PROBE2.0: A Systematic Framework for Routability Assessment From Technology to Design in Advanced Nodes

Chung-Kuan Cheng, *Life Fellow, IEEE*, Andrew B. Kahng, *Fellow, IEEE*, Hayoung Kim, *Member, IEEE*, Minsoo Kim, *Graduate Student Member, IEEE*, Daeyeal Lee, *Student Member, IEEE*, Dongwon Park, *Student Member, IEEE*, and Mingyu Woo, *Student Member, IEEE*

***Abstract*—In advanced nodes, scaling of critical dimension and pitch has not progressed at historical Moore's Law rates. Thus, *scaling boosters* are explored to improve achievable power, performance, area, and cost (PPAC) in new technologies. However, scaling boosters increase complexity of standard-cell architectures, power delivery, design rules, and other aspects of the design enablement, and may not result in design-level benefits. Therefore, design-technology co-optimization (DTCO) methodologies are required to evaluate design-level benefits of scaling boosters. The key challenge for DTCO is that large engineering efforts and long timelines are needed to develop design enablements (e.g., cell libraries) and perform implementation studies in order to assess technology options. We describe a new framework that can systematically evaluate a measure of intrinsic routability, K_{th}, across both technology and design choices. We focus on routability since it is a critical factor in the scaling of area and cost. Our framework includes realistic standard-cell libraries that are automatically generated using satisfiability modulo theory (SMT) methods, and a new pin shape selection method. Routability assessments are based on the PROBE approach and an improved construction of underlying netlist topologies. Our experimental studies demonstrate the assessment of routability impacts for advanced-node technology and design options. We demonstrate learning-based K_{th} prediction to reduce runtime, disk space and commercial tool licenses needed to implement our framework. Our work enables faster and more comprehensive evaluation of technology options early in the technology development process.**

***Index Terms*—Design enablement, design-technology co-optimization (DTCO), machine learning (ML), pathfinding, place-and-route (P&R), routability, standard cell, VLSI CAD.**

Manuscript received August 2, 2020; revised January 14, 2021 and April 27, 2021; accepted June 21, 2021. Date of publication June 29, 2021; date of current version April 21, 2022. This work was supported in part by Defense Advanced Research Projects Agency (DARPA) under Grant HR0011-18-2-0032; in part by NSF under Grant CCF-1564302; in part by Qualcomm; in part by Samsung Electronics; in part by NXP Semiconductors; in part by Mentor Graphics; and in part by the C-DEN Center. This article was recommended by Associate Editor D. Z. Pan. *(Corresponding author: Minsoo Kim.)*

Chung-Kuan Cheng is with the Department of Computer Science and Engineering, University of California at San Diego, La Jolla, CA 92093 USA.

Andrew B. Kahng is with the Department of Computer Science and Engineering and the Department of Electrical and Computer Engineering, University of California at San Diego, La Jolla, CA 92093 USA.

Hayoung Kim is with the Device Solutions Disivion, Samsung Electronics Company, Ltd., Hwaseong 18448, South Korea.

Minsoo Kim, Daeyeal Lee, Dongwon Park, and Mingyu Woo are with the Department of Electrical and Computer Engineering, University of California at San Diego, La Jolla, CA 92093 USA (e-mail: mik226@ucsd.edu).

Digital Object Identifier 10.1109/TCAD.2021.3093015

I. Introduction

With the slowdown of pitch scaling in advanced technology nodes, *scaling boosters* (e.g., buried power rails (BPRs) [33], backside power delivery networks (PDNs) [4] and supervias [14]) have become critical to improve power, performance, area, and cost (PPAC) in future technologies. However, scaling boosters have impacts on cell architecture, design rules, and other aspects of the design enablement. Thus, holistic optimizations between process technology and chip design are required. Design-technology co-optimization (DTCO) has therefore emerged as the key methodology to decide which scaling boosters enter mass production.

DTCO has three main stages: 1) technology; 2) design enablement; and 3) design. As shown in Fig. 1, the *technology stage* includes modeling and simulation methodologies related to process and device technology; these span technology CAD (TCAD), optical proximity correction (OPC), design rules and SPICE models. The *design enablement stage* includes other required inputs to the design process, such as standard-cell libraries, IPs, and signoff environments. Last, the *design stage* covers front-end design, logic synthesis, place-and-route (P&R), parasitic extraction, static timing analysis, and physical verifications (design rule check, layout versus schematic check). For the purposes of this article, technology and design enablement together enable the IC design process. Thus, in the remainder of our discussion we use "technology" to encompass the union of the technology and the design enablement stages, and "design" to refer to the design stage.

The key challenge in today's DTCO is that weeks or months are needed for feedback from design back to technology. This is due to the cost of creating prototype design enablements and performing design experiments. Our contribution lies in automating and greatly speeding up this feedback loop, enabling assessment of hundreds of technology options within days.

Density is the overarching metric for enablement of system-level benefits through scaling [40], and directly determines the Area and Cost aspects of PPAC. In our work, we focus on *routability* implications of technology choices, since routability and density are intimately tied together. The challenge of routability arises as back-end-of-line (BEOL, i.e., metal

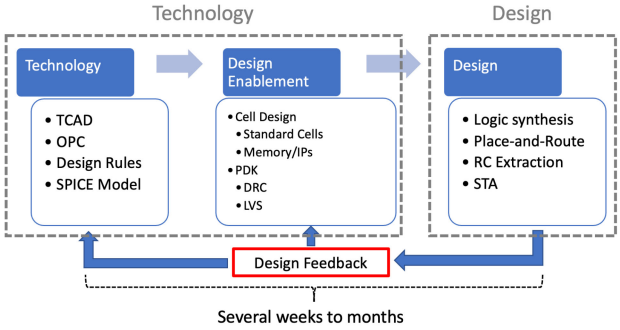

Fig. 1. Three main stages of DTCO. Today, the design feedback loop takes several weeks to months.

layers) technology fails to scale down in step with front-end-of-line (FEOL, i.e., device layers). Also, standard-cell heights are a crucial lever for density scaling [35], but small standard-cell heights challenge area routing and pin accessibility. Furthermore, high BEOL resistances require denser PDNs, which occupy more routing resources and harm routability.

Contributions of This Work: The goal of our work is to enable faster and more comprehensive evaluation of technology options, early in the technology development process. In particular, we describe a framework for *systematic assessment of routability* across the combined space of technology options and design enablement options.

Many measures of routability have been developed and applied over the past decades. These span the use of congestion maps, metrics of pin accessibility, machine learning (ML)-based congestion predictors, and other techniques, as we review in Section II below. However, these previous methods to assess routability do not solve two root causes of the long feedback loop in DTCO (Fig. 1). The first root cause is that efficient simultaneous exploration of technology and design options in DTCO is blocked by the effort and expense of the design enablement stage. Producing layouts and characterizations for standard-cell libraries requires an enormous amount of engineering cost and time, due to complex constraints, such as transistor-level placement, in-cell routing, and pin accessibility. Today's DTCO relies on limited, heuristic layout synthesis (e.g., manual layout of 15-60 key cells) to assess a given set of technology options. The second root cause is that routability assessment methods have mostly focused on assessing design *implementations* (e.g., to predict routability of the placement of a particular netlist), rather than assessing *design enablements*.

Our present work attacks both of the above-mentioned root causes of long design feedback loops in DTCO. To do this, we build on two threads of recent works: 1) automatic standard-cell layout generation using satisfiability modulo theory (SMT) solvers [8], [25] and 2) intrinsic routability assessment of BEOL stack options via the K_{th} metric [20]. We review these works in Section II. Our framework is able to provide assessments of intrinsic routability across a range of technology and design parameters reflecting sub-7-nm technologies. Our main contributions are summarized as follows.

1) We describe a systematic and complete framework to evaluate routability across key parameters of technology and design. Our framework is generalizable and flexible; it enables rapid evaluation of hundreds of technology and design enablement options within hours or days, providing a valuable tool for early technology development.
2) We propose a pin shape selection strategy based on the remaining pin access (RPA) [34], along with a top-metal-only pin shape selection strategy, at our design enablement stage. We also extend methods of [8], [25] to automatically produce more realistic standard-cell libraries (LEF format [49]) in terms of power and ground pins, contacted poly pitches (CPPs) and metal pitches.
3) We extend the method of [20] to assess routability across configurations of technology and design, rather than only BEOL stack options. We study both *cell-level* and *design-level* routability, and show advantages of using *knight's tour-based* artificial netlist topology generation in cell-level routability assessment. (A knight's tour is a sequence of knight's moves in a chessboard that visits each square exactly once. Section IV-A below explains its use in our methodology.)
4) We achieve seamless integrations with commercial P&R tooling, via automated generation of power-ground hookups in cell layouts, and routing technology files to reflect modified design rules.
5) We demonstrate accurate learning-based K_{th} prediction that reduces runtime, disk storage and tool license overheads of our framework.

II. Related Work

A. Standard-Cell Layout Generation

Automatic Standard-Cell Layout Generation: Standard-cell layout synthesis can help library design teams explore cell architectures with holistic consideration of transistor placement, in-cell routing, complicated design rules, and pin accessibility. The methods of [15] and [43] provide co-optimization of transistor placement and in-cell routing, but do not consider such aspects as multipatterning design rules that are seen in advanced technology nodes. Van Cleeff *et al.* [9], Jo *et al.* [19], and Li *et al.* [26] proposed standard-cell layout automation frameworks for sub-7-nm technologies, but these works incorporate multiple heuristic approaches with no guarantees of optimality. Lee *et al.* [25] unified transistor-level placement and routing with dynamic pin allocation (DPA), and apply an SMTs solver to achieve optimal layout solutions.

Pin Accessibility-Aware Standard-Cell Layout: One of the most difficult design features for standard-cell layout generation is the pin accessibility, which is challenged by the limited number of tracks and complicated design rules. The works of [32], [42] define metrics for pin accessibility within their objectives for standard-cell layout optimization. Seo *et al.* [34] proposed the RPA metric to capture pin access interference from access points of neighboring pins. Cheng *et al.* [8] devised "at-least-k" Boolean constraints that guarantee a minimum number of pin openings (access points) per pin in the cell layout.

