Design Rule Evaluation Framework Using Automatic Cell Layout Generator for Design Technology Co-Optimization

Kyeongrok Jo, Seyong Ahn, Jungho Do, Taejoong Song, *Member, IEEE*, Taewhan Kim, *Senior Member, IEEE*, and Kyumyung Choi

***Abstract*— This paper proposes a complete and full automation framework of evaluating design rules (DRs) to facilitate the process of design technology co-optimization (DTCO), which is highly demanded in 14-nm and beyond technologies. Our proposed framework explores the changes of DRs and evaluates the impacts on the number and types of DR violations as well as the resulting cell/chip layout area. Precisely, the core engine of our DR evaluation framework for DTCO, the automatic cell layout generator, consists of key enabling techniques for standard cell layout optimization. They are integrated coherently to seamlessly support the advanced process technologies using FinFET transistors, complex DRs, and double patterning (DP) lithography. Also, the tight integration of our automatic cell layout generation into the DR evaluation framework with diverse analysis features enables the DTCO process to be much faster and more efficient. We provide a set of experimental data not only to show how much our proposed enabling techniques are effective in optimizing layouts but also to show how effectively our framework explores and analyzes the DTCO parameters (e.g., ground DRs and DP DRs).**

***Index Terms*— Cell architectures, design rules (DRs), design technology co-optimization (DTCO), layout, standard cells.**

I. Introduction

THE process from the development of a new semiconductor process technology to the development of mass production products is traditionally a sequential process, which is a sequence of a process/device, a process design kit (PDK), a library and design kit (DK), an IP, and product developments. Consequently, a considerable amount of effort and time is required to resolve the early-stage problems if they are found in a late stage. In addition, since leading semiconductor industries have developed new process nodes every two years during the last 15 years [2]–[4], it is very hard to fix the early-stage problems in time for the sequential development process, which is the main cause of preventing semiconductor industries from developing the most competitive process and design infrastructure.

Design technology co-optimization (DTCO) is the concept that most advanced semiconductor companies consider to adopt in order to overcome the limitation of the sequential development process. DTCO can produce a competitive product infrastructure by resolving sequential development issues of process and design technologies and by performing co-optimization of process and design technologies. In particular, the usefulness of DTCO is highlighted on the layers in between the development of process technology and the development of design rules (DRs). DTCO between the process and DRs to bring better solutions in terms of both chip size and yield has been studied for over ten years. The DRs in consideration have been diversified and extended, from basic DRs to complex DRs, multiple patterning rules, and so on, in accordance with the advancement of process technologies. Liebmann and Topaloglu [5] summarized candidates of DRs to be considered in DTCO, especially for the process technologies of 10 nm and below.

One of the most important technologies to be established in DTCO research is to develop a framework that can enable a fast and accurate DR evaluation and exploration. To our knowledge, DR evaluator (DRE) developed by Ghaida and Gupta [6] and Ghaida *et al.* [7], [8] is the only work for the DTCO framework. DRE evaluates DRs and cell architectures with respect to cell area, variability, manufacturability, and timing. It creates a "virtual" cell layouts and performs the evaluation. In addition, FinFET technology, complex DRs, and double patterning (DP) are adopted and their impacts are explored. However, the cell area estimation based on DRE "virtual" cell layouts may bring a nontrivial gap in accuracy as follows.

1) It generates transistor placement first, followed by routing estimation based on the single trunk Steiner tree and bounding box method. Despite influencing transistor placement on the quality of routing estimation, it does not consider it.
2) Though the virtual route has good advantage of less dependence on routing architecture, it is not sure that the cell size estimation will keep meaningful accuracy in, for example, 2-D M1 routing architecture, complex rules, and so on.

Manuscript received October 28, 2018; revised February 12, 2019; accepted March 26, 2019. Date of publication May 8, 2019; date of current version July 24, 2019. This work was supported by the Samsung-Seoul National University Collaboration Project under Grant SLSI-201604ND059 and Grant Foundry-201708DD031. *(Corresponding author: Taewhan Kim.)*

K. Jo, T. Kim, and K. Choi are with the School of Electrical and Computer Engineering, Seoul National University, Seoul 08826, South Korea (e-mail: jkr2100@snucad.snu.ac.kr; tkim@snucad.snu.ac.kr; kmchoi@snu.ac.kr).

S. Ahn is with the Memory Business, Samsung Electronics Co., Ltd., Hwaseong 18448, South Korea (e-mail: seyong.ahn@samsung.com).

J. Do and T. Song are with the Foundry Business, Samsung Electronics Co., Ltd., Hwaseong 18448, South Korea (e-mail: jeongho.do@samsung.com, tj.song@samsung.com).

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/TVLSI.2019.2910579

In this paper, we focus on exploring DRs rather than cell architectures (layout styles). Layout style comparisons, such as different poly styles and M1/diffusion power-strap styles, have no significant clue/meaning because industries mostly know the right answers. In addition, we focus on the exploration of the effect of DRs' variation on the area only. This is because, during the DTCO stage, one of the most important concerns is to explore the effect of DRs, which include complex design rules and DP rules, on area. Here, we assume 2-D M1 routing architecture and produce actual routing results. This might bring the issue of less flexibility on tool usage. However, in order to produce meaningful results, we should look into the full dependence flow.

For the automatic generation of standard cell layouts, the previous works have partially automated a few steps while integrating the rest manually. This is mainly because full automation by the previous works produces layouts having inferior quality compared with full manual optimized ones. A good survey for cell layout generation with 1-D two-layer routing can be found in [9]. Though there are approaches to 1-D two-layer routing for internal nets, most industries have preferred to use 2-D single-layer (M1) routing with a minimum usage of M2 resource to ensure 100% routing completion. This is because less M2 usage for internal nets allows more M2 available to be used for the chip-level connection, helping save chip area. This trend has been continued until 10 nm though it depends on foundry by foundry. Therefore, our layout generator targets the 2-D single-layer routing architecture. For 2-D one-layer routing, the following recent works are noticeable: Ryzhenko and Burns [10] solved the routing problem by generating candidate routes for point-to-point segments, finding conflicts (DR violations), and solving a Boolean satisfiability (SAT) instance producing a legal routing for all nets. They claimed a high routing completion rate and a further reduction on M2 usage for given transistor placements and by rerouting the manual layouts. However, it was shown that the work suffered runtime and memory explosion for the cells with 65 or more transistors. On the other hand, Hougardy *et al.* [11] addressed the problem of "leaf" cell layout generation. They generated leaf cell layouts through either single-stack or two-stack transistor placement using the branch and bound technique, followed by internal net routing using mixed-integer programming (MIP). However, such as the work in [10], they did not consider routing quality improvement during the transistor placement.

