# PROBE: A Placement, Routing, Back-End-of-Line Measurement Utility

Alex Kahng, Andrew B. Kahng, *Fellow, IEEE*, Hyein Lee, *Student Member, IEEE*, and Jiajia Li, *Student Member, IEEE*

***Abstract*—In advanced technology nodes, correctly choosing among available back-end-of-line (BEOL) stack options is important to meet stringent design quality of results requirements. However, it is nontrivial to evaluate BEOL stack options since the routing outcomes highly depend on the input design (e.g., netlist, placement, etc.). In this paper, we propose a systematic framework to measure routing capacity of a BEOL stack as well as inherent capability of routers. Based on our experimental results, we observe consistent results across mesh-like placement and placements from various placers. Also, our proposed framework enables new insights into important questions regarding BEOL stack options. Using our framework, we further study the relation between the routing hotspot size and routing failure empirically. Lastly, we present an analytical study based on exponentiation of a Markov transition matrix about the impact of design size on routing failure.**

***Index Terms*—Back-end-of-line stack options, benchmark, placement, routability, routing, transition matrix.**

## I. Introduction

PARTICULARLY in advanced technology nodes, interconnects significantly affect the power, area, and performance of integrated circuits. Requirements of high integration density and performance, as well as patterning technology, design rules, and cost implications, make it imperative to determine good back-end-of-line (BEOL) stack options for sub-22-nm technology nodes. Yet, to our knowledge, there has been no systematic framework to measure the routing capacity of a BEOL stack for a given router, or for a combination of router and placement, particularly in a tool-agnostic, intrinsic sense. Moreover, measurement of the routing capacity is nontrivial due to the gear ratio of metal pitches, via blockages and many other factors. A common methodology to measure the routing capacity of a given BEOL stack will simply perform routing on instances of placed designs (e.g., placement solutions for different netlists, possibly with different utilization, and aspect ratio configurations). However, the routing outcomes will highly depend on the input netlist, placement solution quality and technology (e.g., standard cell architecture and track height, BEOL design rules). It has not been previously contemplated in the literature that design-technology co-optimization might be supportable with a "quasi-universal" routing capacity rank-ordering of alternative BEOL stacks, where this rank-ordering is, empirically, general across different netlists and placement solutions.[1]

In this paper, we propose a general framework to measure the routing capacity of a BEOL stack with a given router. Based on the gradual perturbation of an initial placement of a netlist topology (e.g., a 2-D mesh), our framework is able to obtain a ranking of alternative BEOL stacks according to routing capacities, for a given router. Quite interestingly, we further experimentally confirm that the measurement (i.e., routing capacity ranking of BEOL stacks) based on mesh-like placements is largely consistent with those based on placements generated by a commercial P&R tool within a standard SP&R flow (see Section IV-D below). Moreover, our results appear, empirically, to be robust across various mesh-like placements with different characteristics (e.g., types of cells, track height, number of instances, row utilization, pin alignment, and 1-D/2-D routing).

Our proposed framework enables new insights into important questions about BEOL stack options, e.g., "In terms of routing capacity, one P100 (100 nm pitch) layer is equivalent to how many P150 layers, for a given router?" and "On which layer(s) is it most valuable to enable bidirectional routing?" We also study the relationship between routing hotspot size,[2] placement quality (i.e., indicated by the amount of perturbation away from the optimal mesh-like placement), and the number of post-route design rule violations (#DRVs). Last, we present an analytical study based on exponentiation of a Markov transition matrix to demonstrate how, with the same placement density and quality, a larger design is more likely to experience routing failures, an observation which is also supported by our empirical data. Our contributions are summarized as follows.

1) We propose a systematic framework to assess routing capacity of BEOL stack options for a given router.
2) We propose a novel metric, $K$, that intuitively corresponds to a systematic worsening of placement quality for a given standard-cell netlist—achieved by iteratively swapping pairs of adjacently placed cell instances—starting from a given initial placement.

Manuscript received October 5, 2016; revised February 15, 2017 and June 3, 2017; accepted August 2, 2017. Date of publication September 7, 2017; date of current version June 18, 2018. This paper was recommended by Associate Editor L. Behjat. *(Corresponding author: Hyein Lee.)*

A. Kahng is with Harvard University, Cambridge, MA 02138 USA (e-mail: alexkahng@college.harvard.edu).

A. B. Kahng is with the Department of Computer Science and Engineering, University of California at San Diego, San Diego, CA 92093 USA, and also with the Department of Electrical and Computer Engineering, University of California at San Diego, San Diego, CA 92093 USA (e-mail: abk@ucsd.edu).

H. Lee and J. Li are with the Department of Electrical and Computer Engineering, University of California at San Diego, San Diego, CA 92093 USA (e-mail: hyeinlee@ucsd.edu; jil150@ucsd.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCAD.2017.2750072

[1] In this paper, we use the term "quasi-universal" to refer to rank-orderings of BEOL stacks that are empirically highly correlated (e.g., correlation coefficient > 0.9) across different designs and/or design enablements (e.g., different netlists, placers, and routers).

[2] Following common usage in the IC and EDA fields, we use the term "hotspot" to refer to a local window of the layout that is spatially co-located with routing failure.

3) By sweeping $K$, we determine $K_{\text{th}}$ as the minimum $K$ that results in routing failure.[3] We apply $K_{\text{th}}$ in our routing capacity assessment.
4) We empirically demonstrate a quasi-universal rank-ordering of $K_{\text{th}}$ for BEOL stack routing capacity across placements generated by a commercial P&R tool and simple mesh-like placements.
5) Our framework can be used to evaluate routers.
6) We study the size and placement quality of a routing hotspot and their correlation with #DRVs after routing.
7) We perform both an analytical study and an experimental study to demonstrate how, with the same placement density and quality, a larger design is more vulnerable to routing failures.
8) We suggest the possibility of a technology-dependent sweetspot of block size (trading off overheads of design decomposition against overheads of routing difficulty).
9) A utility that implements the above framework for arbitrary enablement (.lef, .lib) is available at [39].

The mesh-like placements in our framework cannot perfectly represent real-life placements which comprehend timing, signal power integrity, and other conflicting objectives and constraints that circuit designers face. However, rather than generating testcases that mimic real designs, our approach enables systematic generation of benchmarks with gradually increasing routing difficulty. This enables a principled rank-ordering of the respective routing capacities of different BEOL stacks, which to our knowledge cannot be straightforwardly achieved using specific real design instances. This paper empirically shows that dependence on specific design instances may be avoidable, which would help resolve a long-standing controversy regarding the utility of artificial versus real testcases. In sum, our results suggest that the mesh-like placements may be usable as proxies of real-life placements, at least to rank-order BEOL stack options in terms of routability. The routability-oriented rank-ordering can help reduce the number of configurations that must be considered as product teams address the BEOL stack optimization problem.

## II. Related Works

In this section, we review the previous literature on 1) benchmark construction and 2) routability estimation.

### A. Benchmark Studies

*1) Artificial Benchmarks:* To evaluate the performance of VLSI optimizations, a number of artificial benchmark generation approaches have been proposed in previous literature. These include *circ/gen* [14], *gnl* [30] and the work of [7]. In the interests of realism, *circ/gen* [14] measures characteristics (e.g., circuit size, number of IOs, path depth, and fanout distribution) of existing circuits and uses these characteristics as constraints in its synthetic circuit generation. As an extension to graph-based benchmark generation (which only considers graph-based properties such as rent parameter [20] and net degree distribution), *gnl* [30] considers functional information, in that it uses a specified component library and avoids combinational loops. Darnauer and Dai [7] proposed a method for generating random circuits for routability measurement. The input parameters of their benchmark generation include design sizes, rent parameter, and number of IOs.

[3]In this paper, if a routed design has #DRVs $>$ 150, we consider it as a design with routing failure.

Several methodologies produce instances with *known optimal solutions*, which enables quantification of a heuristic optimization's suboptimality. For placement optimization, the *PEKO* benchmark generator [5] provides a netlist (with user-specified number of placeable modules and net degree distribution) as well as a constructive placement solution with known minimum wirelength. Attributed to Boese in [10], *PEKU* [6] further improves the realism by including nonlocal nets, while generating instances with known upper bounds on optimal wirelength. In a similar spirit, there are also gate-sizing-oriented benchmarks with known optimal solutions. The work of [15] generates benchmark circuits (called *eye-charts*) of arbitrary size along with a method to compute their optimal solutions using dynamic programming. The work of [16] generates more realistic benchmarks (by comprehending path depth and fanin/fanout distributions) with known optimal solutions for gate sizing problems.

*2) Realistic Benchmarks:* Artificial benchmarks with known optimal solutions can help quantify suboptimality of heuristics for NP-hard optimizations. However, their artificial nature has lessened their impact among practitioners. Certain methodologies quantify suboptimality of heuristics for hard VLSI optimizations based on transformations of existing realistic benchmarks. Hagen *et al.* [10] proposed a general measure of heuristic performance based on the notion of *scaling suboptimality*. The authors construct scaled VLSI instances from initial VLSI instances (e.g., by replicating a netlist or connecting together multiple copies of a netlist) and use these to obtain quantified lower bounds on the suboptimality of VLSI layout heuristics such as placers and partitioners. Kahng and Reda [17] proposed *zero-change transformations* to quantify the suboptimality of existing placers. Given a netlist and its placement from a placer, their zero-change transformations alter the given netlist while keeping its half-perimeter wire length (HPWL) constant, resulting in zero change to achievable HPWL. They showed that placers fail to attain their original HPWL results, with large deviations, on the altered netlists. Their work can provide suboptimality information with respect to a given arbitrary (real) benchmark.

However, none of these previous benchmarks and benchmark generation methodologies helps to evaluate *technology itself*, or the capability of tools in a technology-dependent manner. It would be valuable if semiconductor product companies, foundries, and equipment makers could measure the related routing capacities of alternative BEOL stacks for given combinations of, e.g., placement and routing tools. In this paper, we propose benchmarks and methodologies to measure the routing capacity of BEOL stacks for a given router.

### B. Routability Studies

*1) BEOL Stack Options:* Only a few works in previous literature study the design enablement (i.e., BEOL stack options) in terms of routability. Dong *et al.* [8] proposed an analytical model to estimate the required number of metal layers for a design, based on assumed wirelength distribution and metal layer utilization efficiency. However, realistic constraints such as timing and design rules are not considered in their model. Also, the model does not comprehend the router's behavior. A recent work [4] develops machine learning-based models to predict whether a placement solution is routable for a given number of metal layers; based on this, an early stage estimation of the minimum required number of metal layer can be obtained.

*2) Estimation of Congestion:* Many works perform early estimation of routability and congestion of a placement or

TABLE I
DESCRIPTION OF NOTATIONS USED IN THIS PAPER

| Term | Meaning |
|---|---|
| $D_h$ | gate-level netlist (index of netlist $\equiv h$) |
| $P_k$ | placer (index of placer $\equiv k$) |
| $\Omega_{h,k}$ | placement solution of netlist $D_h$ using placer $P_k$ |
| $U$ | placement utilization |
| $R_j$ | router (index of router $\equiv j$) |
| $B_i$ | BEOL stack (index of BEOL stack $\equiv i$) |
| $N$ | square root of the total number of instances in a design |
| $J$ | number of neighbor-swaps |
| $K$ | number of neighbor-swaps normalized to the total number of instances $N^2$ $(= J/N^2)$ |
| $K_{th}$ | minimum $K$ value that results in routing failure |
| $\Pi_B^{R_j}(D_h, P_k)$ | BEOL ranking in terms of routing capacity on $\Omega_{h,k}$ using $R_j$ |
| $ED$ | sum of edge distances normalized to the total number of instances $N^2$ (See Section V) |
| $CC$ | crossing count normalized to the total number of instances $N^2$ (See Section V) |
| $ED_{th}$ | minimum $ED$ value that results in routing failure |
| $CC_{th}$ | minimum $CC$ value that results in routing failure |

even a netlist. The works [11], [19], [21], [25] simply use academic global routers (e.g., BFG-R [13] or FastRoute 2.0 [24]) to estimate routing congestion. Reference [3] uses pin density and constructs Steiner trees (where multipin nets are split into many two-pin nets) to estimate congestion. Liu and Marek-Sadowska [22] estimated wirelength and routing congestion of a netlist based on net range (i.e., circuit depth spanned by all terminals of a net) and structural pin density (i.e., the ratio between total number of pins of a net versus the total pin count in the design). Spindler and Johannes [29] modeled the routing demand by assuming a rectangular uniform wire density per net. Yang *et al.* [35] used Rent exponent to estimate wirelength distribution of a region, based on which they predict congestion. Taghavi *et al.* [32] proposed a local congestion metric that indicates the routing difficulty for each cell in the design library. Wei *et al.* [33] estimated both global and local routing congestions. Kahng and Xu [18] comprehended blockage effects in their congestion estimation model. The congestion estimation in [23] comprehends layer directive and scenic constraints to limit the routing layer usage and the maximum wirelength of timing-critical nets. Westra *et al.* [34] studied the actual behavior of a routing engine. Their results show that the number of nets with detour (e.g., nets with many bends) is negligible. They also observe that the ratio between *L*-shapes and *Z*-shapes of two-pin nets is roughly a constant value. Based on these observations, they propose a fast congestion prediction model. Saeedi *et al.* [27] also assumes no wires are detoured and proposes an analytical model based on probabilities of *v*-bend paths to estimate congestion. Based on the global routing information, Zhou *et al.* [36] proposed a learning-based model to estimate routing congestion after detailed routing. Based on pin density and congestion map from global routing, Qi *et al.* [26] applied multivariate adaptive regression splines to predict routing congestion.