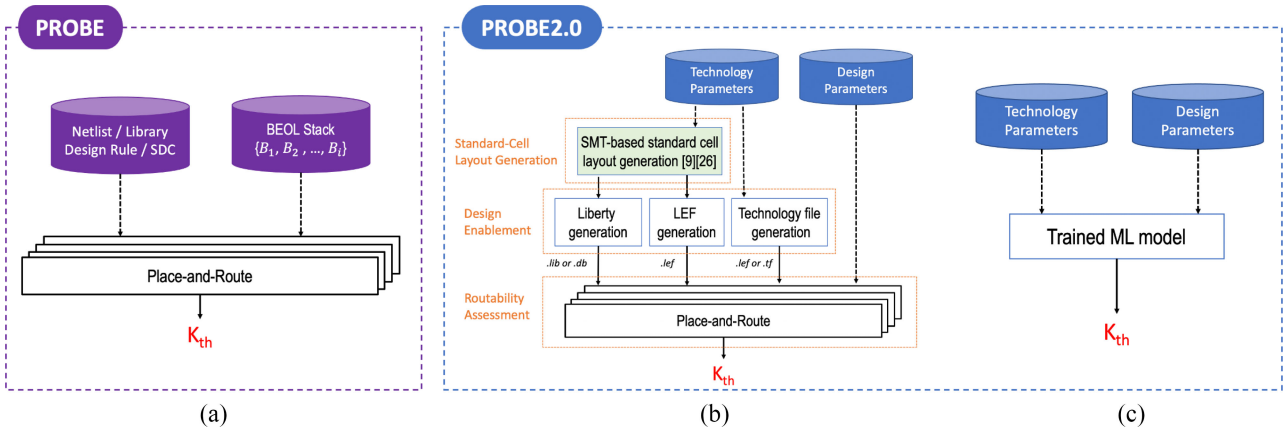

Fig. 2. Overall flows for PROBE [20] and PROBE2.0. (a) PROBE evaluates BEOL stack options (B_i) by performing neighboring cell swaps until routing fails at a (normalized) number of swaps K_{th}. (b) PROBE2.0 flow, including standard-cell layout generation [8], [25], automated design enablement generation, and routability assessment with multiple P&R runs. (c) PROBE2.0 flow using a trained learning-based model to predict K_{th}.

The SMT-Based Standard-Cell Layout Generation of [8], [25]: Our present work builds on the SMT-based parametric standard-cell generation framework of [8] and [25]. This framework takes in three main inputs: 1) *cell architectures:* number of routing tracks and transistor fins, and track pitches; 2) *netlist:* component connectivity of library cell; and 3) *design rules:* parametric conditional design rules depending on cell architecture. Given these inputs, a cell layout is produced that is optimal with respect to cell area, M2 track use (routability) and routed metal length, in lexicographic order of these criteria. A unifying DPA constraint integrates additional design constraints, such as transistor placement, in-cell routing, conditional design rules, and pin accessibility constraints. This yields a constraint satisfaction formulation that produces an optimized cell layout via a single multiobjective optimization.

We observe that these previous works do not provide necessary enablements of commercial standard-cell P&R, such as LEF [49] generation, PDN generation, and routing technology file generation. Nor do these works support control parameters for standard-cell layout generation that are relevant in sub-7-nm nodes. We describe a complete framework to support both standard-cell layout generation and associated P&R enablement. Our layout generation uses RPA-based and top-metal-only pin shape selections to improve pin accessibility.

B. Routability Assessment

Routability is a hard constraint in the modern (fixed-die) P&R context. Thus, many previous works have studied routability-driven placement, as well as ripup-and-reroute methods in global routing. For example, [7], [18], [22], and [27] all propose routability-driven placement based on congestion maps derived from early trial or global routing. Pin accessibility of a given standard-cell instance also affects routability. The work of [34] describes pin accessibility-aware detailed placement based on the RPA metric. However, we do not focus on placement and routing optimizations, but rather on methods for *assessment* of routability.

Routability Analysis and Prediction: Tseng *et al.* [38] proposed a systematic framework with P&R tools to check routability, aiming to improve placement outcomes. The authors propose standard cell-level and placement-level routability scores to generate cell spacing constraints. Han *et al.* [17] proposed an optimal ILP-based detailed router and evaluate feasibility (routability) of routing clips based on an ILP solver. Kang *et al.* [21] and Park *et al.* [29], [30] use Boolean satisfiability (SAT) to analyze routability under conditional design rules. Park *et al.* [29], [30] furthermore extract minimal unsatisfiable subsets to diagnose bottlenecks when designs are proven to be unroutable.

Several recent works propose ML-based routability predictions. Zhuo *et al.* [44] proposed a new routability prediction model based on supervised learning in placement. The works of [6] and [41] predict routability based on convolutional neural networks (CNNs) and support vector machines (SVMs), respectively. Chan *et al.* [2] also proposed SVM-based routability prediction, but aim to evaluate routability for various BEOL stack options.

The PROBE Routability Measurement Utility of [20]: Our present work builds on the "PROBE" framework of [20], which gives a measure of inherent routability of BEOL stack options. PROBE begins with a placement solution that is easy to route—e.g., a regular mesh placement of a mesh netlist topology. PROBE then iteratively swaps the locations of random pairs of neighboring placed cells, progressively "tangling" the placed netlist until the routing fails with more than some threshold number of postroute DRC violations. The number of random neighbor cell swaps performed, normalized to the number of instances in a design, is denoted by K. The number of swaps beyond which routing fails is denoted as the *K threshold* (K_{th}), and captures *intrinsic* routing capacity (e.g., of a given BEOL stack).

Fig. 2(a) shows the scope of PROBE. Given a placement, a set of BEOL stack options $\{B_1, B_2, \ldots, B_i\}$ can be ranked in terms of routability. The framework supports two types of placements, shown in Fig. 3. *Mesh-like* placements do not reflect any specific design; they consist of an array of instances of a given 2- or 3-input cell. Connections are made between neighbors, inducing a near-meshlike netlist topology. *Cell width-regularized* placements are design specific, and are produced by commercial P&R tools. However,

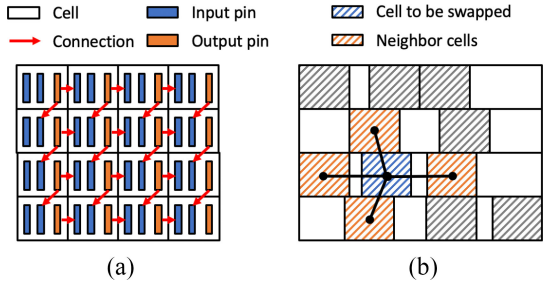

Fig. 3. Placements for PROBE [20]. (a) Mesh-like placement based on a 2-input cell. The red arrows show connections between neighbor instances, inducing a near-meshlike netlist topology. (b) A cell width-regularized placement. The orange-striped cells are considered to be neighbors of the blue-striped cell. The blue-striped cell is swapped with a randomly selected neighbor.

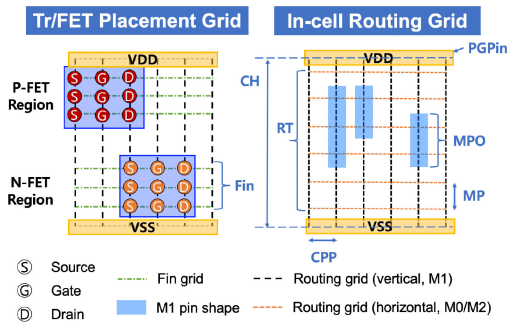

Fig. 4. Grid-based standard-cell architecture and technology parameters for standard cells.

the standard cells in the placements are all given the same cell width to avoid illegal placements after neighbor cell swaps.

We observe that [20] is applied only to BEOL stack options, and does not cover the rich space of FEOL technology and design enablement options. Moreover, the near-meshlike topology can produce only a limited range of routed wirelength (WL) and Rent parameter values that may not match realistic values. The framework we describe below supports DTCO with routability assessment across technology and design enablement options. We use a knight's tour-based construction that can better reflect actual design attributes.

III. PROBE2.0 Framework

Our PROBE2.0 framework is shown in Fig. 2(b). It takes technology and design parameters as primary inputs, and consists of three major stages.

1) The *standard-cell layout generation stage* is based on input technology parameters, and is performed using an extension of an SMT-based standard-cell layout generation [8], [25]. It produces purely grid-based pin locations and cell boundaries.
2) The *design enablement stage* begins with the generated standard-cell layouts, and is also performed according to the input technology parameters. Design enablement generates LEF [49], Liberty [50], and routing technology files. LEF file generation converts the primitive form of layouts to LEF format. The conversion considers real-world constraints for the stability of standard-cell characteristics, as detailed in Section III-D.
3) The *routability assessment stage* uses a *knight's tour-based* topology as well as open-source designs with the PROBE approach [20]. Also, Fig. 2(c) shows how the PROBE2.0 flow can be realized with learning-based K_{th} prediction, where a trained ML model enables more efficient routability assessment.

A. Standard-Cell Architecture

Fig. 4 shows a grid-based standard-cell architecture and technology parameters of standard cells, as used in our work. We follow the 7-nm standard-cell architectures in [10] and [39] to generate the grid-based P&R graph with four layers TS/PC, $M0$, $M1$, and $M2$. TS/PC and $M0$ layers are included in FEOL layers and M1 and M2 layers are included in BEOL layers. Next, we give detailed definitions of the eight technology parameters and five design parameters that PROBE2.0 supports as user inputs.

B. Definitions of Technology Parameters

Technology parameters include various options for process technology as well as design enablement.

1) *Fin:* The number of fins for devices of standard cells. We use 2 and 3 for Fin [10], [39].
2) *CPP:* Contacted poly pitch for standard cells. We use 48 and 54 nm for CPP [35], [39].
3) *MP:* Metal pitch for M2 and M3 routing layers. We use 24, 32, and 40 nm for MP [35], [39].
4) *RT:* The number of available M2 routing tracks for standard cells. We use 4, 5 and 6 for RT [10], [39].
5) *PGpin:* Pin types for power and ground of standard cells. We support three types of power and ground pins: *M1* [39], *M1+M2* [3], and *BPR* [33]. M1 denotes power and ground pins on the M1 layer. M1+M2 denotes power and ground pins on both M1 and M2 layers. BPR has no power and ground pins on BEOL layers. M1 and M2 power and ground pins have width equal to twice the minimum width on Mx (M1, M2 and M3) layers. (Thus, since minimum width and minimum spacing are each equal to half the metal pitch MP, the power and ground pins have widths equal to the Mx pitch.) Also, in enablement of BPR, the width of M1 power pins is the same as the minimum width of Mx. This is because commercial P&R tools require power and ground pins to be connected to PDN. Note that the M1 power and ground pins do not affect routability since the minimum routing layer is M2.
6) *CH:* Cell height of standard cells, expressed as a number of M2 routing track (T) pitches. For example, if M2 routing track pitch is 40 nm and the cell height is 240 nm, then CH is 6. We use and refer to standard cells with 5T, 6T, 7T, and 8T [10], [39] cell heights. Note that CH depends on the combination of RT and PGpin. For example, if RT is 4 and PGpin is M1, then CH is 6. If RT is 4 and PGpin is BPR, then CH is 5.