As a core engine of our DTCO, we develop a new standard cell layout generator to overcome the limitations of the existing works. Specifically, we propose a set of key enabling techniques dedicated to the standard cell generation problem: 1) transistor chaining combined with transistor folding; 2) netlist decomposition; 3) gate poly ordering combined with routing congestion estimation; and 4) 2-D single-layer routing with minimal resource. Also, we develop our standard cell layout generator for 28-nm process technology with a planar transistor first and then migrate it to the advanced process technologies using 14-nm FinFET transistors, complex DRs, and DP lithography. For migrating to the advanced process technologies properly, we devise a new technique for the compaction considering complex DRs. Then, we develop a complete and full automation flow of evaluating DRs to facilitate the process of DTCO. It should be noted that the tight integration of our automatic cell layout generation into the DR evaluation framework with diverse analysis features enables the DTCO process much fast and efficient. The main contributions of this paper can be summarized as follows.

1) We develop a fully automatic and practical standard cell layout generator by devising the following four key enabling techniques to overcome the limitations of the existing works.
   a) We propose a new technique of chaining of folded transistors with no quality degradation to reduce the runtime that the prior works (see [12]) suffer.
   b) We combine prior knowledge by designers with heuristic algorithms. More precisely, we adopt a netlist decomposition method, which keeps feedback loops from mixing each other. This can not only shorten the computation time but also enhance the routing completion rate.
   c) We tightly combine the routing congestion estimation with every instance of the gate poly ordering and devise accurate routing congestion estimation method, especially for 2-D single-layer routing, which increase the routing completion rate.
   d) We propose a new 2-D single-layer routing algorithm, which is particularly effective to the internal routing of standard cells that contain irregular pin positions and dense blockages.
2) We extend our standard cell layout generator to support the advanced process technologies using FinFET transistors, complex DRs, and DP lithography. We propose a new 2-D compaction method in which all complex spacing rules are described by V0 M1 and middle-of-line (MOL) graphs, so that the problem can be solved efficiently using mixed-integer quadratically constrained programming (MIQCP). Also, we decompose layout to adopt DP lithography and develop a built-in internal design rule check (DRC) for efficient DTCO exploration.
3) We develop a complete DTCO framework in which the standard cell layout generator is used as a core engine. It can incrementally change the tuning parameters of DRs and assess the resulting cell/chip area and DR violations very quickly.

The rest of this paper is organized as follows. Section II describes our proposed key enabling techniques that build up the standard cell layout generator, followed by providing experimental results to confirm the effectiveness of the techniques. Section III describes our DTCO framework, called DT-comp, which evaluates DRs, followed by the experimental results of DTCO parameter exploration and analyses in Section IV. Finally, a conclusion is given in Section V.

II. Standard Cell Layout Generator

A. Preliminaries

The generation of standard cell layout for a transistor-level netlist of a particular cell (e.g., 2-input NAND and D-flip-flop)

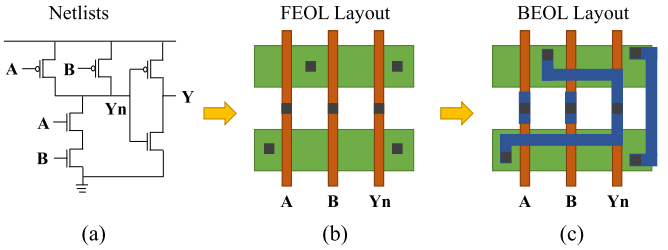

Fig. 1. Illustration of two-step standard cell layout generation. (a) Input (transistor-level) netlist of cell. (b) FEOL layout generated from (a). (c) Cell layout with BEOL generated from (b).

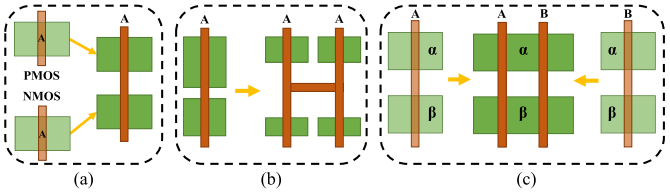

Fig. 2. Three operations performed in the cell topology generation. (a) Transistor pairing. (b) Transistor folding. (c) Transistor chaining.

is commonly performed in two steps: cell topology generation (task 1) and internal net routing (task 2). By task 1, the front-end-of-line (FEOL) layout is determined whereas, by task 2, back-end-of-line (BEOL) routing is completed. For example, for the netlist in Fig. 1(a), an instance of the corresponding FEOL layout is shown in Fig. 1(b), in which the three vertical lines (called gate polys), respectively, indicate the implementation of p-/n-transistor pairs of gates A, B, and Yn. The black dots indicate the contacts, and the two green rectangles represent active areas, the upper one containing p-transistors and lower one n-transistors. An instance of BEOL routing for the nets on the FEOL layout in Fig. 1(b) is shown in Fig. 1(c) where the blue lines indicate the metal routes for nets. Note that the formation and ordering of gate polys in task 1 should be performed in a way that the resulting FEOL layout should be as compact as possible while taking into account a minimal usage of potential routing resources in task 2 as well as satisfying the DR constraints.

If we consider the generation of standard cell layouts with FinFET transistors and DP lithography, two more steps are required: FinFET technology adaptation (task 3) and layout decomposition for DP (task 4), in which task 3 includes FinFET transistor structure, MOL, and complex DRs, while task 4 ensures DP compliant standard cell layouts.