## III. ASSESSMENT OF DESIGN ENABLEMENTS

In this section, we describe our framework to measure routing capacity of BEOL stack options as well as inherent capability of routers and, potentially, placers. We summarize the notations used in this paper in Table I. The basic idea of our framework (Fig. 1) is as follows. We start with (an instance of) a placed netlist that is straightforward to route, i.e., the routing can be completed by the routing tool with no DRVs. We then

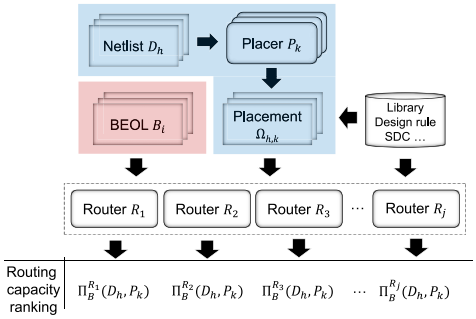



Fig. 1. Our overall goal is to determine whether it is possible to find a quasi-universal ranking of BEOL stacks in terms of routing capacity, and potentially a ranking of place-and-route tools as well.

gradually perturb the placement by randomly swapping the placed locations of adjacently placed cell instances (i.e., a neighbor-swap operation) to increase the routing difficulty.[4] After a number of neighbor-swaps, the routing becomes infeasible (i.e., the number of post-route DRVs exceeds a predefined threshold). We use the number of neighbor-swaps that leads to routing failures as an indicator of the routing capacity of the given BEOL stack, as well as of the capability of the router. For example, we may obtain a ranking of different BEOL stacks in terms of their routing capacities, based on the corresponding values of this indicator.

Our use of the neighbor-swap operator is intuitively reasonable for placement perturbation, for at least two reasons. First, as observed by Alpert *et al.* [1], with high utilizations the placement problem reduces to the ordering problem since there is not much whitespace in which cells can move. Second, a neighbor-swap is an intuitive quantum of suboptimality or error in placement; in this light, starting from a mesh (Rent $p = 0.5$) that is perfectly placed, and then executing random neighbor-swaps, intuitively allows us to dial up particular amounts of suboptimality in placement.

The key benefit of our approach is its ability to systematically evolve a placement from routable to unroutable; this enables evaluation and ranking of BEOL stacks in terms of their routing capacities. It is more challenging to perform such a ranking with real designs, due to larger instance counts (making high-volume experimentation more time-consuming), nonuniformity of topology, and nonuniformity of cell sizes. Further, with real designs, the transition from routable to unroutable is often much more abrupt than with the mesh-based netlists and initial placements that we use. This being said, Section IV-D below confirms the robustness of our analyses and conclusions in multiple ways, including with real netlists and nonuniform (real) cell sizes.

### A. Our Goal

Given a netlist $D_h$, we place it using the placer $P_k$. We then apply our proposed methodology (described below) to measure routing capacities of BEOL stacks $\{B_1, B_2, \ldots, B_i\}$ for a given router $R_j$. We denote the ranking of BEOL stacks in terms of routing capacity as $\Pi_B^{R_j}(D_h, P_k)$. Our goal is to determine a quasi-universal ranking of BEOL stacks (in terms of routing capacity) for a given router, that is, $\Pi_B^{R_j}(D_h, P_k) = \Pi_B^{R_j}(D_{h'}, P_{k'})\ \forall h,\ k,\ h', k'$. In this paper, we measure the

[4]We use a discrete uniform distribution to randomly select a cell, and then a random neighbor of that cell, for swapping. Thus, it is possible for one random swap to revert a previous random swap.

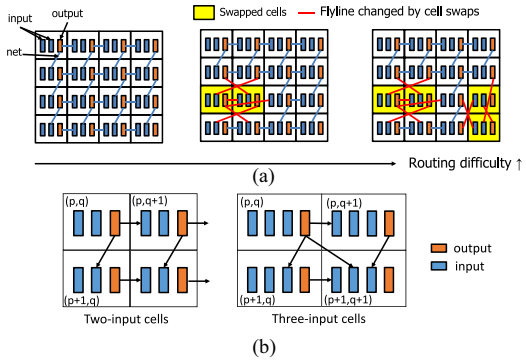



Fig. 2. (a) Illustration of mesh-like placement and perturbations by two neighbor-swap moves. (b) Connections for 2-pin cells and 3-pin cells.

routing capacity of 73 BEOL stack options with various $D_h$, $P_k$, and $R_j$. Detailed information of the 73 BEOL stack options is given in Section IV-A.

### B. Mesh-Like Placement

We first study routing capacity of BEOL stacks based on mesh-like placements. We create a netlist having a square mesh topology (with $M_r$ rows indexed by $p$ and $M_c$ columns indexed by $q$) using a given 2-input or 3-input cell. In the netlist, we connect the output pin of the gate instance with index $(p, q)$ to input pins of the gate instances with indices $(p+1, q)$, $(p, q+1)$, and $(p+1, q+1)$.[5] We then place the netlist (according to its mesh topology) uniformly in a $W_{\text{die}} \times H_{\text{die}}$ region, where $H_{\text{die}} = M_r \cdot H_{\text{gate}}$ and $W_{\text{die}} = M_c \cdot W_{\text{gate}}/U$. Here, $W_{\text{gate}}$ and $H_{\text{gate}}$ are, respectively, the width and height of the given cell, and $U$ is the predefined placement (row) utilization. We note that all gate instances have the same size.

The initial mesh-like placement, where all gate instances are connected only to their physically adjacent gate instances, is easy to route. We gradually increase the routing difficulty by iteratively swapping adjacently placed gate instances. In each move, we randomly select a gate instance and then swap it with one of its (up to four) adjacently placed gate instances (up, down, left, and right) to swap. Fig. 2(a) shows an example with two neighbor-swap moves. Such swap moves will cause routing congestion by progressively worsening the quality (as measured by the sum of embedded edge lengths) of the placement, and eventually lead to a placement that is infeasible to route. We denote the minimum $K$ (number of neighbor-swaps normalized to the total instance count) that leads to an unroutable placement as the *K threshold* ($K_{\text{th}}$). The value of $K_{\text{th}}$ is an indicator of the routing capacity of the given BEOL stack in the context of a (given) router, where a BEOL stack with larger $K_{\text{th}}$ has higher routing capacity.[6] In our experiments, we perform multiple trials of the iterative swapping process for a given initial placement and report the average observed $K_{\text{th}}$ as a measure of routing capacity. Since the $K_{\text{th}}$ measurement requires a number of routing trials, we reduce runtime by first performing a coarse-grained search to narrow down the search space, and then performing a fine-grained search within the reduced search space.[7] The mesh-like placement can be implemented with various different configurations, such as number of total instances, standard cell types, row utilizations, pin alignments, etc. To support the robustness of outcomes and conclusions, we have performed our basic experiment with various mesh-like placements with different configurations. Details are given in Section IV-A.

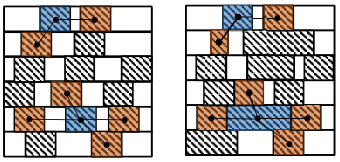


Fig. 3. Illustration of the *neighbor* cell relation in a placement generated by a commercial P&R tool, for the cases of one and adjacent row and two adjacent rows. Brown cells that have line segments drawn to each blue cell are considered as the neighbor cells of that blue cell. Left: cells with uniform (bloated) widths. Right: cells with nonuniform (nonbloated) widths.

### C. Cell Width-Regularized Placement

We also measure routing capacity based on placements generated by a commercial P&R tool, using *cell width-regularized netlists*. Our primary experiment uses bloated standard cells to help maintain placement legality as neighbor-swaps are performed.[8] Of course, a width-regularized netlist is generally not produced in the course of a usual SP&R flow. To generate a width-regularized netlist, we modify the cell LEF [37] such that all gate instances in the netlist have the same size. Here, we simply increase the width of cells without changing the layouts within the cells. We then use a commercial placer to place the netlist with bloated cells. Similar to the method for generating mesh-like placements, we iteratively swap locations of random pairs of adjacently placed cells until the placement becomes unroutable. Again, we use $K_{\text{th}}$ to indicate the routing capacity of the BEOL stack. In this flow, cell bloating avoids placement legalization after each neighbor-swap move. Here, the definition of adjacently placed gate instances within the same row is trivial. However, unlike the placement of the square mesh topology, gate instances are not necessarily aligned vertically. In our experiments, for a given gate instance, we define its neighbors in the two (or, one) adjacent rows as the gate instances with the minimum center-to-center distances in horizontal direction (see Fig. 3, left). As with the experimental flow for *mesh-like placements*, we perform multiple runs of the iterative swapping for a given initial placement and report the average $K_{\text{th}}$ in our experiments. Supplementary studies with two OpenCores [38] designs and nonuniform (nonbloated, i.e., with original and nonregularized widths) cells (see Fig. 3, right) are also reported in Section IV-D.

[5]For a netlist composed of 2-input cells, we only connect the output of $(p, q)$ to the inputs of $(p+1, q)$ and $(p, q+1)$.

[6]The difference between $K$ and $K_{\text{th}}$ is that $K$ is an indicator that represents suboptimality of a placement, while $K_{\text{th}}$ is the minimum $K$ value that results in routing failure. Thus, $K$ can serve as a metric to evaluate the quality of a placement itself in terms of routability (i.e., $K$ indicates the routing difficulty increase with respect to a mesh-like placement); and $K_{\text{th}}$ can be used as a metric to evaluate the routing capacity of a BEOL stack option, or the routing capabilities of a router for a given BEOL stack option and initial placement. This is because $K_{\text{th}}$ shows how much a BEOL stack option (or a router) can sustain (i.e., while still being able to support successfully routing of the design) in terms of suboptimality of the given placement ($K$).

[7]In our experiments with 5K-instance designs, the runtime for routing is ~10 min (single-threaded) on average. For most cases, the runtime for $K_{\text{th}}$ measurement is less than 3 h with a single core, which is a small cost for crucial technology insight. It is possible to perform binary search or other search techniques to further reduce the runtime.

[8]Here, we use cell bloating to make cell widths uniform (i.e., regularized), thus enabling neighbor-swaps without legalization. (When instances have different widths, placement legalization is required to ensure a legal placement after a sequence of neighbor-swaps; hence, the perturbation of the initial placement is not as gradual.) We note that to evaluate routing in terms of pin accessibility, we can use smaller, but still regularized, cell widths for all instances.

TABLE II
TESTCASES

| Name | #Inst. | #Nets | Util. | Clock period ($ns$) |
|---|---|---|---|---|
| AOI-mesh | 5000 | 15000 | 90% | NA |
| AES | 15K | 15K | 65% / 50% | 1.4 |
| VGA | 80K | 80K | 45% | 1.2 |

### D. Extension to Evaluations of Placers and Routers

Similar to evaluating the routing capacity of a BEOL stack option based on $K_{\text{th}}$, we can also use the metric $K_{\text{th}}$ to evaluate capabilities of placers and routers with respect to routability for a given BEOL stack option. For example, a placement that results in a higher $K_{\text{th}}$ value for a given BEOL stack and router is more likely to be routing-friendly. Also, with respect to a given BEOL stack option and a placement, the router that achieves a higher $K_{\text{th}}$ value is more likely to have better performance in terms of routability.

## IV. Experimental Setup and Results

In this section, we describe the setup and results of three basic types of experiments.