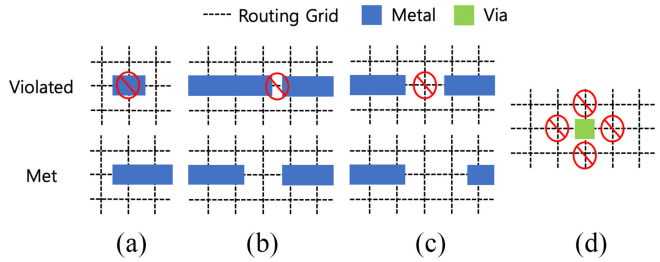

Fig. 5. DR-MAR, DR-EOL, and DR-VR design rules. (a) DR-MAR = 1. (b) DR-EOL = 1. (c) DR-EOL = 2. (d) DR-VR = 1.

7) *MPO:* The number of minimum pin openings (access points) per pin. For example, if MPO is 2, every pin must have at least two access points for the generated standard-cell layout. We use the values of 2 and 3 for MPO [8], [25].
8) *DR:* Design rule sets. In this work, all design rules are defined based on grids. We assume that our technologies are based on extreme ultraviolet (EUV) lithography. We define three fundamental grid-based design rules for Mx layers, *DR-MAR*, *DR-EOL* and *DR-VR* [8], [25]; we also define two design rule *sets*, namely, EUV-loose (EL) and EUV-tight (ET).

Fig. 5 illustrates the three design rules.

1) DR-MAR denotes *minimum area* rules, as shown in Fig. 5(a). When a metal shape occupies only one grid point and DR-MAR is 1, this violates the DR-MAR rule. I.e., a metal shape must be long enough to occupy at least two routing grid points.
2) DR-EOL denotes *end-of-line* rules, as shown in Fig. 5(b) and (c). When edges of two co-linear metal shapes are placed next to each other and DR-EOL is 1, this violates the DR-EOL rule. I.e., there must be at least one unoccupied routing grid point between edges of metal shapes. When DR-EOL is 2, there must be at least two empty routing grid points.
3) DR-VR denotes *via restriction* rules, as shown in Fig. 5(d). The figure shows the prohibited locations for other vias, relative to the placement of a given via. When DR-VR is 1, only four neighbors are blocked by the DR-VR rule.

Our two *design rule sets* each comprise combinations of specific design rule settings, as follows. EL consists of DR-MAR = 1, DR-EOL = 1, and DR-VR = 1. ET consists of DR-MAR = 1, DR-EOL = 2, and DR-VR = 1.

Last, we note that in this work, we assume metal enclosures of vias are 10 nm in a preferred direction and 0 nm in a nonpreferred direction. Also, many practical design rules can be framed using our grid-based design rules. For example, rules for end-to-end spacing and minimum enclosures can be captured with the DR-EOL rule. We note that a wide range of via rules, including center, edge and corner spacing rules, can be captured with the DR-VR rule.

Based on the cell architecture of Fig. 4 and the above technology parameters, our standard-cell layout generation can generate layouts for various cell architectures. In particular, our studies use six types of cell architectures with the combinations of Fin, RT, PGpin, and CH shown in Table I.

TABLE I
STANDARD-CELL ARCHITECTURES IN OUR EXPERIMENTS

Fin	RT	PGpin	CH
2	4	M1/M1+M2	5
		BPR	6
3	5	M1/M1+M2	6
		BPR	7
3	6	M1/M1+M2	7
		BPR	8

TABLE II
FOUR BEOL STACK OPTIONS. R DENOTES *Routing Resource*

BEOL	1X	2X	3.2X	9X	18X	R
9M	3	4	NA	NA	2	104.4
10M	3	4	2	NA	1	117.8
11M	3	4	2	NA	2	120.1
13M	3	2	4	2	2	116.3

TABLE III
DETAILS OF PDN OPTIONS. ALL NUMBERS ARE PITCHES IN UNITS OF μm FOR EACH LAYER. "*P*" INDICATES THAT WE USE A GIVEN LAYER ONLY FOR PDN AT MAXIMUM AREA DENSITY, AND DO NOT ALLOW THE LAYER TO BE USED FOR SIGNAL ROUTING. THE WIDTH OF M5 STRIPES IS 0.96 μm AND THE WIDTH OF M6/M8 STRIPES IS 1.296 μm. THE SPACING BETWEEN VDD AND VSS STRIPES ON 2X AND 3.2X LAYERS IS 0.550 μm

PDN	9M		10M		11M		13M	
Backside	NA		NA		NA		NA	
Sparse	M5	10	M5	10	M5	10	M5	10
	M6	10	M6	10	M6	10	M8	10
	M8	P	M9	P	M10	P	M12	P
	M9	P	M10	P	M11	P	M13	P
Middle	M5	20	M5	20	M5	20	M5	20
	M6	20	M6	20	M6	20	M8	20
	M8	P	M9	P	M10	P	M12	P
	M9	P	M10	P	M11	P	M13	P
Dense	M5	40	M5	40	M5	40	M5	40
	M6	20	M6	40	M6	40	M8	40
	M8	P	M9	P	M10	P	M12	P
	M9	P	M10	P	M11	P	M13	P

C. Definitions of Design Parameters

Our framework uses the following *design parameters*.

1) *BEOL:* Metal stack options. We define *9M*, *10M*, *11M* and *13M* BEOL stack options based on scaling down from a commercial 14-nm technology. Recall that Mx (1X layer) pitch, i.e., MP above, is a technology parameter that we can vary in routability exploration. To scale down a 14-nm BEOL technology to sub-7-nm technologies, we define 2X, 3.2X, 9X, and 18X layer pitches based on 40 nm as the 1X pitch; this reflects advanced-node stacks as well as considerations, such as litho/cost "cliff" (~80nm pitch limit for single-exposure 193i patterning). We calculate the *routing resource* of [20] for each BEOL stack option. The *routing resource* is defined as $\sum_b(1/\text{pitch}_b)$ where b denotes a metal layer and pitch_b is the pitch of b. Essentially, this sums available routing tracks over all routing layers. Table II summarizes layer counts per pitch and *routing resource* (R) for the BEOL stack options.
2) *PDN:* PDN options. These include traditional PDN with different layers and pitch, and backside PDN as a scaling

TABLE IV
TECHNOLOGY AND DESIGN PARAMETERS IN OUR EXPERIMENTS

Type	Parameter	Option
Technology	Fin	2, 3
	CPP	54, 48
	MP	24, 32, 40
	RT	4, 5, 6
	PGpin	BPR, M1, M1+M2
	CH	5, 6, 7, 8
	MPO	2, 3
	DR	EUV-Loose, EUV-Tight
Design	BEOL	9M, 10M, 11M, 13M
	PDN	Backside, Sparse, Middle, Dense
	Tool	Tool A, Tool B
	Util	0.6, 0.7, 0.8
	Design	A knight's tour, AES, LDPC, JPEG, VGA

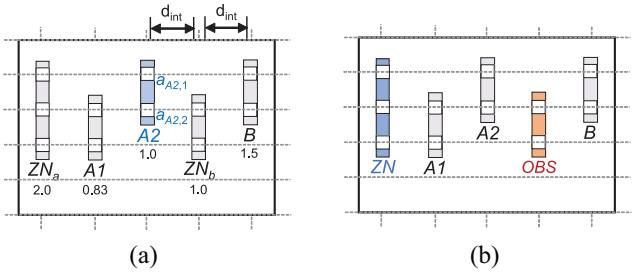

Fig. 6. Example of pin shape selection in a standard cell, with Fin = 2, RT = 5, MPO = 2, and the EL design rule set. (a) Initial layout of OAI21_X1 from SMT-based layout generation, with calculated RPA values of 2.0, 0.83, 1.0, 1.0, and 1.5 for ZN_a, $A1$, $A2$, ZN_b, and B, respectively. (b) OAI21_X1 layout after RPA-based pin shape selection.

booster for advanced technology. We define four PDN options: a) *backside*; b) *sparse*; c) *middle*; and d) *dense*. Table III shows the details of the PDN options.

3) *Tool:* Commercial P&R tools [46], [55], referred to only as *Tool A* and *Tool B* to comply with license agreements.
4) *Util:* Placement utilization (0.6, 0.7, 0.8).
5) *Design:* Designs studied in routability assessment. We use knight's tour-induced artificial topologies, along with four open-source designs (AES, LDPC, JPEG, VGA) from OpenCores [51]. The respective instance counts of AES, LDPC, JPEG, and VGA are approximately 13k, 56k, 69k, and 72k.

Table IV summarizes the technology and design parameters that we use in our experiments. Note that in our framework, the parameter list is flexible and readily extendable. This enables accommodation of new technology requirements or new scaling booster options. For example, sets of smaller CPP and MP values, including with nonunit "gear ratio" values, such as 2:1 or 3:2 (e.g., CPP relative to vertical M1 pitch), can be evaluated with the PROBE2.0 framework. This serves a real DTCO and technology pathfinding problem in industry for sub-3nm technologies, through the end of the lateral scaling roadmap. Also, we can easily add new designs and/or PDN strategies as design parameters, for richer assessments.

D. Design Enablement

Our design enablement produces ready-to-use standard libraries and required inputs for P&R. We generate LEF format from the primitive layout produced by SMT-based cell layout generation. Layouts are fully grid-based, with CPP and MP

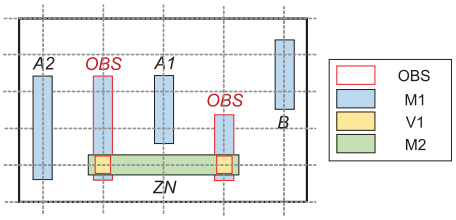

Fig. 7. Example of standard-cell layout with top-metal-only pin shape selection. We show an initial layout of AOI21_X1 with Fin = 2, RT = 5, MPO = 2, and the ET design rule set.

technology parameters defining the grid pitches. Importantly, we propose two pin shape selection schemes: 1) RPA-based pin shape selection and 2) top-metal only pin shape selection.

RPA-Based Pin Shape Selection: The SMT-generated cell layouts can have multiple distinct pin shapes for a single pin. Access to such pins must be carefully handled to avoid instability of timing and power models of the standard cells. Fig. 6(a) shows the initial SMT-generated layout of an OAI21_X1 cell. In the figure, pin *ZN* has two M1 shapes, ZN_a and ZN_b. When connections are made to different pin shapes (i.e., ZN_a or ZN_b), at least one of the cases will not match the cell's characterized timing and power model. For example, when output pin *ZN* is connected through the M1 pin shape ZN_a, the cell delay is 10 ps, but when the connection is made through ZN_b, the cell delay is 8 ps. This instability with respect to the cell timing/power model is unacceptable in modern design enablements. Therefore, when a pin has multiple candidate shapes, our framework chooses one of these shapes to use in the standard-cell layout that is produced.