1) Cell Topology Generation: Four main operations that are used to produce FEOL layout are transistor pairing, transistor folding, transistor chaining, and gate poly ordering. Transistor pairing refers to aligning a p-/n-transistor pair with the same gate signal on a vertical gate poly [see Fig. 2(a)]. Since pairing helps generate a compact FEOL layout by simplifying connections between their p- and n-transistors, it is used as a preprocessing step to pair off as many transistors as possible. Transistor folding refers to splitting a transistor with large width into multiple small ones in parallel connection [see Fig. 2(b)]. Since the folded transistors can abut on each other by sharing their source/drain signals, they can be

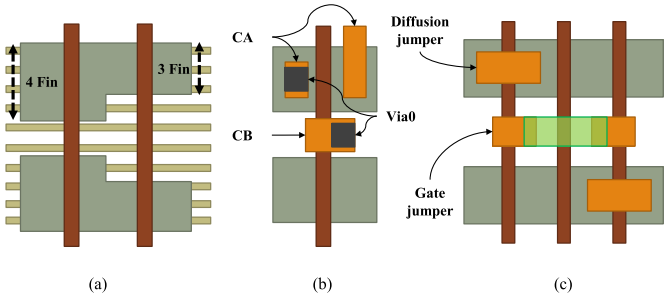

Fig. 3. FinFET technology considerations. (a) Number of fins in a transistor. (b) Connections through MOL layer. (c) Special constructors: jumpers.

implemented on the same active region. Transistor chaining is to abut transistors of the same type (p or n) by sharing the same active region [see Fig. 2(c)]. Thus, to save the cell area, it is essential to abut transistors into an active region as many as possible. At last, gate poly ordering determines a placement order of gate polys, considering their abutment relation. To enable 100% completion of net routing at the internal net routing stage, the operation of gate poly ordering should be properly exploited, tightly coupled with an accurate routing congestion estimation.

Note that during transistor chaining, all ordering candidates of gate polys need to be explored. Thus, as the number of gate polys in a chain increases, which usually occurs when transistor folding is internally applied, the demand for time/space will drastically increase. To resolve this problem, we address the interwinding optimization problem of transistor folding and chaining.

2) Internal Net Routing: Actual routing will be performed for the selected gate poly placement (ordering) that may have the least value of routing congestion estimation. The goal is to route all internal nets using M1 resources while minimizing M2 usage if M1 is exhaustively used. The routing is performed in two steps: routing for local nets and routing for crossing nets. The local nets are those whose terminals all reside only on the top or bottom active regions or gate polys, whereas the crossing nets are those that should be across over different regions (top active, bottom active, and gate). The routing patterns, including contacts, M1, via, and M2, are fixed, as shown in Fig. 1(c).

3) FinFET Technology Adaptation: Three unique characteristics related to the FinFET technology are the discrete number of fins on a transistor, MOL, and special constructors [13], [14]. Unlike the continuous width in the planar technology, the driving strength of FinFET transistor of 3-D channel increases as the number of its fins increases. For example, the left transistor in Fig. 3(a) has the higher driving strength than the right transistor. Thus, the description of a netlist and transistor folding should include the number of fins on each transistor. On the other hand, MOL layers are described by CA and CB, which are used to make a connection between FEOL and BEOL through MOL, that is, to replace every contact in planar technology with CA or CB, as shown in Fig. 3(b), in which the contact inside of CA or CB is called V0 (Via0) and connects the layers between

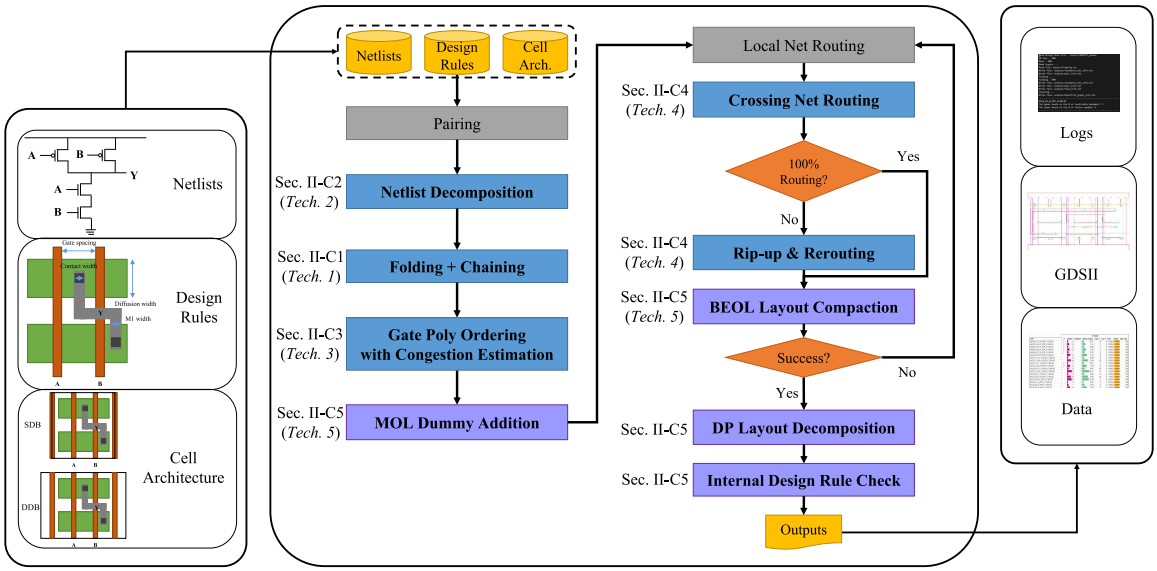

Fig. 4. Our proposed flow for automatic generation of standard cell layouts. Purple boxes support FinFET technology.

BEOL and MOL. In addition, two special constructors on MOL layers are used to mitigate routing congestion by connecting common signals with MOL instead of metal. One is diffusion jumper, as shown on the upper part in Fig. 3(c), which is used to connect a pair of source and drain with a common signal divided by a gate poly. The other is gate jumper, as shown on the lower part in Fig. 3(c), which is used to connect gate polys of the same signal that are separated by a gate poly of different signal.

4) Layout Decomposition for Double Patterning: The resolution limit of lithography in advanced technology nodes requires multiple patterning so that the routing patterns in a layer can be assigned to multiple masks. In this paper, we assume litho-etch-litho-etch (LELE)-type DP [15]. For DP compliant layouts, the routing patterns should be decomposed into small pieces to assign different masks to avoid DP spacing rule violations, in which we use the number of violations as the primary objective to be minimized and the number of stitches as the secondary objective to be minimized.