1) *Expt1:* Measurement of routing capacity of various BEOL stack options for mesh-like placements and cell width-regularized placements with two routers.
2) *Expt2:* Comparison of different BEOL stack options that have the same routing resources to derive new understanding of equivalences (e.g., between different-pitch layers) that can guide the choice of BEOL stack option.
3) *Expt3:* Further verification of the robustness of the rank-ordering of BEOL stack options.

### A. Experimental Setup

*1) Testcases:* In our experiments, we use mesh-like placements, and the AES encryption core and the enhanced VGA core from OpenCores [38]. Information about testcases is summarized in Table II. For AES, the first utilization number is used for cell width-regularized placement; the second is used for nonbloated-cell-based placement. We use 8-track (8T) standard cells from a 28 nm LP foundry library.[9] Mesh-like placements (e.g., AOI-mesh, or a mesh with AOI cells) are implemented as described in Section III. Since routing hotspots occur locally, intuitively we do not need to use very large designs to measure the routing capacity of a BEOL stack.[10] Thus, we use 5K-instance designs. Indeed, we experimentally confirm below in Section IV-D that the number of instances (i.e., design size) does not change the rank-ordering of BEOL stacks. Also, using small designs help to reduce the runtime. In our experiments, the runtime for routing on the 5K-instance design is ~10 min (single-threaded) on average. The AES and VGA designs are synthesized with *Synopsys Design Compiler K-2015.06-SP4* [40] and implemented with bloated combinational cells (e.g., AOI21, AOI22, BUF, INV, NAND2, NOR2, OAI12, OAI211, and OAI22) and one FF cell. For 8T cells, the post-bloating width of combinational cells is eight placement sites, and the flip-flop cell width, which we leave unchanged, is 23 placement sites. The average and standard deviation of width increase are 2.22 and 1.67 placement sites, respectively, for combinational cell masters.

*2) BEOL Stack Options:* In the 28 nm LP foundry library that we use, the minimum metal width and the minimum metal pitch (i.e., width + spacing) are 0.05 $\mu$m and 0.1 $\mu$m, respectively. We introduce 1×, 1.5×[11] and 2× metal layers based on these minimum width and pitch values. The width (resp. pitch) values for 1.5× and 2× metal layers are 0.074 $\mu$m (0.15 $\mu$m) and 0.1 $\mu$m (0.2 $\mu$m), respectively. We generate 73 BEOL stack options according to the following rules.

1) The $M1$ and $M2$ pitch values are 0.136 $\mu$m and 0.1 $\mu$m, respectively, in the 28 nm library that we use.
2) The total numbers of metal layers = 5, 6, 7, 8.
3) Different numbers of 1×, 1.5×, and 2× layers are tried.
4) From $M2$ upward, the pitch of a given metal layer is always greater than or equal to the pitches of all lower metal layers. In other words, pitch values increase monotonically from lower to upper metal layers.[12] Also, a BEOL stack option is determined according to its numbers of 1×, 1.5×, and 2× layers. For example, if #1×, #1.5×, and #2× layers are, respectively, 4, 1, and 1, then the pitch values of $M1$, $M2$, M3, M4, M5, and M6 are, respectively, 0.136 $\mu$m, 0.1 $\mu$m, 0.1 $\mu$m, 0.1 $\mu$m, 0.15 $\mu$m, and 0.2 $\mu$m.

Table III summarizes the numbers of 1×, 1.5×, and 2× layers, and the *routing resource* ($R$), of all the BEOL stacks that we study, with total numbers of metal layers equal to five, six, seven, and eight. There are 27 possible combinations of BEOL stack options for eight metal layers. We ignore BEOL stack options with layer number larger than eight (in Table III, such BEOL stack options are marked with NA). The *routing resource* (i.e., sum of track densities per unit channel height) of each combination of (BEOL stack option, total # layers) is calculated as $\sum_b (1/[\text{pitch}_b])$, where $b$ is a metal layer, and $\text{pitch}_b$ is the pitch value of $b$.[13] Note that we assume routing is unidirectional in all of our experiments, except for the experiments where we specifically study the impact of bidirectional routing (Sections IV-C and IV-D below).

### B. Expt1: Routing Capacity of Various BEOL Stack Options

In this section, we show our measurement of routing capacity for various BEOL stack options. We first demonstrate positive correlation between the number of #DRVs and $K$, and show how we characterize $K_{\text{th}}$. As detailed in Section III above, in our experiments we use $K_{\text{th}}$ as an indicator of the routing capacity of a BEOL stack option (or, the routing capability of a router). We then show $K_{\text{th}}$ versus routing resource

[9] Below, in Section IV-D, we show that results obtained using 12-track (12T) cells are consistent with those obtained using 8T cells.

[10] Routing difficulty increases as design size increases, as quantitatively analyzed in Section V-B and as empirically demonstrated in Section IV-D [Fig. 9(d)]. Increased routing difficulty in larger designs reflects a higher probability of (local) routing hotspots, for a given placement quality. Indeed, the small designs used in this paper can be viewed as sampled routing hotspots within larger design; see Section V. From our studies, we currently believe that standard-cell netlist complexity of 15K instances (e.g., AES) or greater can afford useful conclusions regarding relative capacities of BEOL stacks.

[11] We derive 1.5× metal and via layers from the existing 1× and 2× layers in 28 nm LEF since there is no 1.5× layer in the 28 nm BEOL LEF that we use. The 1.5× metal width and spacing are 0.074 $\mu$m and 0.076 $\mu$m, respectively. We use the same EOL extension spacing as seen for the 1× layer; for the minimum length rule, we use the mean of the 1× and 2× layers' values. The minimum enclosure area rule is set to that of a 2× layer. We set via spacing such that two vias cannot be placed immediately (i.e., horizontally or vertically) next to each other, but can be placed diagonally adjacent.

[12] The $M1$ layer is an exception to this monotonicity, since it is used for pins or internal routing within standard cells, and its pitch follows the contacted poly pitch.

[13] We use the sum-of-track-densities measure to achieve a design-independent, normalized measure for comparison of BEOL stack options. Thus, our definition of *routing resource* is a normalized routing resource. This measure reflects current usage in industry [28]. We observe that using total track length or total number of routing tracks as a measure of routing capacity would require knowledge of layout dimensions, thus introducing a design-dependence.

TABLE III
BEOL STACK OPTIONS

| Idx | 1× | 1.5× | *all* = 8 | | *all* = 7 | | *all* = 6 | | *all* = 5 | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | 2× | R | 2× | R | 2× | R | 2× | R |
| 0 | 2 | 0 | 6 | 40 | 5 | 35 | 4 | 30 | 3 | 25 |
| 1 | 3 | 0 | 5 | 45 | 4 | 40 | 3 | 35 | 2 | 30 |
| 2 | 4 | 0 | 4 | 50 | 3 | 45 | 2 | 40 | 1 | 35 |
| 3 | 5 | 0 | 3 | 55 | 2 | 50 | 1 | 45 | 0 | 40 |
| 4 | 6 | 0 | 2 | 60 | 1 | 55 | 0 | 50 | NA | NA |
| 5 | 7 | 0 | 1 | 65 | 0 | 60 | NA | NA | NA | NA |
| 6 | 8 | 0 | 0 | 70 | NA | NA | NA | NA | NA | NA |
| 7 | 2 | 1 | 5 | 42 | 4 | 37 | 3 | 32 | 2 | 27 |
| 8 | 3 | 1 | 4 | 47 | 3 | 42 | 2 | 37 | 1 | 32 |
| 9 | 4 | 1 | 3 | 52 | 2 | 47 | 1 | 42 | 0 | 37 |
| 10 | 5 | 1 | 2 | 57 | 1 | 52 | 0 | 47 | NA | NA |
| 11 | 6 | 1 | 1 | 62 | 0 | 57 | NA | NA | NA | NA |
| 12 | 2 | 2 | 4 | 43 | 3 | 38 | 2 | 33 | 1 | 28 |
| 13 | 3 | 2 | 3 | 48 | 2 | 43 | 1 | 38 | 0 | 33 |
| 14 | 4 | 2 | 2 | 53 | 1 | 48 | 0 | 43 | NA | NA |
| 15 | 5 | 2 | 1 | 58 | 0 | 53 | NA | NA | NA | NA |
| 16 | 6 | 2 | 0 | 63 | NA | NA | NA | NA | NA | NA |
| 17 | 2 | 3 | 3 | 45 | 2 | 40 | 1 | 35 | 0 | 30 |
| 18 | 3 | 3 | 2 | 50 | 1 | 45 | 0 | 40 | NA | NA |
| 19 | 4 | 3 | 1 | 55 | 0 | 50 | NA | NA | NA | NA |
| 20 | 5 | 3 | 0 | 60 | NA | NA | NA | NA | NA | NA |
| 21 | 2 | 4 | 2 | 47 | 1 | 42 | 0 | 37 | NA | NA |
| 22 | 3 | 4 | 1 | 52 | 0 | 47 | NA | NA | NA | NA |
| 23 | 4 | 4 | 0 | 57 | NA | NA | NA | NA | NA | NA |
| 24 | 2 | 5 | 1 | 48 | 0 | 43 | NA | NA | NA | NA |
| 25 | 3 | 5 | 0 | 53 | NA | NA | NA | NA | NA | NA |
| 26 | 2 | 6 | 0 | 50 | NA | NA | NA | NA | NA | NA |

(number of available routing tracks) of various BEOL stack options. Our experimental results suggest that routing resource may not be the only factor to determine routing capacity of a BEOL stack option. In light of this, we further study the ranking of BEOL stack options with the same routing resource based on our framework using mesh-like placement and cell width-regularized placement, with two commercial routers.

*1) Characterization of $K_{th}$:* We measure #DRVs to characterize the $K_{\text{th}}$. Fig. 4(a) shows #DRVs versus $K$ with the AOI-mesh design. For each $K$ value, we run three trials with different random sequences of neighbor-swaps to avoid noise from tools and randomness of the perturbation process.[14] Each dot corresponds to a pair of a perturbation and a BEOL option. The average values of the sets of three runs are marked by the solid traces. We increase $K$ until the average #DRVs $>$ 150. We then record the current $K$ as $K_{\text{th}}$. In Fig. 4(a), $K_{\text{th}}$ values are zero, four, 11, and $>$14 for BEOL0_6, BEOL1_6, BEOL2_6, and BEOL3_6. The naming convention is {*BEOL stack option index*}_{*total #layers*}. A higher $K_{\text{th}}$ value for a particular BEOL stack means that the BEOL stack has a higher routing capacity to sustain more perturbed placements (i.e., placements with more hotspots).

*2) $K_{th}$ Versus Routing Resources:* We study the ranking of BEOL stack options with respect to $K_{\text{th}}$. Fig. 4(b) shows $K_{\text{th}}$ values versus routing resource values, measured using three commercial routers ($R1$, $R2$, and $R3$). The figure shows a positive correlation between $K_{\text{th}}$ and routing resource values for all the three routers.[15] We find that there are several points which show a reversed correlation between $K_{\text{th}}$ and routing resources, i.e., a smaller routing resource point shows a higher $K_{\text{th}}$. Also, there are points which have different $K_{\text{th}}$ values even though the corresponding routing resource values are the same. This

[14]In our experiments, $K_{\text{th}}$ varies by $\leq 2$ across any set of three trials.

[15]To avoid unnecessary flow complications, we report post-route #DRVs out of the routing tool used. We separately verify that routing tools $R1$, $R2$, and $R3$ report the same number of DRVs for a given routed DEF file.

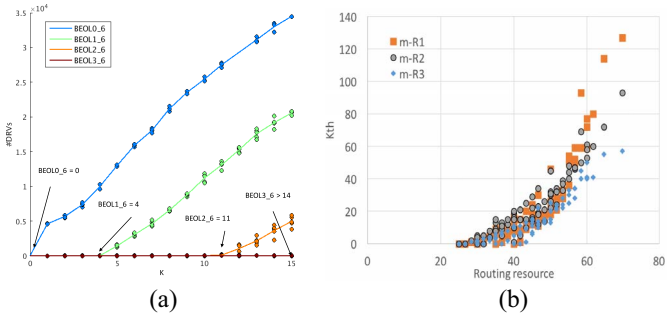



Fig. 4. (a) Number of DRVs versus $K$. This result is from a mesh-like placement implemented with 5000 AOI21 cells, and row utilization of 90%. (b) $K_{\text{th}}$ values versus routing resource with three commercial routers ($R1$, $R2$, and $R3$). The results are extracted using mesh-like placements implemented with 5000 AOI21 cells and 90% row utilization. Each dot corresponds to a BEOL stack option. We observe that BEOL stack options with the same routing resource can show different $K_{\text{th}}$ values.

result suggests that "counting routing tracks" may not be an accurate estimation of the routing capacity of a BEOL stack option.[16] Indeed, intuition tells us that measurement of the routing capacity is nontrivial, since gear ratio of metal pitches and via blockages affect routability. Additionally, there could be effects of "height" of layers—lower metal layers would be more valuable than upper metal layers due to vias. Simply counting routing tracks does not account for such impacts on routability.