Given our focus on routability, we apply pin shape selection based on the RPA pin accessibility metric [34]. Fig. 6 shows an example of our pin shape selection. Accessibility of a given pin is affected by other pins within a distance d_{int}. We set d_{int} as 1.0 for the EL design rule set, and 2.0 for the ET design rule set. We then calculate *RPA* values for each pin shape. From [34], pin access points are defined as $a_{p,m}$ where p is the corresponding pin (i.e., single pin shape) and m is the position (index) in terms of M2 routing tracks. $\mathcal{A}(p)$ denotes the set of access points for a given pin p. Then, $\mathcal{N}(a_{p,m})$ denotes the set of *neighboring* pin access points that (1) do not belong to p and (2) are within distance d_{int} of p on metal layer m. For example, $\mathcal{A}(A2) = \{a_{A2,1}, a_{A2,2}\}$ and $\mathcal{N}(a_{A2,2}) = \{a_{A1,2}, a_{ZN_b,2}\}$ in Fig. 6(a). The used pin access (UPA) of pin p is defined as

$$\text{UPA}(p) = \sum_{a_i \in \mathcal{A}(p)} \sum_{a_j \in \mathcal{N}(a_i)} 1/|\mathcal{A}(p(a_j))|. \tag{1}$$

Finally, RPA of pin p is defined as

$$\text{RPA}(p) = |\mathcal{A}(p)| - \text{UPA}(p). \tag{2}$$

For example $\text{RPA}(A2) = 2.0 - (0.5 + 0.5) = 1.0$ when we apply (1) and (2) to the example of Fig. 6(a). In the same figure, the RPA values are 2.0 and 1.0 for ZN_a and ZN_b, respectively. Since the RPA value of ZN_a is larger than that of ZN_b, we choose the ZN_a pin shape for *ZN* and make ZN_b into an

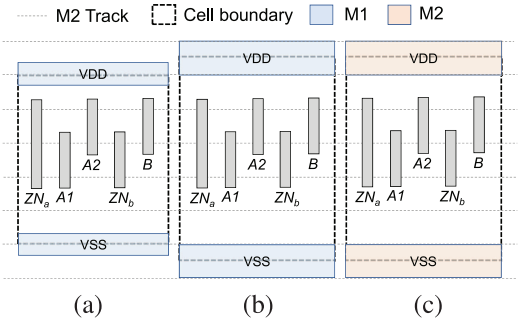

Fig. 8. Power and ground pin (PGpin) examples: (a) BPR, (b) M1, and (c) M1 + M2. The signal pin shapes (gray) are the same regardless of PGpin.

obstacle on M1, as shown in Fig. 6(b). We empirically confirm the benefits of the pin shape selection for routability, as measured by the K_{th} metric, in Section V-B.

Top-Metal-Only Pin Shape Selection: When standard cells have multiple pin shapes on multiple metal layers, we incorporate a *top-metal-only* pin shape selection step. In the example of Fig. 7, the initial layout of the OAI21_X1 cell has three pin shapes for the *ZN* pin; two on M1 (and V1) and one on M2. In this multiple-layer situation, we do not calculate RPA values for the pin shapes. Instead, we choose the M2 pin shape for *ZN*, and the two M1 pin shapes (and V1) become obstacles. This methodology brings several benefits.

1) Replacing M1 (and V1) pin shapes with obstacles benefits overall pin accessibility by reducing the complexity of pin access: P&R tools can solve the pin access problem more easily and with fewer DRCs.
2) As with the above-described pin shape selection, we avoid instability of timing and power with respect to characterized models.
3) Our top-metal-only pin shape selection mitigates susceptibility to electromigration (EM) by removing M1 pin shapes (e.g., Posser *et al.* [31] showed a cell-internal signal EM problem in accordance with output pin position).
4) Last, changing M1 (and V1) pin shapes into obstacles improves accessibility of other neighboring M1 pins.

Power and Ground Pin Generation: To make SMT-produced layouts usable by P&R tools, power and ground pins must be added. Fig. 8 shows how we define power and ground pins for standard cells. As noted in Section III-B for BPR, the width of power and ground pins is equal to the minimum M1 width. For M1 and M1 + M2, it is double this width. M2 routing grids of standard cells with BPR have an offset by half of the M2 track pitch. The figure also shows how the height of standard cells is determined by the RT and PGpin parameters.

Liberty and Technology File Generation: The remaining parts of our design enablement generate Liberty and technology files. We use dummy Liberty files to avoid any potential errors in the commercial P&R tools. Our technology file generation takes as inputs the technology parameters CPP, MP and DR, and converts design rules into corresponding file formats (e.g., LEF and routing technology files) to enable use of commercial P&R tools.

IV. Routability Assessment and Learning-Based K_{th} Prediction

A. Cell-Level Routability Assessment Based on Knight's Tour

In advanced technologies, pin accessibility of individual standard-cell masters has become a critical challenge with decreasing cell heights. We therefore propose *cell-level routability assessment* to evaluate routability at the individual cell level. The previous work of [20] uses baseline *mesh-like placements* to rank BEOL stacks by routability. These induce nearly square-mesh netlist topologies constructed using a single cell master from a fixed library. By contrast, our goal is to assess routability for a given choice of technology and design parameters, leveraging our standard-cell generation capability. The mesh-like placements used in [20] have key drawbacks: 1) they do not closely reflect the WL and Rent parameter attributes of real design testcases and 2) their construction encompasses only 2- and 3-input cells. In this work, we overcome these drawbacks by using a *knight's tour* to induce the initial placed netlist that is tangled to find K_{th}.

A *knight's tour* is a sequence of *knight's moves* that visits each square on a chessboard exactly once. A knight moves from its current square to a square that is two by one (or one by two) squares away, in any direction. In our approach, a square in an M by N chessboard corresponds to a placed cell in an initial placement of M by N instances of a given master. A knight's tour ordering induces connections—i.e., a netlist—within this placement. More precisely, for a given layout region with M rows and N columns of placed instances, and a k-input cell master, we create a knight's tour-based topology.

1) In the layout region with M rows by N columns of instances, we generate a knight's tour ordering. For this, we use the well-known Warnsdorff-rule algorithm [45].
2) An instance of the k-input cell master that is at position i in the knight's tour is assigned fanins from cells at positions $i-1$ to $i-k$ in the knight's tour ordering. For a given 2-input cell i, there will be a fanin connection from the output of cell $i-1$ to the first input of cell i, and a fanin connection from the output of cell $i-2$ to the second input of cell i.
3) We end up with an initial placement ("chessboard") having $M \times N$ instances, and an artificial netlist having connections induced by the knight's tour.

Fig. 9 shows an example of a knight's tour topology with 2-input cells. The red arrows indicate the connections between cell i and cell $i-1$. The green arrows indicate the connections between cell i and cell $i-2$.

Our studies confirm improved flexibility and correspondences to real design netlists when we apply the knight's tour-based approach. Table V shows placed WLs of the AES design, and of mesh-like placements [20] and knight's tour-induced netlists. INV_X1, NAND2_X1, NAND3_X1 and NAND4_X1 cell masters are used, with default technology and design parameters as specified in Section V-A. Particularly when based on the NAND3_X1 master, the knight's tour topologies can give more realistic WLs than mesh-like placements. (As noted above, the mesh-like placement construction does not support use of 1- or 4-input cell masters.)

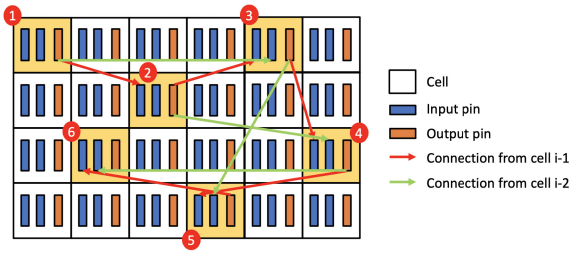

Fig. 9. Example of knight's tour-based connections in a placement of 2-input cells. Numbers in red circles indicate the knight's tour ordering.

TABLE V
PLACED DESIGN INFORMATION FOR AES, MESH-LIKE PLACEMENTS AND KNIGHT'S TOUR-BASED TOPOLOGIES. MESH-LIKE PLACEMENTS AND KNIGHT'S TOURS HAVE 115 × 115 PLACED INSTANCES. AREA DENOTES SUM OF INSTANCE AREA IN THE DESIGNS

Design	Cell	Area (μM^2)	#Insts	WL (μM)
AES	-	1088	13221	46880
Mesh	INV_X1	Not supported		
	NAND2_X1	799	13225	11001
	NAND3_X1	999	13225	13932
	NAND4_X1	Not supported		
Knight's Tour	INV_X1	599	13225	10264
	NAND2_X1	799	13225	23610
	NAND3_X1	999	13225	41103
	NAND4_X1	1199	13225	62251

TABLE VI
RENT PARAMETERS FOR MESH-LIKE AND KNIGHT'S TOUR-BASED PLACED NETLISTS. WE SHOW RANGES OF RENT PARAMETERS OBTAINED FROM PLACEMENTS BY USING [52]. WE USE *Rectangle Sampling-Based Methods I and II* FOR THE EVALUATION METHOD, *Types I and II* FOR THE PIN COUNTING METHOD, AND *Geometric Mean* FOR THE AVERAGING METHOD IN [52]

Source	Design	Rent Parameter
OpenCores [51]	AES	[0.726, 0.832]
	JPEG	[0.617, 0.813]
	VGA	[0.656, 0.879]
	LDPC	[0.812, 0.891]
PROBE [20]	**Mesh (NAND2_X1)**	**[0.530, 0.535]**
	Mesh (NAND3_X1)	**[0.500, 0.585]**
PROBE2.0	**Knight's tour (NAND2_X1)**	**[0.676, 0.761]**
	Knight's tour (NAND3_X1)	**[0.708, 0.820]**

We also note that mesh-like placements have a simple topology wherein connections are only between neighbor cells in the initial placement. This inflexibility can weaken correspondences to real-world designs. Notably, the Rent parameter p of the (2-dimensional) mesh topology is 0.5 [16], [24] according to Rent's rule. We have studied achievable Rent parameters for mesh-like placements and knight's tour-based topologies by using RentCon [52]. Table VI shows that Rent parameters of real circuits (AES, LDPC, JPEG, VGA) range from 0.617 to 0.891. On the other hand, the mesh-like placements p values from 0.500 to 0.585, while our proposed knight's tour-based placements have p values from 0.676 to 0.820. Again, the knight's tour construction can more closely reflect real circuits. Based on these studies, we use the knight's tour-based topology construction in our cell-level routability assessment.