B. Proposed Flow of Cell Layout Generation

Fig. 4 shows our proposed flow for automatic generation of standard cell layouts. The inputs are the transistor-level netlist, DRs to be met, and the preference of cell architecture (e.g., the number of tracks).

Our flow can generate various shapes of cell layouts by different inputs of the DRs and cell architecture constrained by the current process technology and enables designers to explore layouts that are best fitted to the objective of area minimization. The outputs are a cell layout in the GDSII format with its statistical data, such as the number of gate polys and M2 usage, and generation log file. Our proposed techniques are denoted by the colored boxes with their subsection indication in Fig. 4. Note that the boxes with gray and blue colors indicate the techniques that are applied to both the planar and FinFET technologies, while the boxes with purple color indicate the techniques that are applied to the FinFET technology only.

C. Techniques for Cell Layout Optimization

1) Technique 1 (Chaining Combined With Folding): On the one hand, transistor chaining is the process of abutting transistors of the same type by sharing the same active area. We employ the optimal chaining method proposed in [12].

On the other hand, transistor folding is the process of folding a transistor that has a larger width than their (pMOS or nMOS) active height into multiple transistors, in parallel connection, of the same total width. The conventional methods perform transistor folding before transistor chaining in the hope to improve the chaining solution that includes all candidates of gate poly ordering. However, it sacrifices runtime significantly. Note that if all transistors in netlist are folded n times each, the time complexity of the method in [12] is theoretically bounded by $O(n^n) \times T$, where T is the time complexity for the nonfolded case. For this reason, when we observed from experiments that for a test case with over 20 folds, the method in [12] was not able to find a solution in 12 h. Instead, we devise the following three-step technique that can partially integrate transistor folding into transistor chaining.

1) The method in [12] is applied to the bipartite graph of input netlist without folding to produce a minimal number of chains. These chains are used as an initial partition from which we find gate signals to be abutted after folding. Following the step, we perform transistor folding for each chain.
2) We derive two graphs named p-graph and n-graph to represent a possibility of transistor abutment for each pull-up and pull-down logic of input netlist using given transistors in the chain obtained from step 1 and their folded ones. A vertex is a net connected to at least one

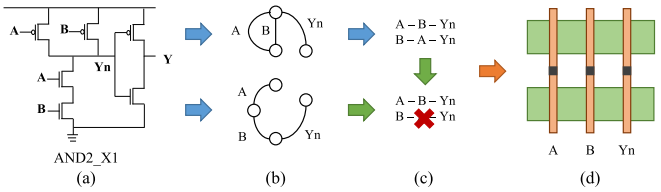

Fig. 5. Example of chaining using the Euler trail-based method. (a) Input netlist. (b) Generation of p-graph and n-graph for (a). (c) Finding Euler trails for p-graph and removal of incompatible trails for n-graph. (d) Single chain consisting of (A, B, and Yn) is extracted from valid trails in (c).

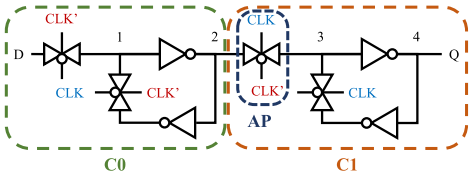

Fig. 6. Decomposing netlist into C0 and C1 subnetlists with feedback loop by cutting off the netlist at the input of transistor AP.

transistor's source/drain. An edge is a transistor denoted with its gate signal and is connected to vertices of its source/drain signals. An example of the graph generation is shown in Fig. 5(a) and (b).

3) Finding a chain that all transistors in the graph are abutted can be transformed to a problem of finding a trail in the graph which visits every edge exactly one, which is the definition of a Euler trail. However, as a chain should have the same p-/n-transistor order, a chain can be generated only when Euler trails from both p-graph and n-graph are the same. In this paper, we used a heuristic in finding Euler trails to make chains. For the p-graph, all possible Euler trails are found using the depth-first search (DFS) recursive algorithm. Each trail is tested if it is a Euler trail in the n-graph. If the test is failed, the trail is considered to be incompatible and removed from the set of valid trails. Valid trails for both graphs are made into chains. As shown in Fig. 5(c) and (d), an incompatible trail ($B-A-Yn$) is removed and an only valid trail ($A-B-Yn$) is made into a chain.

In step 1, an optimal solution may be missing due to initial partitioning. (For example, if three transistor pairs have the same source/drain combination in one side, while another side is all different, it cannot be a chain without folding, but, if any one pair is folded into two pairs, it can be.) Thus, schemes, such as chain flipping and source–drain flipping, are used and solutions are checked if they can be chained further.

2) Technique 2 (Netlist Decomposition): The complexity of generating a minimal number of transistor chains together with the number of candidates of gate poly ordering of the chains exponentially increases as the number of edges in the bipartite graph model for the input cell increases. We observe that some cells, such as flip-flops, should not necessarily be formed into a single bipartite graph. An articulation point in netlist graph can be a candidate to split the netlist into multiple parts, each of which is then transformed into a model of the separate bipartite graph. We perform the splitting by removing the articulation point only if each of the resulting connected components has a signal flow of feedback loop. (The transistor node corresponding to the articulation point is included to the bipartite graph of its feeding subnetlist.) Fig. 6 shows an example of netlist decomposition, in which the node labeled with AP is an articulation point, and the subnetlist labeled with C0 and C1 is connected components with the feedback loop. If any transistor in the feedback loop of subnetlist C0 or C1 is moved to the other side, an intersection between feedback loops occurs, which increases routing congestion. This is one differentiated point of our approach, where we combine the designer's knowledge and our heuristic algorithms to produce better solutions. Moreover, we have observed from experiments that this preprocessing step of netlist decomposition not only shortens the computation time of chain enumeration but also reduces the routing complexity, increasing the rate of routing completion, using M1 only, by 22%.