*3) BEOL Stack Options With Same Routing Resource:* Since we notice that routing resource may not be a good indicator of routing capacity, we further analyze different BEOL stack options that have the same routing resource value. As $R3$ is an older release of a commercial router, in subsequent experiments below we focus on two routers, i.e., $R1$ and $R2$, which are essentially the latest releases available of two commercial routers. We group BEOL stack options according to routing resources, i.e., all BEOL stack options in each group have the same routing resource value that corresponds to the group. Fig. 5 and Table IV show $K_{\text{th}}$ values for different BEOL stack options with the same routing resource values. The results of six types of implementations, e.g., 1) mesh-like placement with $R1$ (*m*-$R1$); 2) mesh-like placement with $R2$ (*m*-$R2$); 3) cell width-regularized placement with placer $P1$ and router $R1$ (*r*-$P1$-$R1$); 4) placer $P2$ and router $R1$ (*r*-$P2$-$R1$); 5) placer $P1$ and router $R2$ (*r*-$P1$-$R2$); and 6) placer $P2$ and router $R2$ (*r*-$P2$-$R2$), are reported in the figure.[17]

Fig. 6 shows the correlations of the rank-ordering of BEOL stack options (in increasing order of $K_{\text{th}}$) between mesh-like placement results and cell width-regularized placement results. We observe that the rank-ordering based on mesh-like placement and the rank-ordering based on cell width-regularized placements are well-correlated. The correlation coefficients of the rank-orderings of BEOL stacks between the six types of implementations are all larger than 0.9.

We summarize our observations from the results as follows.

1) BEOL stack options with the same routing resources show different $K_{\text{th}}$ values. This suggests that the routing resource alone is not sufficient to quantify the routing capacity of given BEOL stack options.

[16]Nevertheless, total track count has been used (and is still used) as a routing capacity metric in industry, and it is an important factor that decides business and technology strategic plans for patterning technology and equipment manufacturing [28].

[17]For cell width-regularized placements, placements change depending on BEOL stack options, since BEOL stack options affect placements due to in-built trial routing mechanisms inside the placement tool.

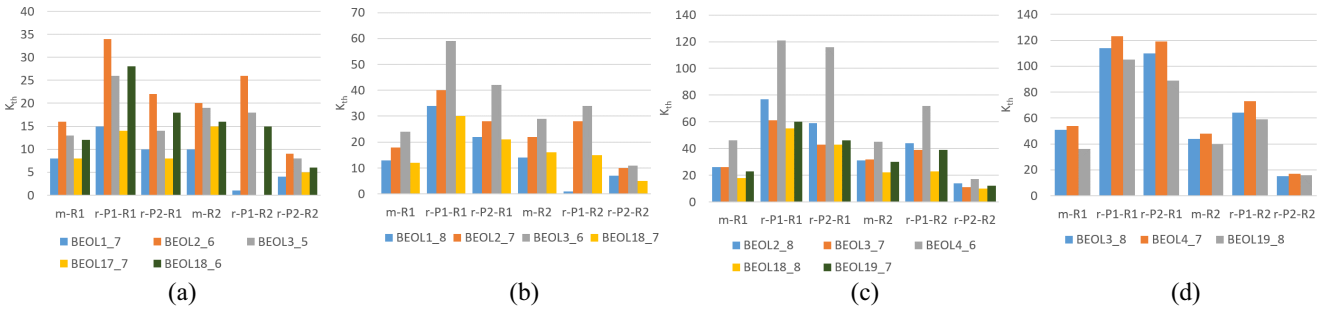



Fig. 5. $K_{th}$ values for (a) Group 1, (b) Group 2, (c) Group 3, and (d) Group 4 of BEOL stack options in Table IV. The results based on six types of implementations are reported: 1) mesh-like placement with $R1$ (*m*-$R1$); 2) mesh-like placement with $R2$ (*m*-$R2$); 3) cell width-regularized placement with placer $P1$ and router $R1$ (*r*-$P1$-$R1$); 4) placer $P2$ and router $R1$ (*r*-$P2$-$R1$); 5) placer $P1$ and router $R2$ (*r*-$P1$-$R2$); and 6) placer $P2$ and router $R2$ (*r*-$P2$-$R2$).

TABLE IV
$K_{th}$ VALUES FOR GROUPS OF BEOL STACK OPTIONS HAVING THE SAME ROUTING RESOURCE. THE ROUTING RESOURCE ($T$) VALUE FOR EACH GROUP IS SHOWN IN THE FIRST COLUMN. $K_{th}$ VALUES ARE MEASURED BASED ON SIX TYPES OF IMPLEMENTATIONS: 1) *m*-$R1$; 2) *r*-$P1$-$R1$; 3) *r*-$P2$-$R1$; 4) *m*-$R2$; 5) *r*-$P1$-$R2$; AND 6) *r*-$P2$-$R2$

| Group ($T$) | Name | (i) | (ii) | (iii) | (iv) | (v) | (vi) |
|---|---|---|---|---|---|---|---|
| Group 1 (40) | BEOL1_7 | 8 | 15 | 10 | 10 | 1 | 4 |
| | BEOL2_6 | 16 | 34 | 22 | 20 | 26 | 9 |
| | BEOL3_5 | 13 | 26 | 14 | 19 | 18 | 8 |
| | BEOL17_7 | 8 | 14 | 8 | 15 | 0 | 5 |
| | BEOL18_6 | 12 | 28 | 18 | 16 | 15 | 6 |
| Group 2 (45) | BEOL1_8 | 13 | 34 | 22 | 14 | 1 | 7 |
| | BEOL2_7 | 18 | 40 | 28 | 22 | 28 | 10 |
| | BEOL3_6 | 24 | 59 | 42 | 29 | 34 | 11 |
| | BEOL18_7 | 12 | 30 | 21 | 16 | 15 | 5 |
| Group 3 (50) | BEOL2_8 | 26 | 77 | 59 | 31 | 44 | 14 |
| | BEOL3_7 | 26 | 61 | 43 | 32 | 39 | 11 |
| | BEOL4_6 | 46 | 121 | 116 | 45 | 72 | 17 |
| | BEOL18_8 | 18 | 55 | 43 | 22 | 23 | 10 |
| | BEOL19_7 | 23 | 60 | 46 | 30 | 39 | 12 |
| Group 4 (55) | BEOL3_8 | 51 | 114 | 110 | 44 | 64 | 15 |
| | BEOL4_7 | 54 | 123 | 119 | 48 | 73 | 17 |
| | BEOL19_8 | 36 | 105 | 89 | 40 | 59 | 16 |

2) The correlation of the rank-orderings of BEOL stack options between cases *m*-$R1$ and *m*-$R2$ is high (i.e., correlation coefficient $> 0.9$). This suggests at least a possibility that the rank-ordering of BEOL stack options is quasi-universal across different routers for the same placements. (From our studies, we conjecture that such quasi-universality holds regardless of the tool that produced the given placement.)
3) As shown in Fig. 6, the correlation of the rank-orderings of BEOL stack options between cases *m*-$R1$, *r*-$P1$-$R1$, and *r*-$P2$-$R1$ is high (i.e., all correlation coefficients $> 0.9$). Also, the correlation of the rank-orderings of BEOL stack options between cases *m*-$R2$, *r*-$P1$-$R2$, and *r*-$P2$-$R2$ is high (i.e., all correlation coefficients $> 0.9$). This could indicate that at least in this routability-centric evaluation where other constraints such as timing, power and noise are not considered, the folklore gap between artificial and real benchmarks—in terms of ability to provide insight into CAD heuristic performance—may not be as significant as previously believed.

### C. Expt2: Comparison of BEOL Stack Options

In this section, we seek to understand further the implications of the results from our framework, and which BEOL stack options can better apply a given amount of routing resource. Here, we regard the BEOL stack having the larger $K_{th}$ among all of the candidate BEOL stacks as the better BEOL stack option. We study the following.

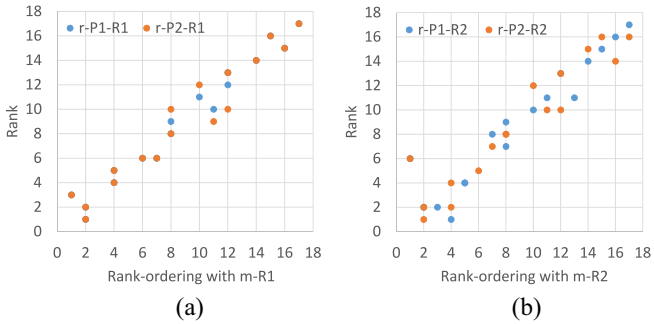



Fig. 6. Correlations of the rank-ordering of BEOL stack options (in increasing order of $K_{th}$) between mesh-like placement results and cell width-regularized placement results: (a) $R1$ and (b) $R2$.

*1) Correlation Between $K_{th}$ and Maximum Achievable Utilization:* To study the correlation between $K_{th}$ and area, we run P&R with different initial row utilization values and see if there are DRVs after routing. We implement cell width-regularized placements (AES) with bloated cells and sweep the initial row utilization from 45% to 75%.[18] The BEOL stack options in Group 1 are tested with unidirectional routing, placer $P1$ and router $R1$ (*r*-$P1$-$R1$ in Fig. 5). We record the maximum achievable utilization values such that #DRVs $< 150$. The placements are perturbed with $K = 20$ to make routing harder. Fig. 7 shows the $K_{th}$ values obtained from cell width-regularized placements (*r*-$P1$-$R1$) in blue bars (left $y$-axis) and the corresponding maximum achievable (initial) row utilization values in orange trace (right $y$-axis). We observe that a BEOL stack option with a higher $K_{th}$ achieves a higher maximum achievable utilization.

*2) Analyses of BEOL Stack Options With Same $K_{th}$:* Fig. 8 shows the results of *m*-$R1$ mesh-like placements for subsets of BEOL stack options with same $K_{th}$ values. Fig. 8(a)–(d) shows the results of BEOL stack options with $K_{th} = 4$–$6$, $K_{th} = 9$–$11$, $K_{th} = 14$–$16$, and $K_{th} = 19$–$21$, respectively. Since $K_{th}$ indicates routing capacity, BEOL stack options with the same $K_{th}$ values are regarded as equivalent with respect to routing capacity. Example findings from *m*-$R1$ results are as follows.

1) The added or incremental routing capacity of a layer depends on the height of the layer according to the *m*-$R1$ results, as one would expect. For example, in Fig. 8(a),

[18]For row utilizations $> 75\%$, the results do not change since bloated standard cells introduce porosities in placements that we cannot control using initial row utilizations. For example, for 8T cell masters, the average width increase is 2.22 placement sites, which means that every cell effectively has $\sim$2 placement sites of embedded, i.e., internal and whitespace.

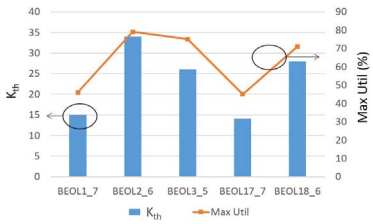



Fig. 7. $K_{th}$ values obtained from cell width-regularized placements versus maximum achievable (initial) row utilization values. The BEOL stack options in Group 1 are tested using the AES design and $R1$.

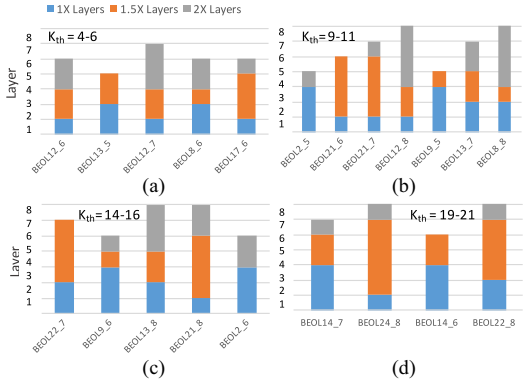



Fig. 8. Results of *m-R*1, showing for subsets of BEOL stack options that have the same $K_{th}$ values. (a) BEOL stack options with $K_{th}$ = 4–6; (b) BEOL stack options with $K_{th}$ = 9–11; (c) BEOL stack options with $K_{th}$ = 14–16; and (d) BEOL stack options with $K_{th}$ = 19–21.

one 1× layer on M3 is equivalent to two 2× layer on M5 and M6 (BEOL12_6 and BEOL13_5); or, one 1× layer on M3 and one 2× layer on M5 are equivalent to two 1.5× layers on M3 and M5 (BEOL8_6 and BEOL17_6). In Fig. 8(b), two 1× layers on M3 and M4 are equivalent to four 1.5× layers on M3, M4, M5, and M6 (BEOL2_5 and BEOL21_7).