B. Design-Level Routability Assessment

To perform design-level routability assessments, we begin with open-source RTL designs from OpenCores [51] and use commercial logic synthesis to obtain gate-level netlists. Following [20], we evaluate design-level routability based on *cell width-regularized* placements [Fig. 3(b)] of these netlists. For each standard-cell library, we modify LEF such that all combinational cells are bloated to have the same width as the maximum-width cell among all 38 combinational cells in the library. (Our designs also instantiate D flip-flops, which have the largest cell width in our generated standard-cell libraries. We fix the locations of flip-flops in designs after placement and do not swap them.)

Cell width regularization is proposed in the work of [20]. Its purpose is to avoid illegal placements (with cell overlap) when neighbor cells are swapped during tangling. Without the cell width regularization, the neighbor-cell swap operations might cause cell overlap due to different cell widths. Of course, the standard P&R flow does not use width-regularized cells, which raises the question of whether use of bloated cells could lead to misleading design-level routability assessments. Our experimental study of K_{th} obtained with bloated cells, versus maximum achievable utilization obtained with unbloated cells, is reported below in Section V-E and offers some reassurance in this regard. Aside from this, we use bloated cells for two main reasons. 1) our cell-level routability assessment focuses on intrinsic routability of cell layouts without any bloating of cell widths and 2) despite the extra widths of cells, we can still assess pin accessibility for cells. Bloated cells still retain their original order and shapes of pins. So, the intrinsic routability for standard cells can still be measured. Below, design-level routability assessments with various technology and design parameters are reported in Sections V-D and V-E.

C. Learning-Based K_{th} Prediction

Although K_{th} provides a useful means of routability assessment, it has three potential logistic disadvantages: large runtimes, large data footprints, and high consumption of commercial EDA licenses. First, as we perform neighbor-swaps to increase routing difficulty for a given placement, the increasing number of DRCs leads to large runtimes. This is because P&R tools perform increasingly expensive search-and-repair iterations in the detailed router. Furthermore, although we set a maximum runtime per each P&R run,[1] our overall runtime burden is still large, since the total number of tools runs itself is large. For example, when we perform P&R with $K = 1$ to 30 to obtain K_{th}, and there are just 100 sets of standard-cell libraries to be evaluated, up to 3000 P&R runs are required for this assessment. Second, large amounts of disk space may be used by our framework. For example, just 3000 runs of P&R with the VGA design occupy around 300 GB of disk space in our experiments. Third, in many contexts the number of available commercial P&R tool licenses will also be limited.

To mitigate these potential disadvantages, we have developed *learning-based* K_{th} prediction. That is, to reduce the number of tool runs (thus saving runtime, disk space and license usage), we predict K_{th} via ML techniques, without performing all P&R runs. Fig. 2(b) and (c) shows PROBE2.0

[1]We assume that runs with #DRCs under/around the threshold are finished within the maximum runtime. We set maximum runtimes based on the instance counts of the designs. In this work, we use maximum runtimes of two hours for a knight's tour and AES, and three hours for LDPC, JPEG, and VGA.

TABLE VII
INPUT FEATURES OF THE K_{th} PREDICTION

Type	Name	Data	Notes
Technology	CPP	Integer	
	MP	Integer	
	PGpin	Category	BPR, M1, M1+M2
	CH	Integer	
	MPO	Integer	
	DR	Category	EL, ET
Design	BEOL	Integer	
	PDN	Category	Backside, Sparse, Middle, Dense
	Tool	Category	Tool A, Tool B
	Util	Float	[0.0, 1.0]
	Inst	Integer	
	Net	Integer	

flows without, and with, K_{th} prediction. The flow without K_{th} prediction in Fig. 2(b) requires generation steps and multiple P&R runs. On the other hand, the flow with prediction in Fig. 2(c) finds K_{th} using only inference with a trained ML model. We use 12 input features in our ML modeling, based on the technology and design parameters in Sections III-B and III-C, respectively. Table VII summarizes the types, names, and data types of the input features that we use in the K_{th} prediction. From the original set of technology parameters, we omit Fin and RT from our feature list since Fin and RT are derivable from PGpin and CH. From the original set of design parameters given in Section III-C, we omit "Design" and add the number of instances (*Inst*) and the number of nets (*Net*) as design features in our model.

We present our experimental results for the ML-based K_{th} prediction in Section V-F. Importantly, our ML models are trained in less than an hour using approximately 1968 data points (80% out of a total of 2460 data points); inference then requires only seconds for any technology-design parameter combination. By contrast, collecting a single K_{th} value for a given technology-design parameter combination requires not only the generation of standard-cell layouts and design enablements, but also up to 30 P&R runs that each consume significant average and maximum tool runtimes of 0.4 and 3.0 h, respectively. Thus, runtime overhead is 12 (resp., 90) hours on average (resp., at most) per K_{th} value. In light of such large runtimes, the primary goal of our ML-based K_{th} prediction is to reduce the number of required P&R runs by predicting unseen data (that is, K_{th} values) with reasonably small errors. In Section V-F below, we also perform an experiment to show the tradeoff of model accuracy versus training data (TD) overheads.

V. EXPERIMENTAL SETUP AND RESULT

We demonstrate routability assessment capability of PROBE2.0 via five main experiments.

1) *Expt. 1:* Cell-level assessment with a knight's tour.
2) *Expt. 2:* Design-level assessment with technology parameters.
3) *Expt. 3:* Design-level assessment with design parameters.
4) *Expt. 4:* Achievable utilization study with K_{th} results.
5) *Expt. 5:* Learning-based K_{th} prediction.

TABLE VIII
LIST OF 39 STANDARD CELLS PER GENERATED LIBRARY

Cell List	Size
Inverter (INV), Buffer (BUF)	X1, X2, X4, X8
2-input AND/OR/NAND/NOR (AND2/OR2/NAND2/NOR2)	X1, X2
3-input AND/OR/NAND/NOR (AND3/OR3/NAND3/NOR3)	X1, X2
4-input NAND/NOR (NAND4/NOR4)	X1, X2
2-1 AND-OR-Inverter (AOI21), 2-2 AND-OR-Inverter (AOI22)	X1, X2
2-1 OR-AND-Inverter (OAI21), 2-2 OR-AND-Inverter (OAI22)	X1, X2
2-input MUX/XOR (MUX2/XOR2), D flip-flop (DFF)	X1

A. Experimental Setup

We generate 39 cells (38 combinational and 1 sequential) for each standard-cell library that we study, as listed in Table VIII. We name cells according to functionality, number of inputs, and size. For example, INV_X1 is an X1-sized inverter, and AND3_X2 is an X2-sized 3-input AND gate. Our SMT-based standard-cell layout generation tool is open-sourced in [53].

Our experiments are based on a commercial 14-nm technology. We modify design rules of Mx (M1, M2 and M3) layers (and of vias above) in routing technology files to enable scaling to sub-7-nm technologies. We draw FEOL and BEOL (Mx) layer parameters from [10], [39] (see Section III-A). Layer parameters and rules for M4 and above are drawn from the 14-nm technology. For example, we choose the 9M (9-layer) BEOL stack option from the 14-nm technology, and modify rules for the Mx layers (M1, M2, M3). We use existing design rules for the upper layers (M4 to M9). Mx layers are constrained to use unidirectional and min-width routing for standard-cell layout generation and P&R. We also assume that M1 is unavailable for signal routing.

All design rules are described in LEF format and routing technology file format [55] for use by commercial P&R tools. SMT-generated standard-cell layouts are converted into LEF format. For cell-level routability assessments, we use a knight's tour with 50 by 50 instances, constructed using an open-sourced implementation of the Warnsdorff-rule algorithm [45]. For design-level assessments, we perform synthesis by using a commercial tool [54] and P&R by using two commercial P&R tools [46], [55].

We also use GNU parallel [47] to perform multiple evaluations in parallel. We assume that our designs with #DRCs less than the threshold finish within three h. Setting the maximum runtime in GNU parallel reduces the total runtime of the framework. We use 500 DRCs as the DRC threshold to determine K_{th}. (The work of PROBE [20] used a threshold of 150 DRCs. We set the higher threshold of 500 DRCs since SMT-generated standard-cell layouts might not be as well-optimized for pin accessibility as mature libraries from industry.) Finally, we set default parameters as shown in Table IX. In all of our experiments, when not otherwise specified, we use these predefined default parameters.

B. Expt. 1 (Cell-Level Assessment With Knight's Tour)

In this experiment, we perform knight's tour-based cell-level routability assessments. To show K_{th} of standard cells with various cell heights, we draw cells from 6T, 7T and 8T libraries. Among the library cells, we choose 15 combinational cells that have X1 size and more than one input pins.

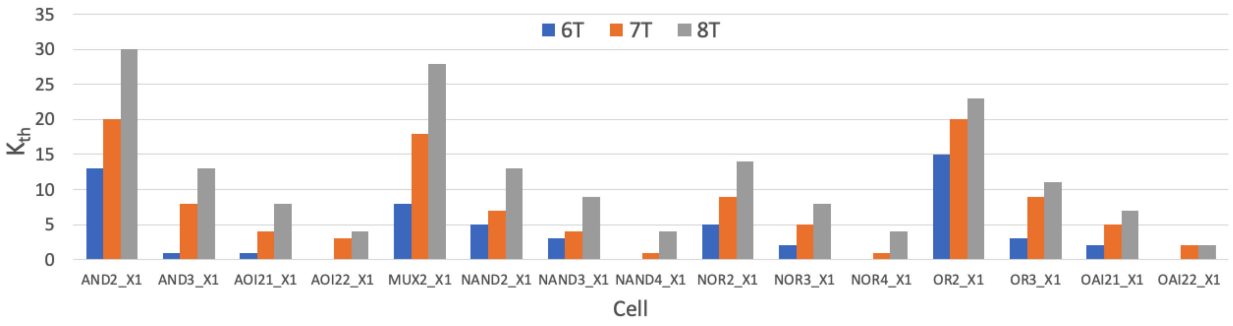

Fig. 10. K_{th} results from knight's tour-based cell-level routability assessments across various track heights and cell masters. Fifteen cell masters from 6T, 7T, and 8T standard-cell libraries are used in this experiment.

TABLE IX
DEFAULT PARAMETERS IN OUR EXPERIMENTS

Design Parameter		Technology Parameter	
Name	Value	Name	Value
Fin	3	BEOL	10M
CPP	54	PDN	Middle
MP	40	Tool	A
RT	5	Util	0.7
PGpin	M1	Design	AES
CH	7	-	-
MPO	2	-	-
DR	EL	-	-

Fig. 10 shows K_{th} results from the cell-level routability assessment. From the results, we observe larger K_{th} as cell height increases. Also, by examining K_{th} per cell, we can see which cell types are routing-critical. For example, 4-input cells have relatively small values of K_{th} according to this experiment. Thus, at an early stage of technology development, cell-specific K_{th} is a promising indicator to determine which cells need to be improved in terms of routability.

