changed. Sometimes the P&R tools stop fixing timing or congestion violations at the placement stage when the utilization is very tight or the BEOL stack has insufficient number of metal layers. To overcome these challenges, we devise an interpolation and extrapolation method as described in Section III-B using machine learning.

Our test dataset in each technology contains around 100–300 implementations of *Cortex* and *jpeg_x5* designs that span different utilizations, aspect ratio values and BEOL stacks. We choose 20 of these placements per design that are implemented with the smallest clock period for these designs from Table II.[19] We then execute our method in Section III-B. Obtaining 20 placements is inexpensive – especially, relative to the cost of a failed routing job or wasting area and/or wafer cost – from both CPU and wall time standpoints.[20]

We set the error upper bound UB_{error} for each metric to be 20%. We use cubic splines for the MARS technique and use grid search to determine the best values of hyperparameters (e.g., the SVM regularization hyperparameter C, error margin ξ and RBF weight for each radius γ [5]) for SVM with RBF kernel. We train one model for each parameter, e.g., {max, average} $\times$ {pin density, #complex cells, sum of incoming and outgoing edges}, etc. as described in Section III-A. Figures 6(a) and (b) compare average pin density and average #complex cells respectively for the 20 placements of *jpeg_x5* in 28nm FDSOI when using our method to interpolate or extrapolate parameter values and actual values obtained from the placements. We then create a test dataset using estimates from the models of each parameter, and use our models (developed using the DoE for training dataset in Table I) to predict routability.

Figures 7(a) and (b) show the predicted Pareto frontiers of #metal layers, utilizations and aspect ratios for *Cortex* and *jpeg_x5* respectively in 28nm FDSOI. Figures 8(a) and (b) show the predicted Pareto frontiers of #metal layers, utilizations and aspect ratios for *Cortex* and *jpeg_x5* respectively in 45nm GS. Figures 9(a) and (b) show the ground truth Pareto frontiers of #metal layers, utilizations and aspect ratios, respectively for *Cortex* in 28nm FDSOI and *jpeg_x5* in 45nm GS.[21] From Figures 7(a) and 9(a), we observe that in 28nm FDSOI, *Cortex* is routable with five metal layers when aspect ratio is 1.8 and utilization is 79%, but our model predicts that the maximum utilization in 78% (i.e., 79% requires six metal layers). Similarly, from Figures 8(b) and 9(b), in 45nm GS *jpeg_x5* is routable with four metal layers when aspect ratio is 1.5 and utilization is 77%, but our model predicts that no placement of *jpeg_x5* is routable with four metal layers at aspect ratio 1.5. Across three foundry technologies and two designs (that were not used for training), our predictions of maximum achievable utilization are within 2% of the maximum achievable utilization value in the ground truth.

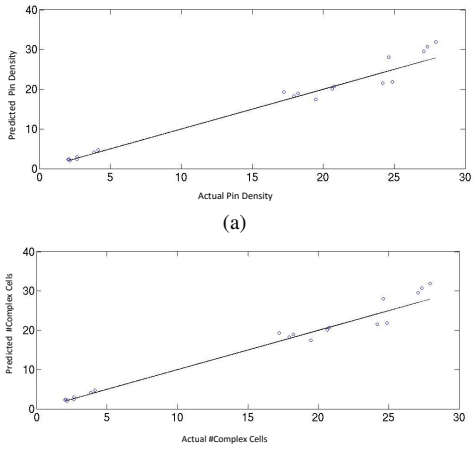


Fig. 6. Prediction accuracy of our interpolation / extrapolation method in 28nm FDSOI for *jpeg_x5* for average (a) pin density, and (b) #complex cells.