3) Technique 3 (Gate Poly Ordering Combined With Routing Congestion Estimation): We tightly link the routing congestion estimation with every instance of the ordering of gate polys in each chain

$$\mathrm{Bin}_{n_i}(x,y) = \begin{cases} 1, & \text{if } (x,y) \text{ is occupied by a pin} \\ \dfrac{m+n-1}{mn}, & \text{otherwise} \end{cases}$$

$$\mathrm{Con}(x) = \sum_{y}\sum_{n_i} \mathrm{Bin}_{n_i}(x,y), \quad x = 1,\ldots,N_X$$

$$\mathrm{CellCon} = \frac{\sum_x \mathrm{Con}(x)}{N_X} \cdot w + \max_x\{\mathrm{Con}(x)\} \cdot (1-w)$$

N_X: the last X-axis index of bins

w: a weighting factor ($0 \le w \le 1$). (1)

We define bin as the intersection x- and y-grid. x-grid contains vertical routing resources along the poly and in the middle of two polys for diffusion. y-grid is the same as the horizontal metal tracks. Our estimation method assumes that nets are routed in these grids and a net n_i is routed inside of its bounding box BB_{n_i} of $m \times n$ size. We define $\mathrm{Bin}_{n_i}(x,y)$ for a net n_i in (1) to represent the expected track occupation at bin location (x,y) in BB_{n_i}. If the bin is occupied by a pin, $\mathrm{Bin}_{n_i}(x,y)$ is 1. Otherwise, it is set to the ratio of the number of bins that amounts to covering the half-perimeter of BB_{n_i} to the total number of bins in BB_{n_i}. We accumulate the values of $\mathrm{Bin}_{n_i}(x,y)$ for all nets, which are then used to the estimated vertical congestion $\mathrm{Con}(x)$, in (1), at X-location x. For example, Fig. 7 shows the values of $\mathrm{Bin}_{n_i}(x,y)$ and $\mathrm{Con}(x)$ for two nets of sizes 3×6 and 2×4. Then, we formulate the routing congestion, CellCon, as a weighted sum of the average and peak values of $\mathrm{Con}(x)$, as expressed in (1). CellCon values of every gate poly order are used to enable an easy internal routing.

The routing congestions of gate poly-ordered candidates are computed by this method. Then, the congestions are arranged from the lowest to the highest and take one-by-one in the following process (from "MOL dummy addition" to "Internal DRC" in Fig 4). After that, we choose the best solution in

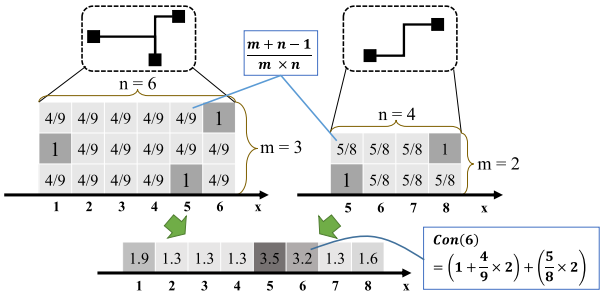

Fig. 7. Illustration of our congestion estimation. Congestion estimation of two multipin nets of size 3×6 and 2×4. The first step is calculating the congestions on the bins in the left bounding box, in which the pins of the net are located in positions (1,2), (5,1), and (6,3). Thus, the probability of bin occupation is 1 for each of those pin locations, while the probability becomes the value of $(m+n-1)/mn$ for each of the remaining bins. The next step is then calculating the horizontal congestions. Each estimated congestion value, as shown at the bottom, is obtained by summing all congestion values of the bins on the same x-coordinates in all nets.

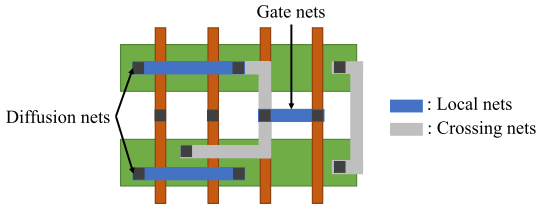

Fig. 8. Illustration of local nets and crossing nets.

terms of area, routing completion, and the number of DR violations. The use of our estimation increases the routing completion rate, while using M1 resource only, by 9% for flip-flops, showing its strong effectiveness especially on the standard cells with high routing complexity.

4) Technique 4 (2-D Routing With Minimal Resource): We classify the internal nets as illustrated in Fig. 8 and as follows.

1) *Diffusion Nets:* Nets whose contacts are in p- or n-active, i.e., connections among the sources/drains of the same type (p or n).
2) *Gate Nets:* Nets whose contacts are in gate polys.
3) *Local Nets:* A collection of diffusion and gate nets.
4) *Crossing Nets:* A collection of nets that are not local.

Routing for Local Nets: We apply Maze routing [16] for local nets. Our careful consideration is that instead of applying the common approach of adding Steiner point to Steiner tree, we develop a new method for finding "virtual" Steiner points. As shown in Fig. 9, during maze routing from source to destination, if the propagation from source reached the same net's already routed point (see the Routed net in Fig. 9), backpropagation begins from that point rather than the destination. Therefore, actual routing becomes a connection between the source and the point, which is regarded as "virtual" Steiner point.

Routing for Crossing Nets: Crossing net routing is performed after the completion of routing diffusion and gate nets in order to prohibit blockages on horizontal metal tracks by vertical routing for crossing net. Thus, in crossing net routing, there may exist lots of blockages. In addition, only

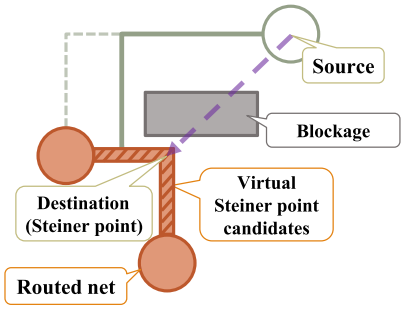

Fig. 9. Illustration of virtual Steiner point.

M1 is the available resource. Thus, it is very important to solve the routing planarity (i.e., 2-D single-layer) in this stage. Here, we precisely apply the concept of rubber band equivalent (RBE) and critical cut [17] to solve the routing planarity.

1) *RBE:* It represents the shortest path connecting two points, making a detour around blocking.
2) *Critical Cut:* It represents a line starting from corners and tips of a block and ending at the nearest boundary of another block.