We also show the benefit of our RPA-based pin selection. We generate standard-cell layouts using two pin shape selection methods, *RPA-Best* and *RPA-Worst*. RPA-Best denotes our proposed RPA-based pin shape selection which chooses the pin shape with the largest RPA value. To show that RPA-Best brings meaningful benefits, we also create standard-cell libraries by choosing a pin shape per pin that has *smallest* RPA value; this yields the RPA-Worst method. We study six cells from 6T libraries and two cells from 7T libraries. We perform cell-level routability assessments on RPA-Best and RPA-Worst versions of these standard cells.

Fig. 11 plots the number of DRCs versus K, for the eight standard cells with different pin shape selection methods. Also, Table X shows the K_{th} values for each standard cell. Seven standard cells (out of eight cells) with RPA-Best show larger K_{th} than those with RPA-Worst. One cell (OAI_X2) shows the same K_{th} for both methods. We see that our RPA-based pin shape selection helps to improve the routability.

C. Expt. 2 (Design-Level Assessment With Technology Parameters)

In this experiment, we perform design-level routability experiments with various technology parameters. The goal is to show the routability impact of technology and standard-cell

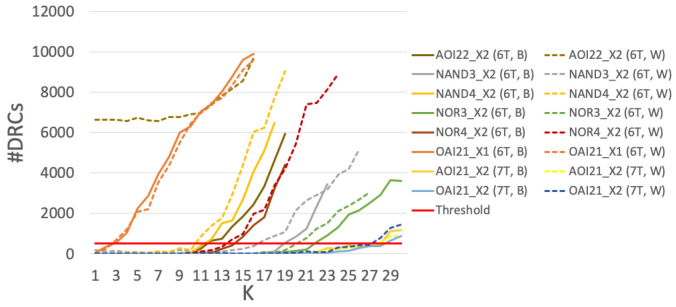

Fig. 11. Cell-level routability assessments showing #DRCs versus K with different pin shape selection methods. Six 6T and two 7T standard cells with RPA-Best (B) and RPA-Worst (W) pin shape selection methods are shown. Solid lines are for cells with RPA-Best, and dashed lines are for cells with RPA-Worst. Table X shows the corresponding K_{th} values for each case.

TABLE X
K_{th} RESULTS WITH RPA-BEST AND RPA-WORST PIN SHAPE SELECTION METHODS. THIS TABLE SHOWS THE K_{th} VALUES CORRESPONDING TO THE DATA SHOWN IN FIG. 11

CH	Cell	K_{th}	
		RPA-Best	RPA-Worst
6T	AOI22_X2	11	0
	NAND3_X2	18	16
	NAND4_X2	11	10
	NOR3_X2	21	20
	NOR4_X2	14	13
	OAI21_X1	2	2
7T	AOI21_X2	28	27
	OAI21_X2	28	27

architecture options. Fig. 12 shows K_{th} results that highlight the individual impacts of five technology parameters. We also experimentally confirm the ability of our framework to assess various design rule options. To study design rule choices, we define two scenarios, *DR1* and *DR2*. DR1 is a scenario which uses CPP = 54 and the EL design rule set. DR2 uses CPP = 48 and the ET design rule set. We choose these two design rule scenarios so that they are more likely to induce different standard-cell architectures.

We make five observations from Fig. 12, as follows.

1) As cell heights (or routing tracks) increase, the corresponding K_{th} values also increase.
2) Standard cells with M1 PGpin (and the same RT) have better routability than those with BPR or M1+M2 PGpin. Also, standard cells with BPR and M1+M2 show similar K_{th} values. This reflects two phenomena. First, cells with

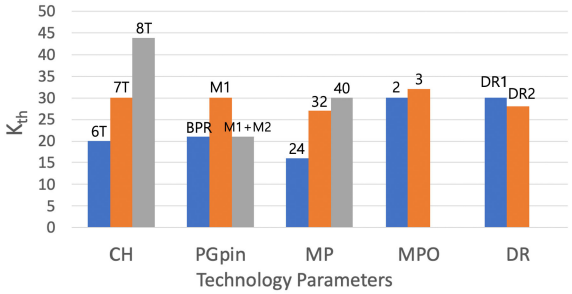

Fig. 12. Design-level assessments with various technology parameters.

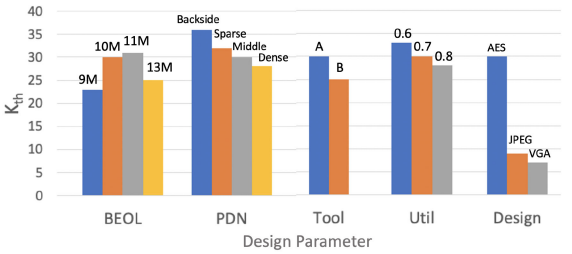

Fig. 13. K_{th} results of design-level routability assessments with various design parameters.

BPR have one track less cell height than corresponding non-BPR cells. Second, cells with M1+M2 have worse routability than those with M1. This is because the M2 pins in standard cells with M1+M2 will block routing resources over the cells.

3) Smaller pitches of Mx layers bring lower K_{th} values, despite smaller pitches nominally providing more routing resources on Mx layers.
4) Minimum pin openings of our standard-cell layouts have a clear impact on routability. Standard cells with 3 MPO show better routability than those with 2 MPO.
5) For the DR cases as shown in Fig. 12, DR2 has smaller K_{th} than DR1, implying that the tight design rule has a harmful impact on routability.

With respect to the fifth observation, we note that while DR2 brings smaller K_{th}, standard-cell layouts with 48-nm CPP (DR2) have around 11% less area than layouts with 54-nm CPP (DR1). We explore the relationship between area (utilization) and K_{th} in Section V-E below. Finally, Table XI gives additional details of K_{th} results for all the combinations with various Fin, MP, RT, PGpin and MPO in our experiments. (Data shown in Fig. 12 is a subset of the table data.)

D. Expt. 3 (Design-Level Assessment With Design Parameters)

In this experiment, we study routability impacts of design parameter options. Fig. 13 shows the K_{th} results from this study, which spans BEOL, PDN, Tool, Util, and Design parameters. We make the following observations.

1) According to the K_{th} values with the various BEOL stack options, 10M has better routability than 9M, and 11M has better routability than 10M. Also, 10M and 11M have better routability than 13M. As shown in Table II, the routing resources of 10M and 11M are larger than those of 13M. This illustrates how overall routing capacity can worsen even if the number of metal layers is increased. However, the additional layers bring other benefits. For example, we may implement a denser, more robust PDN resulting in better design performance due to less resistivity of wide metal layers.
2) As fewer layer resources are used for PDN, the results show better routability.
3) The K_{th} results from Tool A dominate those of Tool B. In this sense, our framework also has the ability to evaluate placers and routers.
4) As Util increases, the K_{th} values decrease since routing with denser placements is more difficult.
5) The base designs used in the design-level routability assessments also affect the results.

In particular, designs with larger instance counts have smaller K_{th}. This is expected: for a given amount of tangling (K), and all else being equal, a larger design's placement is expected to have more routing #DRCs. ([20] gives a probabilistic analysis of routing hotspots and routing failure according to the number of instances.)

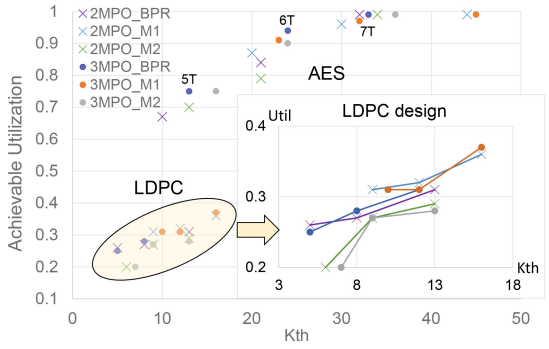

Fig. 14. Comparison between K_{th} and *achievable utilization* for the AES and LDPC designs. The plot shows three instances of each icon. Left to right, these correspond, respectively, to 5T, 6T, and 7T cells for BPR, and to 6T, 7T, and 8T cells for M1 and M2. (To help clarify this, we give 5T, 6T, and 7T labels for *3MPO_BPR* cell libraries.) For AES, we use the same settings as in Table IX. For LDPC, we set Util = 0.2 and enable M1 routing when extracting K_{th} and achievable utilization. (LDPC has much lower achievable utilization than AES due to its complex routing topology.)

TABLE XI
K_{th} RESULTS WITH VARIOUS FIN, MP, RT, PGPIN, AND MPO

Fin	RT	MP	MPO	K_{th}		
				BPR	M1	M1+M2
2	4	24	2	2	11	4
			3	8	11	6
		32	2	10	15	9
			3	11	16	10
		40	2	10	20	13
			3	13	23	16
3	5	24	2	4	16	10
			3	6	19	11
		32	2	17	27	17
			3	19	28	18
		40	2	21	30	21
			3	24	32	24
3	6	24	2	17	28	19
			3	19	27	19
		32	2	27	44	25
			3	28	45	27
		40	2	32	44	34
			3	33	45	36

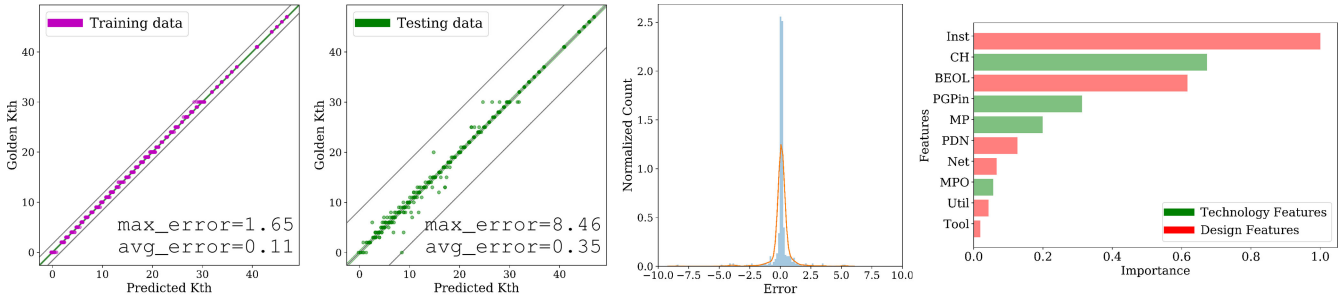

Fig. 15. Comparison between golden K_{th} and predicted K_{th} from the top-1 model in Table XII, i.e., the *StackedEnsemble* model, and extracted feature importance. (a) TD. (b) Testing data. (c) Error distribution on testing data with kernel density estimation (KDE) plot. (d) Extracted feature importance, reported from the *XGBoost* model in Table XII. The green bars denote technology features and the red bars denote design features.