2) We also observe that if the numbers of 1× and 1.5× layers are sufficient, higher metal layers do not significantly affect routing capacity. In Fig. 8(b), if the number of 1 × and 1.5× layers ≥ 6, additional layers do not help.
3) Last, the data indicate that using more 1× layers reduces the required total number of layers for a given routing capacity. Obviously, added dimensions to our study, such as signal integrity performance and/or power delivery, are directions for follow-on research.

*3) 1-D Versus 2-D Routing:* We study the question "On which layer(s) is it most valuable to enable bidirectional routing?" Here, we use 1-D routing to indicate unidirectional routing where routing in the nonpreferred direction is not allowed. We use 2-D routing to indicate bidirectional routing where routing segments in both horizontal and vertical directions can exist on the given layer. We use the same metal pitch for both directions for 2-D routing. We compare $K_{th}$ of various BEOL options with six 1× 1-D layers (e.g., BEOL4_6) and one 2-D routing layer enabled using the *m-R*1 implementation. Table V shows the layer configuration of each BEOL option and the corresponding $K_{th}$ value. We note that the routing resource of each option in this experiment is the same since all metal layers are 1× layers. However, each option has different horizontal and vertical resources. 1-D, 2-D-B, and 2-D-D have three horizontal routing layers and two vertical routing layers. And, 2-D-A and 2-D-C have two horizontal routing layers and three vertical routing layers. All BEOL options with one 2-D routing layer show smaller $K_{th}$ compared to the 1-D BEOL option. This may indicate that having two 1-D layers is always better than having one 2-D layer. Even though the routing resource (measured by pure track count) is the same in both cases, routing rules can limit the utilization of the given routing resource in a 2-D routing layer.

TABLE V
$K_{th}$ RESULTS OF VARIOUS 2-D OPTIONS

| Option | M2 | M3 | M4 | M5 | M6 | $K_{th}$ |
|---|---|---|---|---|---|---|
| 1-D (BEOL4_6) | 1-D | 1-D | 1-D | 1-D | 1-D | 46 |
| 2-D-A | 2-D | 1-D | 1-D | 1-D | NA | 13 |
| 2-D-B | 1-D | 2-D | 1-D | 1-D | NA | 14 |
| 2-D-C | 1-D | 1-D | 2-D | 1-D | NA | 12 |
| 2-D-D | 1-D | 1-D | 1-D | 2-D | NA | 13 |

By comparing between 2-D options, we observe that it may be more valuable to enable 2-D routing on lower metal layers; this can be seen by comparing 2-D-B versus 2-D-D, and 2-D-A versus 2-D-C. We may also infer that horizontal routing resources are more valuable than vertical resources, for our nearly square blocks comparing 2-D-A versus 2-D-B, or 2-D-C versus 2-D-D. That is, BEOL options with more horizontal routing resources show higher $K_{th}$ values (i.e., better routing capacity). We do not claim to be the first to observe such correlations, and our results are each specific to a given enablement. This being said, ours is the first *framework* for quantifying such assessments in a general, design-agnostic manner.

### *D. Expt3: Robustness of the Rank-Ordering of BEOL Stack Options*

In this section, we show further experimental results to support the robustness of our observed quasi-universal rank-ordering of BEOL stack options across mesh-like placements and cell width-regularized placements, and two routers.

*1) Various Mesh-Like Placement Configurations:* We perform experiments for various mesh-like placements implemented with different configurations. Note that the baseline is the AOI-mesh in Table II. The different configurations for mesh-like placements are summarized as follows.

1) Different total numbers of instances (#instances = 5000, 15 000, and 20 000).
2) Different standard cell types (NAND2 and AOI21).
3) Different row utilizations (70%, 80%, and 90%).
4) Different pin alignment.
5) Different routing direction (1-D versus 2-D).
6) Different standard cells (8T and 12T).

Fig. 9 shows $K_{th}$ values for five BEOL stack options, measured with various mesh-like placements with different configurations. The five BEOL stack options have the same routing resources, $T$ = 40 (Group 1 in Section IV-B). Our findings from the results are as follows.

1) Fig. 9(a) shows the results of two mesh-like placements implemented with AOI21 (three-input cell) and NAND2 (two-input cell). The width values of AOI21 and NAND2 are 1.088 $\mu$m and 0.952 $\mu$m, respectively. The NAND2 case has higher $K_{th}$ values. This is because the NAND2-based implementation has a lower net degree. However, the rank-ordering of BEOL stack options with respect to $K_{th}$ values is the same as that of the AOI21 case.

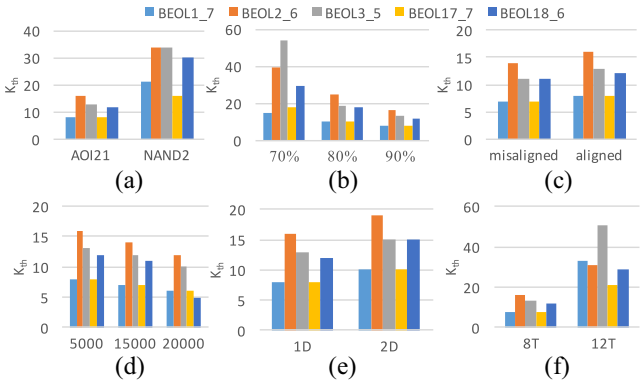



Fig. 9. $K_{th}$ of mesh-like placements with various configurations. (a) Different cell types: AOI21 (three-input cell) and NAND2 (two-input cell); (b) different row utilizations: 70%, 80%, and 90%; (c) different pin alignments; (d) different total number of instances: 5000, 15 000, and 20 000; (e) unidirectional routing (1-D) and bidirectional routing (2-D); and (f) 8T and 12T cells.

2) Fig. 9(b) shows the results of *m-R*1 with different row utilizations, 70%, 80%, and 90%.[19] As expected, placements with lower utilizations achieve higher $K_{th}$. The rank-ordering of BEOL stack options with respect to $K_{th}$ is the same for 80% and 90% cases, but there is a deviation for 70% case. (i.e., reversed ordering of BEOL3_5 and BEOL2_6).
3) Fig. 9(c) shows the impact of different pin alignments. In a mesh-like placement, pins of standard cells are aligned unless we apply different offsets for each placement row. For the "misaligned" case, we add a different offset for each placement row to create vertical misalignment between cells in adjacent placement rows. The result suggests that the impact of pin alignment is negligible.
4) Fig. 9(d) shows the impact of different numbers of instances, i.e., 5000, 15 000, and 20 000. As the number of instances increases, $K_{th}$ slightly decreases. This result suggests that design size is related to $K_{th}$ (see Section VI, below). $K_{th}$ of BEOL18_6 is relatively lower in the 20 000 case, and this results in a different rank-ordering of BEOL stack options. However, except for BEOL18_6, the rank-ordering of BEOL stack options remains consistent.
5) Fig. 9(e) shows results of 1-D and 2-D routing, where bidirectional routing is enabled for all metal layers for the 2-D routing. (Note that this is a different experiment from 1-D versus 2-D routing in Section IV-C, where bidirectional routing is enabled for only one layer.) $K_{th}$ is higher with 2-D routing, which suggests routing capacity of 2-D routing is larger. The rank-ordering of BEOL stack options remains the same.
6) Fig. 9(f) shows results of 8T and 12T cells. In this experiment, we use 8T and 12T cells with the same width. $K_{th}$ is higher with 12T cell, as one would expect. The rank-ordering of BEOL2_6 and BEOL3_5 is reversed with 12T cell, as compared to other configurations.[20]

[19]We observe that in the implementation with 70% utilization, due to rounding effects the space between two horizontally adjacent cells can vary (three or four placement sites). Such cases, where cells are not distributed uniformly, lead to more DRVs in the router outcome. To make a mesh-like placement with uniform distribution, the row utilizations we can implement are limited to $w/(w+s)$, where $w$ and $s$ are the cell width and the spacing between cells, respectively. Since $w$, $s$ are integer (in placement sites) and since $w = 8$ in the experiment, we can implement row utilizations 72.7%, 80.0%, and 88.9% with $s = 3, 2, 1$, respectively. To implement an exact 70% utilization, we would need to use multiple values of $s$, which would result in nonuniform mesh-like placements. Therefore, we use 72.7%, 80.0%, and 88.9% utilizations for 70%, 80%, and 90%, respectively, to make a fairer comparison among the three cases.

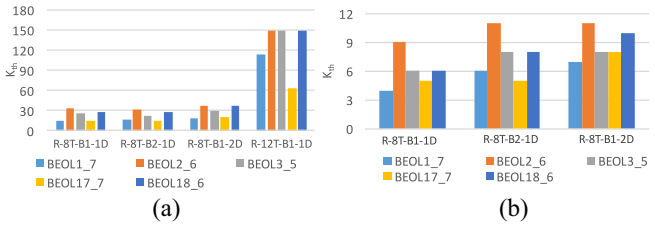



Fig. 10. $K_{th}$ values of the BEOL stack options in Group 1 for various cases: 1) a placement implemented with bloated (i.e., width-regularized) 8T cells and unidirectional routing (*R-8T-B1-1-D*); 2) a placement implemented with bloated cells of a wider width, and unidirectional routing (*R-8T-B2-1-D*); 3) a placement implemented with bloated cells and bidirectional routing (*R-8T-B1-2-D*); and 4) a placement with bloated 12T cells (*R-12T-B1-1-D*). The results are obtained (a) using (*P*1 and *R*1) and (b) using (*P*2 and *R*2).

In summary, the rank-ordering of the BEOL stack options with respect to $K_{th}$ does not change significantly across different mesh-like placements with a wide range of configurations. However, there do exist deviations across configurations (track height, utilization, and routing directionality), indicating that no one configuration perfectly represents all other possible configurations. Thus, it would be important for designers to select a proper set of configurations to reflect the properties of target designs.

*2) Cell Width-Regularized Placement With Standard Cell Variants:* We perform similar experiments with cell width-regularized placements using more standard cell variants,[21] for five BEOL stack options in Group 1. Fig. 10 shows $K_{th}$ values of the BEOL stack options in Group 1 for four cases among the combinations of two heights (8T and 12T), two widths (eight and ten placement sites) and two routing directions (1-D and 2-D). That is: 1) 8T bloated cells (width = 8) and unidirectional routing (*R-8T-B1-1-D*); 2) 8T bloated cells of a wider width (width = 10), and unidirectional routing (*R-8T-B2-1-D*); 3) 8T bloated cells (width = 8) and bidirectional routing (*R-8T-B1-2-D*); and 4) 12T bloated 12T cells (width = 8) and unidirectional routing (*R-12T-B1-1-D*). The results are obtained using *P*1 and *R*1 [Fig. 10(a)], and *P*2 and *R*2 [Fig. 10(b)].[22] We observe that *R-8T-B2-1-D* shows higher $K_{th}$ values than *R-8T-B1-1-D*. This may be due to larger spacings between pins in the *R-8T-B2-1-D* case. We also see that *R-8T-B1-2-D* shows slightly higher $K_{th}$ values when compared to *R-8T-B1-1-D*. We observe that using 12T cells increases routing capacities dramatically (*R-12T-B1-1-D*).

*3) Placements Generated by Standard SP&R Flow:* We support our observed routing capacity-based ordering of BEOL stacks using placements that are implemented by a standard SP&R flow. Specifically, we implement placements for the AES and VGA testcases using a full set of library cells without any modification, and apply our neighbor-swap-based approach. As noted in footnote 8, to maintain placement legality we check whitespace after every neighbor-swap operation between placement rows. If the row utilization with the updated

[20]We have applied multiple runs (>10) to further remove noise due to randomness of the sequence of neighbor-swaps, but have found results to be consistent (i.e., stable).

[21]We use 8T and 12T cells bloated to have different widths (i.e., eight and ten placement sites). The width of an FF is 23 placement sites.

[22]We could not obtain results for case 4) with *P*2 and *R*2, since *R*2 could not produce DRV-clean results for $K = 0$ with 12T cells.