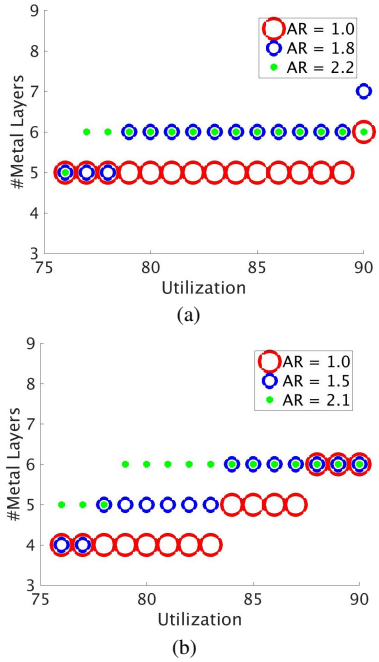


Fig. 7. Pareto frontiers of #Metal Layers (y-axis) versus utilization (x-axis) for multiple aspect ratios using our models in 28nm FDSOI: (a) *Cortex*, and (b) *jpeg_x5*.

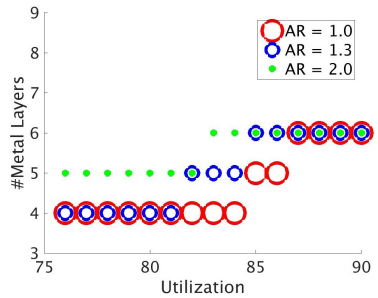
V. CONCLUSIONS

Efficient exploration of the space of utilization, aspect ratio and BEOL stack at iso-performance is very important for design

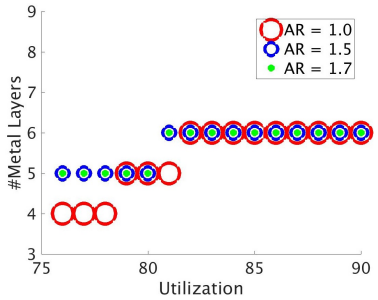
[19]For example, in 28nm FDSOI we choose utilizations {80, 82, 83, 84, 85}% for both designs; the corresponding numbers of metal layers for these utilizations are {4, 4, 5, 5, 6}. For *Cortex*, we use aspect ratios {1.0, 1.8, 2.0, 2.2}, and for *jpeg_x5* we use aspect ratios {1.0, 1.2, 1.5, 2.1}.

[20]Only the ground truth of the Pareto frontiers is obtained from the actual routing information of these placements.

[21]The two designs show limited value from the M5 layer because beyond 82% utilization, both horizontal and vertical routing tracks are required for routability. Adding only M5 does not cure the routability issues.

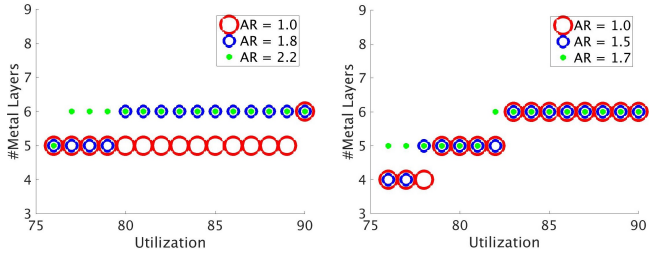


(a)



(b)

Fig. 8. Pareto frontiers of #Metal Layers (y-axis) versus utilization (x-axis) for multiple aspect ratios in 45nm GS: (a) *Cortex*, and (b) *jpeg_x5*.



(a) (b)

Fig. 9. Ground truth Pareto frontiers of #Metal Layers (y-axis) versus utilization (x-axis) for multiple aspect ratios of (a) *Cortex* in 28nm FDSOI, and (b) *jpeg_x5* in 45nm GS.

turnaround time and to achieve good quality of results. Currently, physical design engineers use congestion maps of P&R tools from the placement stage to predict routability as measured by the #DRCs. However, our experimental results indicate that these maps can sometimes be misleading and inaccurate in predicting routability. We present new modeling parameters that enable us to analyze local hotspots in a placement and achieve accurate predictions of routability. We also present a new method of using only a few placements to predict (using our models) the Pareto frontiers of utilizations, aspect ratios and BEOL stacks at iso-performance. Our experimental results indicate that our predictions are pessimistic by 2% of the maximum achievable utilization across three different foundry technologies and two designs (that were not used for training). Overall, our classification accuracies are 87.0% in 28nm FDSOI, 90.4% in 28nm LP and 85.9% in 45nm GS. We achieve significant improvements as compared to using only congestion maps; classification accuracies by using only congestion maps are 61.7% in 28nm FDSOI, 73.5% in 28nm LP and 66.3% in 45nm GS. Our ongoing works include (i) predicting timing and routability with nonuniform BEOL stacks, (ii) adding confidence intervals of our routability predictions, and (iii) extending our models to predict routability of 3DIC placements.