E. Expt. 4 (Achievable Utilization Versus K_{th})

In production IC design, maximizing the utilization of die area is a common objective toward achieving the best PPAC. Also, the maximum achievable utilization is a first-order indicator for routability, even as it captures the A, C dimensions of PPAC. In this experiment, we explore the relationship between achievable utilization (in design implementations using unbloated cells) and K_{th} (in design-level routability assessment using cell width-regularized libraries). Such a study could inform the future use of K_{th} to predict die utilization and area. We define *achievable utilization* as the highest utilization for which #DRCs is smaller than the predefined DRC threshold.

In our study of achievable utilization, we use two designs, AES and LDPC. For AES, we collect achievable utilizations from 0.60 to 0.99, with step size of 0.01. For LDPC, which is a notoriously difficult to route testcase, we collect achievable utilizations from 0.20 to 0.40, again with step size of 0.01. To extract K_{th}, we set Util as 0.7 for AES and 0.2 for LDPC. Fig. 14 shows our K_{th} results and the corresponding achievable utilizations. The plot shows that K_{th} values and achievable utilizations are sensibly related for the given design parameters. The data also show how proper combination of technology and design parameter choices can potentially bring routing-friendliness and high utilization: large cell heights, large cell areas, sparse PDN, ample routing resources, and/or relaxed design rules all lead to routability with maximum utilization. Especially, in the enlarged view of LDPC design plot, we observe similar tendencies as seen in Fig. 12. For example, the cells with M1 PGpin have better K_{th} compared to cells with M2 and BPR. Also, both K_{th} values and achievable utilizations gradually increase with cell heights. The cells with 2MPO and 3MPO have similar K_{th} values and achievable utilizations. This experiment hints that K_{th} could be useful in a predictor of design-specific achievable utilization. We leave this possibility for future work.

F. Expt. 5 (Learning-Based K_{th} Prediction)

We now show results from our learning-based K_{th} prediction. We compile a dataset of 2460 K_{th} values corresponding to a wide range of settings for the various features described in Table IV. We use the open-source AutoML package [48] to predict K_{th}. AutoML has a hyperparameter tuning ability on various models including gradient boosting machine (GBM) [12], XGBoost [5], distributed random forest (DRF), extremely randomized tree (XRT) [13], deep learning (DL) and generalized linear model (GLM) [28]. AutoML also suggests combined models that outperform a single model.

TABLE XII
RESULTS ON TD OF TOP-5 ML MODELS BY USING AUTOML. THE THREE *GBM* MODELS (*GBM_1*, *GBM_2*, AND *GBM_3*) HAVE DIFFERENT CONFIGURATIONS FOR HYPERPARAMETERS

Model	Deviance	RMSE	MAE
StackedEnsemble_BestOfFamily	0.733	0.857	0.353
XGBoost	0.865	0.930	0.300
GBM_1	0.896	0.947	0.374
GBM_2	0.947	0.973	0.350
GBM_3	0.950	0.975	0.440

We down-sample and up-sample our data since the data are imbalanced in terms of K_{th} distributions.[2] We use 60 as a target sample number to generate balanced data. We randomly partition our augmented (i.e., after sampling schemes are applied) dataset as 80% used for training and 20% used for testing. We perform all experiments using an Intel Xeon Gold 6148 2.40GHz server (80 threads) with 256-GB RAM.

We use the 3.30.0.6 version of AutoML to train our models, with default input parameter settings for AutoML. The default settings of AutoML include *nfolds* = 5 for cross-validation, *leaderboard_frame* = testing set, and *sort_metric* = mean residual deviance to rank the trained regression models. Details of these settings are found at [48]. We set the training time limit as one hour with 80 threads. The suggested models from AutoML and the corresponding trained results are described in Table XII. Note that *Deviance* denotes mean residual deviance, *RMSE* denotes root mean squared error, and *MAE* denotes mean absolute error.

Fig. 15 shows comparisons of *golden* K_{th} (i.e., ground truth data which comes from actual results obtained from our framework) versus *predicted* K_{th} using the model *StackedEnsemble_BestOfFamily*. The model contains six models: 1) GBM; 2) XGBoost; 3) DRF; 4) XRT; 5) DL; and

[2]When generated standard cells are not routing-friendly, routing for designs that use those cells might be infeasible. This causes the observed distribution of K_{th} values to be biased toward zero. The down-sampling and up-sampling schemes help us to avoid over-fitting in the prediction of K_{th}.

TABLE XIII
RESULTS OF DL MODELING WITH AUTOML. *DL_1h* (RESP., *DL_24h*) DENOTES A DL MODEL WITH A 1-H (RESP., 24-H) RUNTIME LIMIT FOR TRAINING

Model	Deviance	RMSE	MAE
DL_1h	6.275	2.505	1.789
DL_24h	1.469	1.212	0.615

6) GLM. AutoML generates a stacked ensemble model with GLM as *meta_learner*, based on those six models. We show *max_error* and *avg_error* metrics, which are the maximum and average difference between golden and predicted K_{th}, respectively. Fig. 15(c) shows a balanced error distribution for the testing set. To further understand our K_{th} prediction, we extract feature importance from the XGBoost model. We do this for XGBoost since *StackedEnsemble_BestOfFamily* consists of six models and does not support such a feature extraction. The extracted feature importance is visualized in Fig. 15(d). We conclude that K_{th} is mostly affected by Inst, CH and BEOL.

From the results in Table XII, we can infer that boosting ML models (XGBoost and GBM) outperform DL models in our experiment. (Put another way: no DL model made it into the AutoML Top-5 models.) However, given the recent intense interest in DL, we also explicitly study prediction with DL models. Table XIII shows the DL model results with AutoML, for the same training set as in Table XII. In this experiment, we train the DL models with two maximum runtimes for training. DL_1h denotes a DL model obtained with the same 1-h training runtime limit used for Table XII. DL_24h denotes a DL model with 24-h training runtime limit. Even with the longer training time, DL does not surpass the other ML models in Table XII. (The recent study [11] provides insights on boosting models outperforming DL models.)

As noted in Section IV-C, a primary benefit of learning-based K_{th} prediction [Fig. 2(c)] is that it can save up to 90 h of P&R tool runtime with each predicted value. To characterize the tradeoff of model accuracy versus TD overheads, we further conduct an experiment with different numbers of TD. Since obtaining a single K_{th} requires 30 P&R runs (averaging 0.4 h each) in our experiment, obtaining a total of 2460 K_{th} values requires approximately 15.4 days of runtime. We have studied K_{th} prediction with various numbers of TD, and with fixed testing data. Table XIV shows results with different amounts of TD ($\sim$20%, $\sim$40%, $\sim$60%, and $\sim$80% of the total dataset size). Generating 20% of the total data as TD reduces schedule overhead to approximately 3.1 days, and leads to 1.155 average K_{th} error in predicting the held-out 20% test data. As expected, more TD leads to better model accuracy (smaller errors) at the cost of schedule overhead.

TABLE XIV
MODEL ACCURACY RESULTS WITH DIFFERENT AMOUNTS OF TD, I.E., 20, 40, 60, AND 80% OF THE TOTAL DATA (2460 K_{th} VALUES). TESTING DATA (20%) ARE FIXED FOR AN APPLES-TO-APPLES COMPARISON. *ME* AND *AE*, RESPECTIVELY, DENOTE MAXIMUM AND AVERAGE ERROR OF K_{th} PREDICTION ON TESTING DATA

TD	ME	AE	Deviance	RMSE	MAE
20%	14.696	1.155	4.916	2.217	1.155
40%	8.131	0.610	1.702	1.304	0.610
60%	8.553	0.392	1.006	1.003	0.392
80%	8.459	0.353	0.733	0.857	0.353

VI. CONCLUSION

We have proposed a novel framework to evaluate routability impacts of advanced-node scaling options, spanning across technology, design enablement, and design parameters. Several options for *scaling boosters* are assessed in terms of their routability impacts. Our framework is well matched to the needs of DTCO, particularly for early stages of technology development. Crucially, our framework can reduce the timelines for design enablement and design implementation that limit today's DTCO methodologies. Also, our framework can flexibly support additional technology and design options.

We integrate a powerful SMT-based standard-cell layout generation capability. Optimal layout solutions are obtained via a unified constraint satisfaction formulation that spans technology and cell architecture parameters. We also build upon the previous routability assessment framework of [20].

Our new framework provides automatic generation of cell libraries and collaterals (LEFs, Liberty, routing technology files) for commercial P&R tooling. We propose RPA-based and top-metal-only pin shape selection to improve routability of our generated standard-cell libraries. We validate these aspects of our methodology using a novel *cell-based* routability assessment with knight's tour-based topology generation. We also perform *design-based* routability assessment using open-source testcases, and show correlations of the K_{th} routability metric to achievable utilizations in P&R. Furthermore, we propose learning-based K_{th} prediction to reduce runtimes and required disk space, and to mitigate P&R tool license overheads. Finally, experimental studies confirm the capability of our framework to produce routability assessments across a large range of technology and design parameters.

Open directions for future research include the following.

1) Our framework focuses on the Area and Cost dimensions of PPAC. This has value in early technology development, especially since density is the dominant driver for foundry technology and design enablement [40]. However, future work should broaden assessments to include power and performance. Automation and/or prediction of library characterizations is a related challenge for future research.
2) Our framework today covers a number of technology options, design rules and standard-cell architectures. However, extensions to support other technology options and/or design rules for advanced technologies may be desirable. For example, a target technology might require self-aligned double patterning (SADP) design rules and bidirectional routing. Also, new standard-cell architectures might be required, such as multiheight standard cells and rectilinear pin shapes.
3) Extending our framework to incorporate an open-source P&R tool, such as [1], [56], may help to scale exploration and turnaround times beyond the limits of available commercial P&R tool licenses. In this context, learning to map routability and other PPAC-related assessments between implementation tool chains will be a valuable future contribution.

ACKNOWLEDGMENT

The authors thank Dr. S.C. Song for providing valuable feedback.

REFERENCES

[1] T. Ajayi *et al.*, "Toward an open-source digital flow: First learnings from the OpenROAD project," in *Proc. DAC*, 2019, pp. 1–4.

[2] 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*, 2016, pp. 41–48.