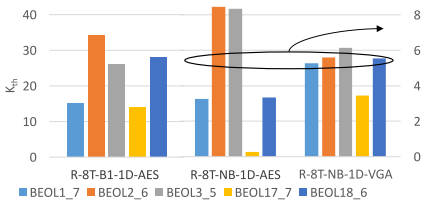



Fig. 11. $K_{\text{th}}$ values of the BEOL stack options in Group 1 for AES and VGA placements with nonbloated 8T cells (*R-8T-NB-1-D-AES* and *R-8T-NB-1-D-VGA*). *R-8T-B1-1-D-AES* is shown for a reference. The results are obtained using ($P1$ and $R1$).

**Algorithm 1** Placement Perturbation

**Input:** an input placement, total number of instances $N^2$, target $K$, target utilization $U$, utilization margin $M$
**Output:** a perturbed placement

```
1: num_swap ← 0
2: while num_swap < N^2 · K do
3:     cell ← RandomlySelectCell
4:     dir ← RandomlySelectDirection
5:     neighbor_cell ← GetNeighborCell(cell, dir)
6:     Swap cell, neighbor_cell
7:     Update row utilization
8:     num_swap ← num_swap + 1
9:     if ((dir == north) || (dir == south)) && w_cell ≠ w_neighbor_cell then
10:        if row utilization > U + M for any row then
11:            Swap cell, neighbor_cell (revert)
12:            num_swap ← num_swap − 1
13:        end if
14:    end if
15: end while
```

whitespace exceeds a predefined sum of initial row utilization + margin, we revert the neighbor-swap operation. For AES and VGA, we use 50% and 45% for initial row utilization, respectively, and we use 1% margin for both designs.

Algorithm 1 gives details of the procedure for perturbing real-cell-based real placements. We initialize the number of neighbor-swaps performed (*num_swap*) in line 1. We iteratively perform neighbor-swaps until the total number of neighbor-swaps reaches the target (calculated from the given $N^2$ and $K$). For a neighbor-swap, we randomly select target cell (cell) and direction (dir) (north, south, east, and west) in lines 3 and 4. This determines the neighbor cells of a cell (line 5). In line 6, the target cell and its neighbor cell are swapped, and row utilizations are updated (line 7). We increment *num_swap* in line 8. If the neighbor-swap is performed between rows and the widths of the target cell and its neighbor cell are different (line 9), we check that updated row utilizations do not exceed a predefined limit (line 10). If this check fails, we revert the neighbor-swap operation (lines 11 and 12).

The use of real cell widths leads to less-gradual perturbation due to the effect of placement legalization before routing. The results of the nonbloated-cell-based implementation are shown in Fig. 11 (*R-8T-NB-1-D-AES* and *R-8T-NB-1-D-VGA*). *R-8T-B1-1-D-AES* is given as a reference (*R-8T-B1-1-D* in Fig. 10). We see that indeed, $K_{\text{th}}$ values of *R-8T-NB-1-D-AES* and *R-8T-NB-1-D-VGA* are dramatically smaller than those of *R-8T-B1-1-D-AES*. Although our focus is on the rank-orderings of BEOL stack routing capacities based on $K_{\text{th}}$, as opposed to the magnitudes of $K_{\text{th}}$ values, this experiment clearly shows a gap between cell width-regularized placements and placements generated within production SP&R flows. We emphasize that the AES-derived, width-regularized testcase is a compromise between real and synthetic, due to the cell bloating. And, further understanding of the significance of the cell bloating used in our studies of the PROBE methodology remains an important direction for future research. In terms of the rank-ordering, the result of *R-8T-NB-1-D-AES* remains the same as the reference, while there is a deviation (BEOL2_6) in the *R-8T-NB-1-D-VGA* case. This data point may indicate that the rank-ordering has additional dependencies on netlist structure and size.

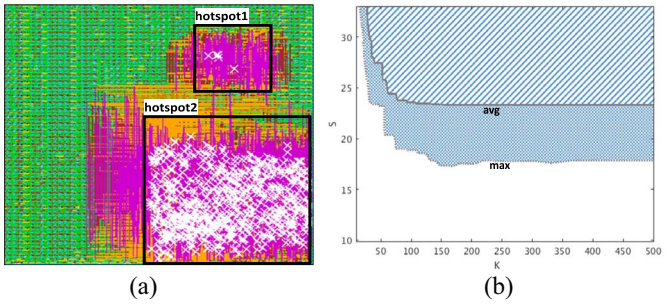



Fig. 12. (a) Routed layout with two hotspots. DRVs are indicated by white crosses. Other colors (green, pink, and orange) indicate metal layers. Although each hotspot is perturbed by the same $K$ ($\times S^2$, where $S$ is the dimension of the $S \times S$ hotspot), hotspot1 ($S = 16$) produces noticeably fewer DRVs than hotspot2 ($S = 33$). (b) Contour map that indicates routability for various ($S$, $K$) pairs. The solid line is the contour based on the average number of DRVs, and the dotted line is the contour based on the maximum number of DRVs in 10 trials per each ($S$, $K$) pair. Based on the contour lines, the upper shaded regions show where routing fails.

## V. Additional Study of Routing Hotspot and Routing Failure

The studies above focus on ranking of BEOL stack options based on fixed-size designs and use of the $K_{\text{th}}$ criterion as an indicator of routing capacity. The $K_{\text{th}}$-based BEOL stack ordering is shown empirically to be stable across a number of factors—routing tool, netlist topology, utilization, porosity, layer directionality, etc. Notably, Fig. 9(d) suggests that the $K_{\text{th}}$-based stack ordering is independent from design size, even as the $K_{\text{th}}$ values themselves change with design size.

By necessity, our studies involve small netlists, which raises the important question of how to extend insights to when designs are large. In this section, we provide additional studies of: 1) the size and placement quality of a routing hotspot and 2) the impact of design size on routing failure. These provide context for the preceding experimental results: 1) routing failure (in hotspots) is a function of both hotspot size and placement suboptimality ($K_{\text{th}}$) and 2) the observed change of $K_{\text{th}}$ with design size [Fig. 9(d)] matches the outcomes of a more quantitative analysis.

### A. Routing Hotspots

Routing failure is caused by local routing hotspots in many cases. However, not every routing hotspot contributes equally to routing failure. Fig. 12(a) shows hotspots of two different sizes, along with post-route DRVs (white crosses). Both hotspots are generated with the same number $K \times S^2$ of neighbor-swaps (where $S$ is the dimension of the $S \times S$ hotspot, and defines the size of the hotspot). We observe that while the two hotspots have the same $K$, the numbers of DRVs are different, and hotspot1 does not contribute much to DRVs. This example suggests that the size of routing hotspots is another key factor, in addition to $K$, that determines routing failure. Thus, in this section, we further empirically study various sizes of routing hotspots.

In mesh-like placements, we generate routing hotspots with various locations and sizes. Specifically, we vary the size of a hotspot (i.e., by performing $K \times S^2$ random neighbor-swaps

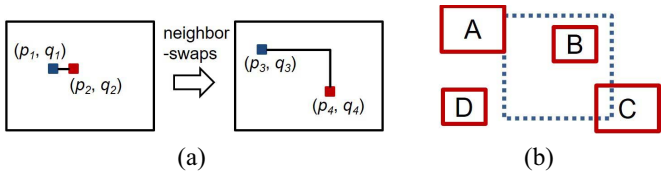



Fig. 13. Illustrations of edge mapping, ED and CC. (a) Initial lattice edge $[(p_1, q_1), (p_2, q_2)]$ [vertices $(p_1, q_1)$, and $(p_2, q_2)$ are adjacent in the initial mesh] is *mapped to* $[(p_3, q_3), (p_4, q_4)]$ after neighbor-swaps. For this edge, the ED $= |p_3 - p_4| + |q_3 - q_4|$. (b) In red are net bounding boxes, and in blue is the window, where Net B and Net C contribute to CC, but Net A and Net D do not.

within a specific subregion). An $S \times S$ hotspot is a subregion with $S$ columns and $S$ rows. We study various $(S, K)$ pairs and record #DRVs. For each $(S, K)$ pair, we generate ten random data points. Fig. 12(b) shows a contour map that indicates routability for various $(S, K)$ pairs. The $x$-axis shows $K$ normalized by $S^2$ because there are $O(S^2)$ total edges, and the $y$-axis shows $S$. The solid (resp. dotted) line is the contour based on average (resp. maximum) numbers of DRVs (of ten trials for the corresponding $(S, K)$ pair), where the upper shaded regions correspond to routing failures. The figure shows that: 1) if the size of hotspots ($S$) is smaller than a certain value, then $K_{\text{th}} = \infty$ and 2) the $K_{\text{th}}$ values increase as the size of hotspots decreases.

### *B. Impact of Design Size on Routing Failure*

Design size is an important factor (along with routing algorithms, BEOL stack options, netlist topology, placement utilization, etc.) that affects routing difficulty. To better understand the connection between design size and routing difficulty that is seen in Fig. 9(d), we now provide a more quantitative study of the impact of design size on the probability of routing failure. That is, given similar placement quality (measured by the size-normalized number of neighbor-swaps from a mesh-like placement), we examine the relation between design size and the probability of routing failure.

As reviewed in Section II-B, many metrics have been proposed to estimate routability, such as pin density [3], pin shape [32], wire density [29], Rent exponent [35], net range [22], and the number of incoming/outgoing edges [4]. We study two metrics that have been closely related to routing failures in the recent work of [4]. The first metric is the sum of edge distances (ED), i.e., the sum of half-perimeter wirelengths of nets corresponding to lattice edges initially in a given window. (An $S \times S$ window of the initial square mesh contains $2 \cdot S \cdot (S-1)$ lattice edges.) The second metric is crossing count (CC), which is the total number of nets that *cross* a given local window. In other words, nets having positive-area overlap with a given window are counted in CC (overlap only along the window boundary is not counted). Fig. 13 shows examples of edge mapping, ED and CC. We perform exponentiation on transition matrices to estimate the ED and CC changes after neighbor-swaps, which further provide an estimation for the probability of routing failures [i.e., ED (or CC) is greater than a threshold $\text{ED}_{\text{th}}$ (or $\text{CC}_{\text{th}}$)].

We describe our transition matrix-based estimation as follows. Given a mesh-like placement (where an edge exists between each pair of neighboring instances) with size $(N \times N)$, and number of neighbor-swaps $(K \times N^2)$, we estimate the expected ED within the hotspot based on a transition matrix (i.e., a matrix used to describe the transition of iterative

**Algorithm 2** Create Transition Matrix (Exact), $M_T$

1: $M_T \leftarrow N^4 \times N^4$ zero matrix
2: **for** $p_1 := 1$ to $N$, $q_1 := 1$ to $N$, $p_2 := 1$ to $N$, $q_2 := 1$ to $N$, $p_3 := 1$ to $N$, $q_3 := 1$ to $N$, $p_4 := 1$ to $N$, $q_4 := 1$ to $N$ **do**
3: **if** $(((p_1, q_1) == (p_3, q_3)$ && isNeighbor$((p_2, q_2), (p_4, q_4)))$ || $((p_2, q_2) == (p_4, q_4)$ && isNeighbor$((p_1, q_1), (p_3, q_3)))$ || $((p_1, q_1) == (p_4, q_4)$ && $(p_2, q_2) == (p_3, q_3)$ && isNeighbor$((p_1, q_1), (p_2, q_2))$ && $((p_1, q_1) \neq (p_2, q_2)$ || $(p_3, q_3) \neq (p_4, q_4)))$ **then**
4: $M_T[N^3 \cdot p_1 + N^2 \cdot q_1 + N \cdot p_2 + q_2][N^3 \cdot p_3 + N^2 \cdot q_3 + N \cdot p_4 + q_4] \leftarrow 1$
5: **end if**
6: **end for**
7: **for** $i := 1$ to $N^4$ **do**
8: $rowSum \leftarrow \sum_j M_T[i][j]$
9: $M_T[i][i] \leftarrow 2 \cdot N \cdot (N-1) - rowSum$
10: **end for**
11: $M_T \leftarrow M_T/(2 \cdot N \cdot (N-1))$

**Algorithm 3** Expected Edge Distance (Approximation)

1: $M_D \leftarrow$ create distance matrix
2: $M_T \leftarrow$ create transition matrix
3: $M_E \leftarrow$ exponentiate transition matrix $M_T$ to the $K^{th}$ power
4: $dist \leftarrow$ calculate expected edge distance
5: **return** $dist$

neighbor-swaps, where $M_T[i][j]$ is the probability that a vertex moves from location $i$ to location $j$ in a single neighbor-swap [2]). Specifically, from a given matrix that is the exponentiation (to the $K$th power) of a given transition matrix, we compute the ED and CC over a given hotspot region.