The flow of a critical cut is defined as the number of RBEs crossing the cut, and the capacity of a critical cut is defined as the number of RBEs that can pass the cut. Thus, if there is no critical cut in which its flow exceeds its capacity, M1 planar routing for all crossing nets is feasible. We develop a new method to find the optimal points of each region (p-active, n-active, and gate poly) to connect crossing nets through integer linear programming (ILP) formulation where planarity, connectivity, and resource usage are considered simultaneously. The variables used in our ILP formulation are listed as follows.

$x_i ab$: Set to 1 if the subnet of net i between terminals a and b is connected, and 0 otherwise.
$w_i ab$: Manhattan distance of the subnet of net i between terminals a and b.
C_k : Capacity of critical cut k.

The objective and constraints are

$$\begin{aligned} &\text{min.} \sum_i \sum_a \sum_b (w_{iab} \cdot x_{iab}) \\ &\text{s.t.} \sum_a x_{iab} \geq 1 \text{ for } \forall i,\ b \\ &\quad \sum_b x_{iab} \geq 1 \text{ for } \forall i,\ a \\ &\quad \sum_{\text{Every subnets crossing k}} x_{iab} \leq C_k \text{ for } \forall i. \end{aligned} \tag{2}$$

The first two constraints ensure the net connectivity. The third constraint ensures the routing planarity. The objective is to minimize the resulting total wire length.

After finding the solution, each pair of pins in the endpoint of selected RBEs is connected by Maze routing.

Rip Up and Rerouting: For unrouted nets, we use the following two schemes. If nets are not routed because of local blockages, planarity, and sequentiality issues, scheme 1 is applied with the consideration of detour around the M2 layer if needed. If nets are still unrouted because of insufficient routable resources, scheme 2 is applied.

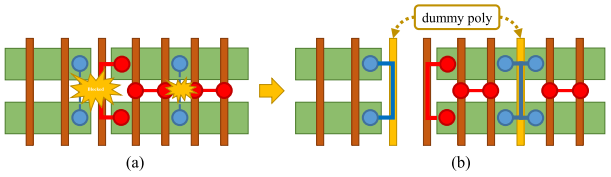

Fig. 10. (a) Before adding dummies. Blue lines: nets to be routed. Red lines: blocking nets. (b) After adding dummies. One dummy is added to the outside of active, and another dummy is inserted to active, requiring two jumpers.

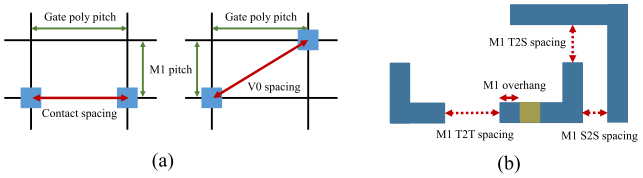

Fig. 11. Illustrations of important complex DRs considered in this paper. (a) V0 spacing. (b) M1 overhang and M1 spacing rules.

Scheme 1 (Rip Up and Reroute): We rip up the nets that satisfy one of the following two properties: if the straight projection line of the routing candidate intersects any routed net (*projection_crossing*) and if more than two sides of a terminal point of a routing candidate are surrounded by other nets which are movable (not a contact) (*surround_blocking*).

Scheme 2 (Adding Dummy Polys): When the vertical routing resource in M1 is not sufficient, we attempt to add dummy polys. For example, when the blue lines in Fig. 10(a) cannot be routed due to the already routed nets shown in red, two dummy polys are added, as shown in Fig. 10(b) to make them routable. Note that if a dummy is placed outside an active region, it becomes a diffusion break, while if a dummy is inserted to active, jumper(s) is needed to fly over the dummy.

5) Technique 5 (Complex Design Rules and Double Patterning): The complex DRs considered in the FinFET process technologies can be classified into four categories.

1) V0 spacing [i.e., the spacing specified by the red lines in Fig. 11(a)], which becomes larger than the gate poly pitch, causing the gate V0 placement much hard. For example, the spacing by the left red line in Fig. 11(a) is no more valid, and thus, the right contact needs to be moved up.
2) M1 overhang [i.e., the more metal spacing specified by the red solid line on a metal in Fig. 11(b)] to guarantee V0 connection to M1.
3) M1 tips [i.e., three spacing types: tip to side (T2S), tip to tip (T2T), and side to side (S2S) specified by the red dotted lines in Fig. 11(b)], which should increase to prohibit DR violation after optical proximity correction.
4) MOL spacing that considers CA and CB space.

The four complex DRs make the gate V0 placement and M1 routing much hard for a given area. In the following, we propose a preprocessing, called dummy poly addition, and a main task, called compaction considering all complex DRs, of layout generation that meets all the additional DRs or minimizes the number of DR violations if there is no legal layout. Subsequently, our last step for performing DP is performed. Note that to our knowledge, there is no work in the literature that has considered the complex DRs all together in the automatic layout generation.

Dummy Poly Addition: This step inserts dummy polys so that the FEOL layout produced does not violate the V0 spacing and M1 S2S spacing rules. As shown in the process following the left arrow in Fig. 12, by transforming the spacing relation into a graph G, in which a vertex represents a V0 or a vertical M1 and an edge exists if there is V0 or M1 S2S spacing

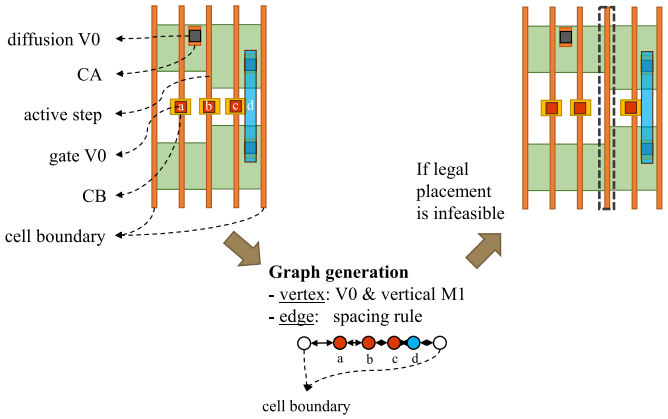

Fig. 12. Graph model of V0 and M1 S2S spacing constraints, and an illustration of dummy poly insertion.

constraint between the two terminals of the edge, we derive a set of inequality constraints for the edges in G, together with the horizontal spacing (to be minimized) between the left and right cell boundaries. Then, we formulate the problem into linear programming (LP) with a set of inequality constraints to be met. If the constraints are not satisfied, the determination of insertion location of a dummy poly is performed in a greedy manner. The primary location is where a gate poly has already been placed over an active step [see the location of dummy poly in Fig. 12 (right)] since the active step is a vulnerable area to manufacturing yield, while the secondary location is a region of light routing congestion. Once a dummy poly is added, the spacing relation graph G and its LP formulation are updated accordingly, and the process repeats until all spacing constraints are met.