[3] B. Chava *et al.*, "DTCO exploration for efficient standard cell power rails," in *Proc. SPIE Design Process Technol. Cooptim. Manuf. XII*, vol. 10588, 2018, pp. 1–6.

[4] B. Chava *et al.*, "Backside power delivery as a scaling knob for future systems," in *Proc. SPIE Design Process Technol. Cooptim. Manuf. XIII*, vol. 10962, 2019, pp. 1–6.

[5] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system,," in *Proc. ACM SIGKDD Int. Conf. Knowl. Disc. Data Min.*, 2016, pp. 785–794.

[6] L. Chen, C. Huang, Y. Chang, and H. Chen, "A learning-based methodology for routability prediction in placement," in *Proc. VLSI-DAT*, 2018, pp. 1–4.

[7] C.-K. Cheng, A. B. Kahng, I. Kang, and L. Wang, "RePlAce: Advancing solution quality and routability validation in global placement," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst*, vol. 38, no. 9, pp. 1717–1730, Sep. 2019.

[8] C.-K. Cheng, D. Lee, and D. Park, "Standard-cell scaling framework with guaranteed pin-accessibility," in *Proc. ISCAS*, 2020, pp. 1–5.

[9] P. Van Cleeff, S. Hougardy, J. Silvanus, and T. Werner, "BonnCell: Automatic cell layout in the 7nm era," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 39, no. 10, pp. 2872–2885, Oct. 2020.

[10] P. Cremer, S. Hougardy, J. Schneider, and J. Silvanus, "Automatic cell layout in the 7nm era," in *Proc. ISPD*, 2017, pp. 99–106.

[11] M. Fernández-Delgado, M. S. Sirsat, E. Cernadas, S. Alawadi, S. Barro, and M. Febrero-Bande, "An extensive experimental survey of regression methods," *Neural Netw.*, vol. 111, pp. 11–34, Mar. 2019.

[12] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," *Ann. Stat.*, vol. 29, no. 5, pp. 1189–1232, 2001.

[13] P. Geurts, D. Ernst, and L. Wehenkel, "Extremely randomized trees," *Mach. Learn.*, vol. 63, no. 1, pp. 3–42, 2006.

[14] A. Gupta, J. Bommels, Y. Saad, I. Ciofi, and C. J. Wilson, "Integration scheme and 3D RC extractions of three-level supervia at 16 nm half-pitch," *Microelectron. Eng.*, vol. 191, pp. 20–24, May 2018.

[15] M. Guruswamy *et al.*, "Cellerity: A fully automatic layout synthesis system for standard cell libraries," in *Proc. DAC*, 1997, pp. 327–332.

[16] L. Hagen, A. B. Kahng, F. Kurdahi, and C. Ramachandran, "On the intrinsic rent parameter and new spectra-based methods for wireability estimation," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 13, no. 1, pp. 27–37, Jan. 1994.

[17] K. Han, A. B. Kahng, and H. Lee, "Evaluation of BEOL design rule impacts using an optimal ILP-based detailed router," in *Proc. DAC*, 2015, pp. 1–6.

[18] X. He, Y. Wang, Y. Guo, and E. F. Y. Young, "Ripple 2.0: Improved movement of cells in routability-driven placement," *ACM Trans. Design Autom. Electron. Syst.* vol. 22, no. 1, pp. 1–26, 2016.

[19] K. Jo, S. Ahn, J. Do, T. Song, T. Kim, and K. Choi, "Design rule evaluation framework using automatic cell layout generator for design technology co-optimization," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 27, no. 8, pp. 1933–1946, Aug. 2019.

[20] A. Kahng, A. B. Kahng, H. Lee, and J. Li, "PROBE: Placement, routing, back-end-of-line measurement utility," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 37, no. 7, pp. 1459–1472, Jul. 2018.

[21] I. Kang, D. Park, C. Han, and C.-K. Cheng, "Fast and precise routability analysis with conditional design rules," in *Proc. SLIP*, 2018, pp. 1–4.

[22] M.-C. Kim, J. Hu, D. J. Lee, and I. L. Markov, "A SimPLR method for routability-driven placement," in *Proc. ICCAD*, 2011, pp. 67–73.

[23] G. Ke *et al.*, "LightGBM: A highly efficient gradient boosting decision tree," in *Proc. NIPS*, 2017, pp. 3146–3154.

[24] B. S. Landman and R. L. Russo, "On a pin versus block relationship for partition of logic graphs," *IEEE Trans. Comput.*, vol. C-20, no. 12, pp. 1469–1479, Dec. 1971.

[25] D. Lee *et al.*, "SP&R: SMT-based simultaneous place—Route for standard cell synthesis of advanced nodes," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, early access, Nov. 13, 2020, doi: 10.1109/TCAD.2020.3037885.

[26] Y.-L. Li *et al.*, "NCTUcell: A DDA-aware cell library generator for FinFET structure with implicitly adjustable grid map," in *Proc. DAC*, 2019, pp. 1–6.

[27] T. Lin and C. Chu, "POLAR 2.0: An effective routability-driven placer," in *Proc. DAC*, 2014, pp. 1–6.

[28] J. A. Nelder and R. W. M. Wedderburn, "Generalized linear models," *J. Roy. Stat. Soc. A Stat. Soc.*, vol. 135, no. 3, pp. 370–384, 1972.

[29] D. Park, I. Kang, Y. Kim, S. Gao, B. Lin, and C.-K. Cheng, "ROAD: Routability analysis and diagnosis framework based on SAT techniques," in *Proc. ISPD*, 2019, pp. 65–72.

[30] D. Park *et al.*, "Grid-based framework for routability analysis and diagnosis with conditional design rules," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.* vol. 39, no. 12, pp. 5097–5110, Dec. 2020.

[31] G. Posser, V. Mishra, P. Jain, R. Reis, and S. S. Sapatnekar, "A systematic approach for analyzing and optimizing cell-internal signal electromigration," in *Proc. ICCAD*, 2014, pp. 486–491.

[32] N. Ryzhenko, S. Burns, A. Sorokin, and M. Talalay, "Pin access-driven design rule clean and DFM optimized routing of standard cells under Boolean constraints," in *Proc. ISPD*, 2019, pp. 41–47.

[33] R. Sengupta, J. G. Hong, and M. Rodder "Semiconductor device having buried power rail," U.S. Patent 9 570 395, 2017.

[34] J. Seo, J. Jung, S. Kim, and Y. Shin, "Pin accessibility-driven cell layout redesign and placement optimization," in *Proc. DAC*, 2017, pp. 1–6.

[35] S. M. Y. Sherazi *et al.*, "Track height reduction for standard-cell in below 5nm node: How low can you go?" in *Proc. SPIE Design Process Technol. Cooptim. Manuf. XII*, vol. 10588, 2018, pp. 1–13.

[36] S. M. Y. Sherazi *et al.*, "Standard-cell design architecture options below 5nm node: The ultimate scaling of FinFET and nanosheet," in *Proc. SPIE Design Process Technol. Cooptim. Manuf. XIII*, vol. 10962, 2019, pp. 1–15.

[37] S. Ma *et al.*, "Low track height standard cell design in IN7 using scaling boosters," in *Proc. SPIE*, vol. 10148, 2017, pp. 1–8.

[38] I. Tseng, Z. C. Lee, V. Tripathi, C. M. T. Yip, Z. Chen, and J. Ong, "A system for standard cell routability checking and placement routability improvements," in *Proc. APCCAS*, 2019, pp. 125–128.

[39] V. Vashishtha, M. Vangala, and L. T. Clark, "ASAP7 predictive design kit development and cell design technology co-optimization," in *Proc. ICCAD*, 2017, pp. 992–998.

[40] H.-S. P. Wong, "Semiconductor technology: A system perspective," in *Proc. ACM/IEEE Design Autom. Conf.*, 2020, pp. 5–9. [Online]. Available: http://www2.dac.com/events/eventdetails.aspx?id=295-151

[41] Z. Xie *et al.*, "RouteNet: Routability prediction for mixed-size designs using convolutional neural network," in *Proc. ICCAD*, 2018, pp. 1–8.

[42] X. Xu, Y. Lin, V. Livramento, and D. Z. Pan, "Concurrent pin access optimization for unidirectional routing," in *Proc. DAC*, 2017, pp. 1–6.

[43] A. M. Ziesemer and R. A. da Luz Reis, "Simultaneous two-dimensional cell layout compaction using MILP with ASTRAN," in *Proc. IEEE Comput. Soc. Annu. Symp. VLSI*, 2014, pp. 350–355.

[44] Q. Zhou, X. Wang, Z. Qi, Z. Chen, Q. Zhou, and Y. Cai, "An accurate detailed routing routability prediction model in placement," in *Proc. ASQED*, 2015, pp. 119–122.

[45] *An Open-Source Knight's Tour Solver by Warnsdorff-Rule Algorithm*. Accessed: Aug. 2, 2020. [Online]. Available: https://github.com/NiloofarShahbaz/knight-tour-warnsdorff

[46] *Cadence Innovus User Guide*. Accessed: Aug. 2, 2020. [Online]. Available: http://www.cadence.com

[47] O. Tange. (2018). *GNU Parallel*. [Online]. Available: https://doi.org/10.5281/zenodo.1146014

[48] *H2O AutoML*. Accessed: Jan. 14, 2021. [Online]. Available: https://www.h2o.ai

[49] *LEF/DEF Reference 5.8*. Accessed: Aug. 2, 2020. [Online]. Available: http://www.si2.org/openeda.si2.org/projects/lefdefnew

[50] *Liberty Library Format*. Accessed: Aug. 2, 2020. [Online]. Available: http://www.opensourceliberty.org

[51] *OpenCores: Open Source IP-Cores*. Accessed: Jan. 14, 2021. [Online]. Available: http://www.opencores.org

[52] K. Jeong, A. B. Kahng, and H. Yao. *RentCon: Rent Parameter Evaluation Using Different Methods*. Accessed: Jan. 14, 2021. [Online]. Available: https://vlsicad.ucsd.edu/WLD/index.html

[53] C.-K. Cheng, D. Lee and D. Park. *SMT-Based Standard-Cell Layout Generator for PROBE2.0*. Accessed: Jan. 14, 2021. [Online]. Available: https://github.com/ckchengucsd/SMT-based-STDCELL-Layout-Generator-for-PROBE2.0

[54] *Synopsys Design Compiler User Guide*. Accessed: Aug. 2, 2020. [Online]. Available: http://www.synopsys.com

[55] *Synopsys IC Compiler II User Guide*. Accessed: Aug. 2, 2020. [Online]. Available: http://www.synopsys.com

[56] *The OpenROAD Project GitHub Repository*. Accessed: Aug. 2, 2020. [Online]. Available: https://github.com/The-OpenROAD-Project