Algorithm 2 describes the transition matrix construction. Based on the transition matrix, we are able to estimate the probability of *mapping* an edge $[(p_1, q_1), (p_2, q_2)]$ to another edge $[(p_3, q_3), (p_4, q_4)]$ [i.e., embedded in the matrix as the entry in the $(N^3 \cdot p_1 + N^2 \cdot q_1 + N \cdot p_2 + q_2)$th row and the $(N^3 \cdot p_3 + N^2 \cdot q_3 + N \cdot p_4 + q_4)$th column]. We then multiply such probabilities by the Manhattan length of the (post-neighbor-swapping) mapped edge to achieve an estimation of the expected ED. Lastly, we take the sum over expected EDs of the lattice edges (i.e., edges between adjacently placed cells in the initial mesh). In line 9, we set $M_T[i][i]$ to the difference between $2 \cdot N \cdot (N-1)$ and *rowSum* so that when we normalize the rows by $2 \cdot N \cdot (N-1)$ (line 11), every entry will represent a marginal probability of transitioning to that state.[23] However, due to the large size of the transition matrix $O(N^4)$, the runtime complexity of the dense matrix exponentiation is $O(N^{4k})$, where $O(N^k)$ = matrix multiplication complexity. Thus, the complexity of the sparse matrix exponentiation is $O(N^8)$.

To reduce the runtime complexity, we propose an *approximate calculation approach*, shown in Algorithm 3. The transition matrix in Algorithm 3 is of size $N^2 \times N^2$ as opposed to $N^4 \times N^4$ in Algorithm 2, because it keeps track of the mapped vertex locations as opposed to the mapped edges. Thus, Algorithm 3 assumes that the mapped vertex locations are independent, which is intuitively reasonable given that the expected number of neighbor-swaps for each vertex is still the same. In Algorithm 3, we first create a distance matrix $M_D$ of size $N^2 \times N^2$, where $M_D[N \cdot p_1 + q_1][N \cdot p_2 + q_2]$ is the Manhattan distance between $(p_1, q_1)$ and $(p_2, q_2)$ in the mesh-like placement (line 1). We then create a transition matrix $M_T$ (line 2), as presented in Algorithm 4. We exponentiate the transition matrix to the $K$th power, to approximate the resultant placement after $K$ moves (line 3). Lastly, according to $M_E$, we calculate the expected ED, as summarized in Algorithm 5. The runtime

[23]The normalization factor is $2N(N-1)$ because this is the total number of events or possible neighbor-swaps that can occur. Thus, in line 11, we divide the matrix through by $2N(N-1)$ to obtain a transition matrix in which each row sums to 1.

**Algorithm 4** Create Transition Matrix (Approximation)

1: $M_T \leftarrow N^2 \times N^2$ zero matrix
2: **for** $p_1 \coloneqq 1$ to $N$, $q_1 \coloneqq 1$ to $N$, $p_2 \coloneqq 1$ to $N$, $q_2 \coloneqq 1$ to $N$ **do**
3: **if** isNeighbor($(p_1, q_1)$, $(p_2, q_2)$) **then**
4: $M_T[N \cdot p_1 + q_1][N \cdot p_2 + q_2] \leftarrow 1$
5: **end if**
6: **end for**
7: **for** $i \coloneqq 1$ to $N^2$ **do**
8: $rowSum \leftarrow \sum_j M_T[i][j]$
9: $M_T[i][i] \leftarrow 2 \cdot N \cdot (N-1) - rowSum$
10: **end for**
11: $M_T \leftarrow M_T/(2 \cdot N \cdot (N-1))$

**Algorithm 5** Calculate Expected Edge Distance

1: $dist \leftarrow 0$
2: **for all** edges $((p_1, q_1), (p_2, q_2))$ **do**
3: **for** $i \coloneqq 1$ to $N^2$, $j \coloneqq 1$ to $N^2$ **do**
4: $dist$ += $M_E[N \cdot p_1 + q_1][i] \cdot M_E[N \cdot p_2 + q_2][j] \cdot M_D[i][j]$
5: **end for**
6: **end for**
7: **return** $dist$

**Algorithm 6** Expected Crossing Count Approximation

1: $M_T \leftarrow$ create transition matrix
2: $M_E \leftarrow$ exponentiate transition matrix $M_T$ to the $K^{th}$ power
3: $count \leftarrow$ calculate expected crossing count
4: **return** $count$

complexities for construction of the distance matrix (line 1), construction of the transition matrix (line 2), exponentiation of the transition matrix (line 3) and calculation of the expected EDs (line 4) are, respectively, $O(N^4)$, $O(N^4)$, $O(N^6 \cdot \log(K))$ and $O(N^6)$.[24] Overall, the runtime complexity of our procedure is $O(N^6 \cdot \log(K))$.

Similarly, Algorithm 6 describes our approximate calculation of CC. Algorithms 4 and 7, respectively, describe the transition matrix construction and expected CC calculation. To calculate CC, in Algorithm 7 line 4, the variable *count* contains the expected number of edges, where we add in the probability that $[(p_1, q_1), (p_2, q_2)]$ is mapped to $[(p_3, q_3), (p_4, q_4)]$ by assuming independence and multiplying the probability that $(p_1, q_1)$ is mapped to $(p_3, q_3)$ by the probability that $(p_2, q_2)$ is mapped to $(p_4, q_4)$. Furthermore, for every pair of undirected edges, we add the probability of that mapping four times.[25] Specifically, we count the edge mapping of $(v_1, v_2)$ to $(v_3, v_4)$ once for each of the two edge permutations of $(v_1, v_2)$ and once for each of the two mapped edge permutations of $(v_3, v_4)$. So, we count this probability $2 \cdot 2 = 4$ times; we then divide by 4 to eliminate double counting (line 7). We note that for a large $K$ value, the approximated ED and CC should be off by a factor of $(N^2)/(N^2-1)$ with respect to the exact values. This is because the approximation algorithms count the set of degenerate edges.[26]

Based on the transition matrix exponentiation, we estimate the probabilities of ED $\geq$ $\text{ED}_{\text{th}}$ and CC $\geq$ $\text{CC}_{\text{th}}$ for an $S \times S$ hotspot ($\text{ED}_{\text{th}}$ and $\text{CC}_{\text{th}}$ values are normalized to $N^2$) as follows. For ED, we assume a Gaussian distribution for the distribution of ED after $K \times N^2$ neighbor-swaps. Using the transition matrix exponentiation, we thus find the mean and variance

[24]We use Python np.linalg library to perform matrix exponentiation.

[25]Here, an undirected edge is an unordered pair of vertices (e.g., edge $[(p_1, q_1), (p_2, q_2)]$ in Fig. 13(a)). In our constructed mesh placement, there can be at most one net between two instances, and we therefore, treat each net as an undirected edge in our analyses.

[26]There are $(N^2)(N^2-1)$ total edges, and there are $(N^2)(N^2)$ ordered pairs of vertices, which the approximation algorithm counts. Therefore, for large $K$, we expect to be off by around a factor of $(N^2)/(N^2-1)$ with respect to the exact values.

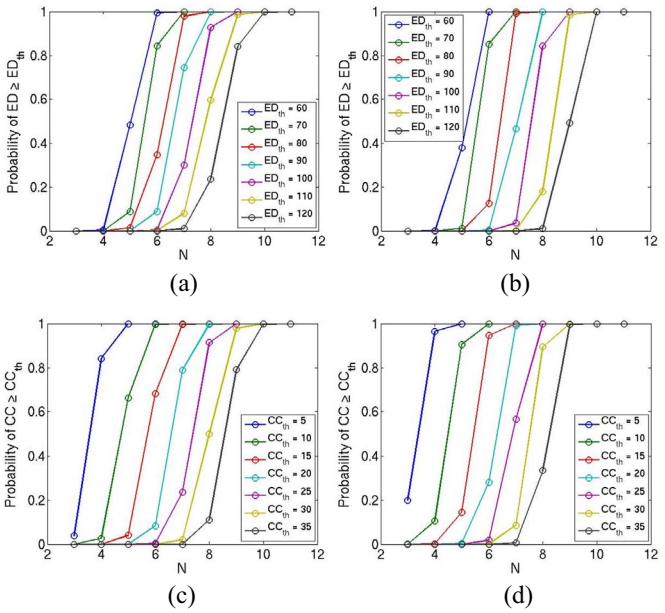



Fig. 14. Probability of ED $\geq$ $\text{ED}_{\text{th}}$ increases with $N$. Shown: $K = 50$ based on (a) Monte Carlo simulation and (b) transition matrix exponentiation; probability of CC $\geq$ $\text{CC}_{\text{th}}$ increases with $N$. Shown: $K = 50$ based on (c) Monte Carlo simulation and (d) transition matrix exponentiation.

**Algorithm 7** Calculate Crossing Count

1: $count \leftarrow 0$
2: **for all** edge $((p_1, q_1), (p_2, q_2))$ **do**
3: **for all** mapped edge $((p_3, q_3), (p_4, q_4))$ that overlap with the given $S \times S$ hotspot **do**
4: $count$ += $M_E[N \cdot p_1 + q_1][N \cdot p_3 + q_3] \cdot M_E[N \cdot p_2 + q_2][N \cdot p_4 + q_4]$
5: **end for**
6: **end for**
7: $count \leftarrow count/4$
8: **return** $count$

in distance for each lattice edge. We then treat ED for each $S \times S$ window as the sum of Gaussian random variables—the lattice edges composing the window—which gives a Gaussian distribution with mean $= \sum_{\text{lattice edges}}\{\text{means}\}$ and variance $= \sum_{\text{lattice edges}}\{\text{variances}\}$. We further assume that the window probabilities are independent. Finally, we can approximate the probability that ED $\geq$ $\text{ED}_{\text{th}}$. For CC, we assume that the probabilities that each of the lattice edges is mapped to an edge that crosses a given fixed window are independent, i.e., each mapped edge's respective crossing of the given window is an independent Bernoulli-distributed random variable. Accordingly, the distribution for the number of crossers for a fixed window (i.e., the number of edges crossing over a fixed window) can be represented as a sum of independent Bernoulli random variables, which is the Poisson Binomial distribution [9]. This is then approximated using the Poisson distribution, with error bound of $9 * \max_{i \in \text{windows}} p_i$, as described in [12].

Figs. 14 shows the probabilities of ED $\geq$ $\text{ED}_{\text{th}}$ and CC $\geq$ $\text{CC}_{\text{th}}$ for $S = 2$ in an $N \times N$ mesh-like placement with $(K = 50) \cdot N^2$ neighbor-swaps, based on both Monte Carlo simulations (with 500 trials) and transition matrix-based estimations.[27] We observe that the probabilities of having some window's ED

[27]The small discrepancy between the results from Monte Carlo simulations versus those from transition matrix-based estimations is due to errors of the approximation from the Normal and Poisson distributions. Note that in this section, we use mesh placements, which are more general and can be modeled using the Markov transition matrix technique, instead of real placements.

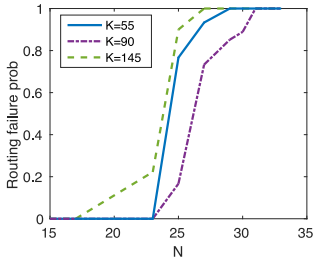



Fig. 15. Routing failure probabilities versus $N$ for $K = 55$, 90, and 145.

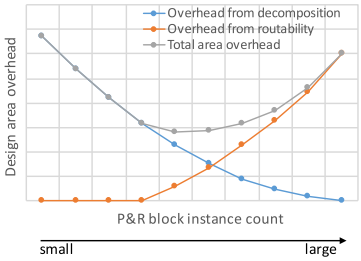



Fig. 16. Hypothesized tradeoff between overall design area and instance count in individual P&R blocks. The $x$-axis shows block instance count, and the $y$-axis shows overall design area overheads induced by 1) decomposition and 2) difficulty of routing.

or CC value larger than given thresholds increase with $N$ (where $N \times N$ indicates the design size). This observation can also be confirmed by application of the Pigeonhole Principle. Suppose we have $N_1 < N_2$ with corresponding neighbor-swaps $J_1 = K \cdot N_1^2$ and $J_2 = K \cdot N_2^2$, respectively, ($J_1$ and $J_2$ are absolute values of #neighbor-swaps, and $K$ is a normalized #neighbor-swaps). By the Pigeonhole Principle, we can find an $N_1 \times N_1$ window within the $N_2 \times N_2$ mesh-like placement that has at least $J_1$ neighbor-swaps performed on it. Thus, for each window of this $N_1 \times N_1$ design, the probability of exceeding both $\text{ED}_{\text{th}}$ and $\text{CC}_{\text{th}}$ will be at least as large as that of the $N_1 \times N_1$ mesh-like placement. This observation indicates that with the same placement quality (i.e., normalized $K$ value with respect to $N^2$), a larger design is more vulnerable to routing failures.