M1-V0-MOL Compaction Considering All Complex Design Rules: This step converts all complex spacing rules (i.e., V0, M1 overhang, M1 tips, and MOL CA and CB spacing) in an FEOL layout with dummy poly insertion into a set of two graphs, called V0-M1 graph and MOL graph to model both of the horizontal and vertical spacing constraints among the V0 and M1 instances. As shown in Fig. 13(a), the V0-M1 graph models the spacing constraint (edges) between gate V0 instances (red nodes), M1 segments of two terminals (blue and purple ones), and cell boundaries (white ones), in which V0 instances can move vertically or horizontally, while the horizontal M1 segments can move on the vertical direction only and the vertical M1 segment can move on the horizontal direction only, as represented by movable area (box with dashed line). On the other hand, the MOL graph models the spacing relation among the MOL CA and CB instances, which can move in a vertical or horizontal direction.

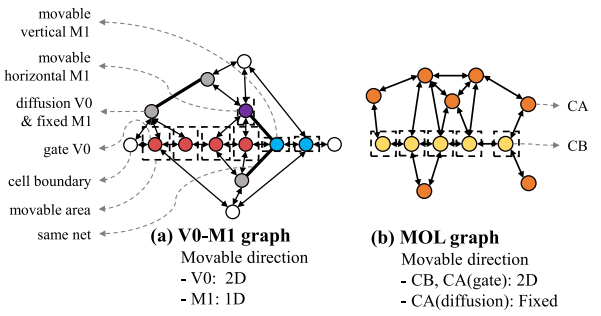

Fig. 13. Graph models for representing the spacing rules. (a) Models for spacing constraints among V0 and M1 instances. (b) Models for spacing constraints among MOL CA and CB instances.

We formulate the (grid-based) spacing constraints in the V0-M1 and MOL graphs into a mixed-integer quadratic programming (MIQCP) as follows.

Variables:

1) (x_i, y_i): Position of a V0, M1, CA, or CB vertex v_i.
2) dx_i, dy_i: Displacement of v_i.
3) $d_{i,j}$: Spacing limit between v_i and v_j.
4) $mx_{L_i}, mx_{R_i}, my_{U_i}$, and my_{D_i}: Boundary position of MOL vertex v_i.
5) min w, h: Minimum width and fixed height of gate MOL vertex.
6) $V0$_enc_MOL: V0 enclosure rule within MOL.
7) p_{L_i} and p_{R_i}: Left/right boundary of the ith gate poly.
8) ac_{U_i} and ac_{D_i}: Upper/lower boundary of active with the ith gate poly.
9) MOL_poly_sp and MOL_ac_sp: Minimum spacing rule between MOL and poly/active.

Objective:

$$\text{Min} \sum_{\text{movable } v_i} dx_i + dy_i. \tag{3}$$

Constraints:

1) *Movable Area:*

$$x_{i,\min} \le x_i \le x_{i,\max} \text{ and } y_{i,\min} \le y_i \le y_{i,\max}. \tag{4}$$

2) *1-D Spacing Rule for M1, CA, and CB:*

$$x_i - x_j \ge d_{i,j} \text{ or } y_i - y_j \ge d_{i,j}. \tag{5}$$

3) *2-D Spacing Rule for V0:*
Euclidean distance is quadratically approximated by using the technique in [18] as follows:

$$\begin{aligned} a * (&|x_i - x_j|(z_i + (\sqrt{2}-1)(1-z_i)) \\ &+ |y_i - y_j|((1-z_i) + (\sqrt{2}-1)z_i)) < d \end{aligned} \tag{6}$$

where $a = 2/(1+\sqrt{1+(\sqrt{2}-1)^2}) - \alpha$, and $z_i = 1$ if $|x_i - x_j| >= |y_i - y_j|$ and 0 otherwise. α is constant that makes approximation result lower than the actual 2-D distance value.

4) *Gate MOL Width and Height Rules:*

$$mx_{R_i} - mx_{L_i} \ge \min w \text{ and } my_{U_i} - my_{D_i} = h. \tag{7}$$

5) *V0 Must Be Within Mol:*

$$\begin{aligned} mx_{L_i} + V0\text{\_enc\_MOL} <= x_j <= mx_{R_i} - V0\text{\_enc\_MOL} \\ my_{D_i} + V0\text{\_enc\_MOL} <= y_j <= my_{U_i} - V0\text{\_enc\_MOL}. \end{aligned} \tag{8}$$

6) *MOL Spacing Between Poly and Active:*

$$\begin{aligned} mx_{L_i} &>= p_{R_i} + \text{MOL\_poly\_sp} \\ mx_{R_i} &<= p_{L_{i+1}} - \text{MOL\_poly\_sp} \\ my_{U_i} &<= \text{ac}_{D_i} - \text{MOL\_ac\_sp} \\ my_{D_i} &>= \text{ac}_{U_i} + \text{MOL\_ac\_sp}. \end{aligned} \tag{9}$$

Double Patterning: For DP compliant layouts, we need a layout decomposition method of the DP for the layouts produced in Section II-C5. There have been many layout decomposition methods for meeting DP spacing rules, most of which focus on (primary objective) finding a minimal number of coloring conflicts on a conflict graph of DP spacing rules for a layout, and then (secondary objective) minimizing the number of stitches to resolve the conflicts. Among the layout decomposition methods, we adopt the work in [19] that addresses the decomposition on the inside of layout, which exactly matches our intention. [Note that most DP works (see[20] and [21]) consider the patterning spacing on the outside of cell layouts, not inside of the cells.] A stitch candidate found in [19] is an interval, on which exact location is not fixed. As it may cause DR violations if the colors of neighboring metals and stitch overlapping rules are considered, we propose a method to locate the exact stitch position considering them. Our method deliberately finds legal stitch location based on stitch overlapping rule while considering the colors of neighboring metals.