REFERENCES

[1] A. Andreev, I. Pavisic and P. Raspopovic, "Multi-Layer Assignment", *U.S. Patent* No. 6,182,272, January 2001.

[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] X. Dong, J. Zhao and Y. Xie, "Fabrication Cost Analysis and Cost-Aware Design Space Exploration for 3D-ICs", *IEEE Trans. on CAD* 29(12) (2010), pp. 1959-1972.

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

[6] 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.

[7] 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.

[8] 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.

[9] K. Jeong and A. B. Kahng, "Methodology from Chaos in IC Implementation", *Proc. ISQED*, 2010, pp. 885-892.

[10] 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.

[11] A. B. Kahng and S. Mantik, "Measurement of Inherent Noise in EDA Tools", *Proc. ISQED*, 2002, pp. 206-211.

[12] A. B. Kahng and D. Stroobandt, "Wiring Layer Assignments with Consistent Stage Delays", *Proc. SLIP*, 2000, pp. 115-122.

[13] A. B. Kahng, S. Mantik and D. Stroobandt, "Requirements for Models of Achievable Routing", *Proc. ISPD*, 2000, pp. 4-11.

[14] A. B. Kahng, S. Mantik and D. Stroobandt, "Toward Accurate Models of Achievable Routing", *IEEE Trans. on CAD* 20(5) (2001), pp. 648-659.

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

[16] 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.

[17] M.-S. Lin, "Top Layers of Metal for High Performance ICs", *U.S. Patent* No. 6,383,916, May 2002.

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

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

[20] 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.

[21] G. Palermo, C. Silvano and V. Zaccaria, "ReSPIR: A Response Surface-Based Pareto Iterative Refinement for Application-Specific Design Space Exploration", *IEEE Trans. on CAD* 28(12) (2009), pp. 1816-1829.

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

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

[24] 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.

[25] H. Shojaei, A. Davoodi and J. T. Linderoth, "Congestion Analysis for Global Routing via Integer Programming", *Proc. ICCAD*, 2011, pp. 256-162.

[26] H. Shojaei, A. Davoodi and J. T. Linderoth, "Planning for Local Net Congestion in Global Routing", *Proc. ISPD*, 2013, pp. 85-92.

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

[28] D. Stroobandt, "Recent Advances in System-Level Interconnect Prediction", *IEEE Circuits and Systems Newsletter* 11 (2000), pp. 4-20.

[29] 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.

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

[31] R. Venkatesan, J. A. Davis, K. A. Bowman and J. D. Meindl, "Optimal n-tier Multilevel Interconnect Architectures for Gigascale Integration (GSI)", *IEEE Trans. on VLSI* 9(6) (2001), pp. 899-912.

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

[33] Y. Wei1, C. Sze, N. Viswanathan, Z. Li, C. J. Alpert, L. Reddy, A. D. Huber, G. E. Tellez, D. Keller and S. S. Sapatnekar, "GLARE: Global and Local Wiring Aware Routability Evaluation", *Proc. DAC*, 2012, pp. 768-773.

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

[35] J. Westra and P. Groeneveld, "Is Probabilistic Congestion Estimation Worthwhile?", *Proc. SLIP*, 2005, pp. 99-106.

[36] 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.

[37] 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.

[38] Cadence Innovus User Guide, http://www.cadence.com

[39] OpenCores: Open Source IP-Cores, http://www.opencores.org

[40] Synopsys User Guide, http://www.synopsys.com

[41] Rent Parameter Evaluation Using Different Methods v2.2-2008. http://vlsicad.ucsd.edu/WLD/RentCon.pdf