We also perform a similar study on mesh-like placements. Specifically, we perform some normalized number of neighbor-swaps ($K$) with respect to the design size (i.e., $N^2$), and sweep the value of $N$ to study how routing failure probability changes according to $N$. For each pair ($N$, $K$), we perform ten trials of perturbation and routing, and then report the probability of routing failures. Fig. 15 shows our results, which empirically support the above analysis that the probability of routing failure increases with the design size, when placement quality as captured by normalized $K$ is kept constant.

## VI. Conclusion

In this paper, we propose a systematic framework that measures routing capacity of BEOL stack options as well as inherent capability of routers and, potentially, placers. Using our framework, we demonstrate a "quasi-universal" rank-ordering of various BEOL stacks by routing capacities, for a given router. We also study the relationship between routing hotspot size and placement quality. Lastly, we present an analytical study based on exponentiation of a Markov transition matrix to demonstrate how, with the same placement density and quality, a larger design is more likely to experience routing failures, an observation which is supported by empirical data.

This last observation, that the probability of routing failure increases with the design size (i.e., $N \times N$), is particularly intriguing. This may indicate that there might be an "optimal block size" for dividing an SoC into hard macros for a given design enablement, in a similar spirit to how Sylvester and Keutzer [31] proposed 50K to 100K gates as an optimal size of place-and-route blocks nearly 20 years ago.[28] For example, Fig. 16 shows our notional (hypothesized) tradeoff between design area and block size, which is derived based on several simple assumptions. The $x$-axis shows block size normalized to the entire chip area, and the $y$-axis shows area penalty induced by decomposition (i.e., loss of global optimization in hierarchical physical design relative to flat physical design) and/or routability. When the block size is too large, the probability of routing failure is high, and we will end up with a lower utilization to avoid routing failure (orange curve). On the other hand, if the block size is too small, we will have to pay for the cost of partitioning, i.e., loss of optimality due to decomposition (blue curve). To derive the area penalty from routing failure, we first derive a linear model for $\Delta$ utilization as a function of $K_{\text{th}} - K$ from the result in Fig. 7 (the routing failure is directly related to maximum achievable utilization, as shown in Section IV-C). We then assume that $K_{\text{th}}$ is inversely proportional to the square root of the normalized block area, and $K$ is the same for all block sizes (same quality of placements). To derive the area penalty from decomposition, we simply assume that the blocks are connected as clique model, and there is a certain amount of area overhead for each cut. The minimum total area penalty point is shown as the local minimum in the gray curve. According to such a simple model we see in Fig. 16 that there may be a choice of block size for hard macros in hierarchical design that best compromises between routing failure probability and the cost of partitioning.[29]

Open questions for future researchers might include the following.

1) Extension of our framework to comprehend realistic design considerations such as timing, power, manufacturing cost, complex design rules, multiheight cells, and power delivery network requirements.
2) Estimation of "equivalent $K$" (or another metric to assess and/or rank (windows of) placements with respect to routability) in arbitrary real placements.
3) An analytical model to predict optimal block size for minimum area penalty (considering both cost of decomposition and routability) with a given performance (and, possibly, design turnaround time) requirement.

## References

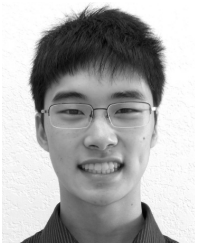
[1] C. J. Alpert, G.-J. Nam, and P. G. Villarrubia, "Effective free space management for cut-based placement via analytical constraint generation," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 22, no. 10, pp. 1343–1353, Oct. 2003.

[2] S. Asmussen, *Applied Probability and Queues*. New York, NY, USA: Springer, 2003. [Online]. Available: http://www.springer.com/us/book/9780387002118

[28] The landmark paper of [31] arrived at its predicted block size based on entirely different justifications, namely, scaling of interconnect RC delay and noise, and of gate drive and leakage. The envisioned size of P&R blocks in a hierarchical physical design methodology has evolved differently from the prediction of [31].

[29] Here, this paper mainly considers routing and corresponding impacts on timing and power. The actual best choice of block size for hard macros would be dependent on many other aspects of designs and design enablements, e.g., standard cell layouts, netlists, and the characteristics of placers and routers.

[3] U. Brenner and A. Rohe, "An effective congestion driven placement framework," in *Proc. ISPD*, Del Mar, CA, USA, 2002, pp. 6–11.
[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," in *Proc. ICCD*, Scottsdale, AZ, USA, 2016, pp. 41–48.
[5] C.-C. Chang, J. Cong, and M. Xie, "Optimality and scalability study of existing placement algorithms," in *Proc. ASP DAC*, 2003, pp. 621–627.
[6] J. Cong, M. Romesis, and M. Xie, "Optimality, scalability and stability study of partitioning and placement algorithms," in *Proc. ISPD*, Monterey, CA, USA, 2003, pp. 88–94.
[7] J. Darnauer and W. W.-M. Dai, "A method for generation random circuits and its application to routability measurement," in *Proc. FPGA*, Monterey, CA, USA, 1996, pp. 66–72.
[8] X. Dong, J. Zhao, and Y. Xie, "Fabrication cost analysis and cost-aware design space exploration for 3-D ICs," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 29, no. 12, pp. 1959–1972, Dec. 2010.
[9] M. Fernandez and S. Williams, "Closed-form expression for the Poisson-binomial probability density function," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 46, no. 2, pp. 803–817, Apr. 2010.
[10] L. W. Hagen, D. J.-H. Huang, and A. B. Kahng, "Quantified suboptimality of VLSI layout heuristics," in *Proc. DAC*, San Francisco, CA, USA, 1995, pp. 216–221.
[11] X. He *et al.*, "Ripple 2.0: High quality routability-driven placement via global router integration," in *Proc. DAC*, Austin, TX, USA, 2013, pp. 1–6.
[12] J. L. Hodges and L. L. Cam, "The Poisson approximation to the Poisson binomial distribution," *Ann. Math. Stat.*, vol. 31, no. 3, pp. 737–740, 1960.
[13] J. Hu, J. A. Roy, and I. L. Markov, "Completing high-quality global routes," in *Proc. ISPD*, San Francisco, CA, USA, 2010, pp. 35–41.
[14] M. D. Hutton, J. Rose, J. P. Grossman, and D. G. Corneil, "Characterization and parameterized generation of synthetic combinational benchmark circuits," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 17, no. 10, pp. 985–996, Oct. 1998.
[15] P. Gupta, A. B. Kahng, A. Kasibhatla, and P. Sharma, "Eyecharts: Constructive benchmarking of gate sizing heuristics," in *Proc. DAC*, Anaheim, CA, USA, 2010, pp. 592–602.
[16] A. B. Kahng and S. Kang, "Construction of realistic gate sizing benchmarks with known optimal solutions," in *Proc. ISPD*, Napa, CA, USA, 2012, pp. 153–160.
[17] A. B. Kahng and S. Reda, "Evaluation of placer suboptimality via zero-change netlist transformations," in *Proc. ISPD*, San Francisco, CA, USA, 2005, pp. 208–215.
[18] A. B. Kahng and X. Xu, "Accurate pseudo-constructive wirelength and congestion estimation," in *Proc. SLIP*, Monterey, CA, USA, 2003, pp. 61–68.
[19] M.-C. Kim, J. Hu, D.-J. Lee, and I. L. Markov, "A SimPLR method for routability-driven placement," in *Proc. ICCAD*, San Jose, CA, USA, 2011, pp. 67–73.
[20] B. S. Landman and R. L. Russo, "On a pin versus block relationship for partitions of logic graphs," *IEEE Trans. Comput.*, vol. C-20, no. 12, pp. 1469–1479, Dec. 1971.
[21] W.-H. Liu, T.-K. Chien, and T.-C. Wang, "Region-based and panel-based algorithms for unroutable placement recognition," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 34, no. 4, pp. 502–514, Apr. 2015.
[22] Q. Liu and M. Marek-Sadowska, "Pre-layout wire length and congestion estimation," in *Proc. DAC*, San Diego, CA, USA, 2004, pp. 582–587.
[23] W.-H. Liu *et al.*, "Routing congestion estimation with real design constraints," in *Proc. DAC*, Austin, TX, USA, 2013, pp. 1–8.
[24] M. Pan and C. Chu, "FastRoute 2.0: A high-quality and efficient global router," in *Proc. ASP DAC*, Yokohama, Japan, 2007, pp. 250–255.
[25] M. Pan and C. Chu, "IPR: An integrated placement and routing algorithm," in *Proc. DAC*, San Diego, CA, USA, 2007, pp. 59–62.
[26] Z. Qi, Y. Cai, and Q. Zhou, "Accurate prediction of detailed routing congestion using supervised data learning," in *Proc. ICCD*, Seoul, South Korea, 2014, pp. 97–103.
[27] M. Saeedi, M. S. Zamani, and A. Jahanian, "Prediction and reduction of routing congestion," in *Proc. ISPD*, San Jose, CA, USA, 2006, pp. 72–77.
[28] B. Sluijk *et al.*, *Personal Communication*, ASML, Veldhoven, The Netherlands, Dec. 2015.
[29] P. Spindler and F. M. Johannes, "Fast and accurate routing demand estimation for efficient routability-driven placement," in *Proc. DATE*, Nice, France, 2007, pp. 1226–1231.
[30] D. Stroobandt, P. Verplaetse, and J. Van Campenhout, "Generating synthetic benchmark circuits for evaluating CAD tools," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 19, no. 9, pp. 1011–1022, Sep. 2000.
[31] D. Sylvester and K. Keutzer, "Getting to the bottom of deep submicron," in *Proc. ICCAD*, San Jose, CA, USA, 1998, pp. 203–211.
[32] T. Taghavi *et al.*, "New placement prediction and mitigation techniques for local routing congestion," in *Proc. ICCAD*, San Jose, CA, USA, 2010, pp. 621–624.
[33] Y. Wei *et al.*, "GLARE: Global and local wiring aware routability evaluation," in *Proc. DAC*, San Francisco, CA, USA, 2012, pp. 768–773.
[34] J. Westra, C. Bartels, and P. Groeneveld, "Probabilistic congestion prediction," in *Proc. ISPD*, Phoenix, AZ, USA, 2004, pp. 204–209.
[35] X. Yang, R. Kastner, and M. Sarrafzadeh, "Congestion estimation during top-down placement," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 21, no. 1, pp. 72–80, Jan. 2002.
[36] Q. Zhou *et al.*, "An accurate detailed routing routability prediction model in placement," in *Proc. ASQED*, 2015, pp. 119–122.
[37] *LEF/DEF 5.7 Reference*. Accessed: Aug. 30, 2017. [Online]. Available: http://www.si2.org/openeda.si2.org/projects/lefdef
[38] *OpenCores: Open Source IP-Cores*. Accessed: Aug. 30, 2017. [Online]. Available: http://www.opencores.org
[39] *PROBE Website*. Accessed: Aug. 30, 2017. [Online]. Available: http://vlsicad.ucsd.edu/PROBE
[40] *Synopsys Design Compiler User Guide*. Accessed: Aug. 30, 2017. [Online]. Available: http://www.synopsys.com



**Alex Kahng** is currently pursuing the B.A. degree in computer science and mathematics with Harvard University, Cambridge, MA, USA.

His current research interests include computation theory, chip design, and machine learning.

**Andrew B. Kahng**, photograph and biography not available at the time of publication.

**Hyein Lee** (S'12) received the B.S. degree in electrical engineering from Yonsei University, Seoul, South Korea and the M.S. degree in electrical engineering from the Korea Advanced Institute of Science and Technology, Daejeon, South Korea, in 2009. She is currently pursuing the Ph.D. degree with VLSI CAD Laboratory, University of California at San Diego, San Diego, CA, USA.

From 2009 to 2012, she was with the Design Technology Team, Samsung Electronics, Suwon, South Korea. Her current research interests include low-power design optimization and design-manufacturing interface.

**Jiajia Li** (S'12) received the B.S. degree in software engineering from Shenzhen University, Shenzhen, China, in 2011 and the M.S. degree in electrical engineering from the University of California at San Diego, San Diego, CA, USA, in 2013, where he is currently pursuing the Ph.D. degree with the University of California at San Diego.

He joined VLSI CAD Laboratory, University of California at San Diego in 2012. His current research interests include physical design and sign-off optimization, margin reduction, and low-power design.