Details on Our Internal DRC: DTCO exploration requires fast analyses of DR violations for the changes of DR. However, the conventional DTCO analyses are confined to the environment that has full DR deck and standalone DRC tool. Our DRC environment overcomes the limitation by installing DRC modules considering the DRs of our interest and finding DR violations quickly within the modules.

Precisely, we focus on the following DRs: M1 spacing, elaborated into ground rule (GR), DP rule, and S2S/T2S/T2T tip relations; V0 center-to-center spacing, indicating the Euclidean distance between the centers of two V0 instances; MOL spacing, indicating the Manhattan distance between two of CA and/or CB instances; M1 minimum area, which is the minimum feature size of M1 layer. Thus, the spacing violation occurs when the measured distance is shorter than the corresponding DR, while the area violation occurs when the measured area of a pattern is smaller than the DR corresponding to its shape (rectangular/nonrectangular).

D. Evaluation of the Proposed Techniques

Performance of our proposed standard cell layout generator using techniques 1–4 is evaluated in two perspectives by using the 28-nm planar transistor-based technology. First, the effectiveness is assessed in terms of routing completion and runtime reduction. Second, the quality of layouts is compared

TABLE I
NET SPECIFICATION OF INDUSTRY PARTNER'S (28 nm) CELLS

	#Cells	#Nets	#Nets per cell
Comb. logic	48	354	4-12
Flip-flops	8	151	14-24
Total	56	505	4-24

with the industry partner's (28 nm) manual layouts and DRE's virtual layouts in terms of layout area and M2 usage. Also, the layout migration using technique 5 to support the 14-nm FinFET transistors is assessed first in terms of the number of DR violations, routing completion, and runtime and then in terms of layout area and M2 usage.

1) Performance Analysis of Cell Layout Generator: Our proposed standard cell layout generator was implemented in C++ and integrated it with the SAT solver WalkSAT in [22] and the MIP/MIQCP solver Gurobi in [23]. We used a Linux machine with Intel Core i7-4770K CPU and 16-GB main memory. The experiments are conducted using 45-nm open source FreePDK [24], NanGate open standard cell library [25], industry partner's 28-nm PDK, and standard cell library with 56 cells. For the industry partner's library, though we perform experiments using a subset of the full standard cell library, our tested netlists are composed of various structures of combinational logic, multiplexers, and flip-flops, which are sufficient to describe most of the typical types of standard cells. Table I summarizes the statistic of the industry partner's 28-nm standard cells we tested.

We chose 9- and 11-track cell architectures for both of 28- and 45-nm cell layout generation and generated cell layouts with 100% completion of internal net routing. For the industry partner's 56 cells, the total runtime to get the layouts is taken less than 1 h, and for all the NanGate's cells, the total runtime is less than 3 h.

Assessment of Routing Completion and Runtime: The techniques we proposed has produced promising results in terms of routing completion and runtime reduction. Because these results have been shown our paper in [1], we will skip to show them. Please refer [1] for more detailed experimental results.

Assessment of Layout Area and M2 Usage: For 45-nm FreePDK and NanGate cell netlists, we cannot provide a fair comparison of our layout results with the NanGate's layouts. The reason is that we assumed the cell architectures with 2-D M1 routing, a minimal use of M2, and 1-D fixed pitch gate poly used solely for p-/n-transistor pair connection, while the NanGate cells used unrestricted 2-D poly architecture, in which poly can also be used for internal routing. To the best of our knowledge, most semiconductor companies have not used unrestricted 2-D poly architecture due to the long delay of poly for internal connection. In addition, it is not possible to fairly compare our results with that of the prior works [10], [11] since their inputs, such as netlist, cell architectures, and DRs, are not identical to ours. Instead, we provide a comparison with industry partner's 28-nm layout developed manually and DRE's virtual layout. For DRE, we used industry partner's 28-nm PDK and standard cells, as ours does. The quality metrics are cell area and M2 usage.

TABLE II
CELL AREA DIFFERENCE BY OURS AND DRE'S WITH RESPECT TO THE INDUSTRY PARTNER'S LAYOUT

		Area diff. (count (ratio))	
Cell type	Cell count	ours/prod.	DRE/prod.
Comb. logic	48	0 (0%)	11 (6∼11% over)
Flip-flops	8	6 (5∼10% over)	7 (4∼18% under)

TABLE III
M2 USAGE DIFFERENCE BY OURS AND DRE'S WITH RESPECT TO THE INDUSTRY PARTNER'S LAYOUT

		M2 usage diff. (count (diff.))	
Cell type	Cell count	ours/prod.	DRE/prod.
Comb. logic	48	8 (6% over)	13 (6% over) 3 (5% under)
Flip-flops	8	8 (8% over)	6 (5% over) 1 (3% under)

Since the cell's width is determined by the number of gate polys times the pitch of gate poly, and cell's height is determined by the number of tracks, which will be given by the chosen cell architecture, we can measure the cell area by counting the number of gate polys. Table II shows the comparison of the area difference between ours versus industry partner's layout and DRE's versus industry partner's layout. For ours, among 56 cells, 50 cells have no change in the area and 6 cells increase area (5%∼10% increase for flip-flops), whereas for DREs, among 56 cells, 38 cells have no change in the area, 11 cells increase area (6%∼11% for combinational cells), and 7 cells decrease area (4%∼18% for flip-flops). When we analyze these results, ours generate more cells of similar size to that of industry partner's cells than DRE's virtual cells. Also, we have questions about the decreased flip-flops area by DRE's virtual cells in comparison with the manual layouts. Finally, when we estimate the digital block area, considering that only cell occupation with the assumption that 60% area is used by combinational cells and 40% by flip-flops, the digital block area overhead is 2%∼4%.

Table III shows the comparison of the M2 usage difference between ours versus industry partner's and DRE's versus industry partner's layout. M2 usage is a ratio of used M2 grids to the total available M2 grid in a cell. Among 56 cells, ours shows different M2 usages at 16 cells and their average difference is 6%∼8% over, whereas DRE's layout shows different M2 usages at 23 cells and their average difference is from 3%∼5% under to 5%∼6% over. The comparison of results shows that our cell layout generator produces more cells of a similar layout to the industry partner's manual layouts than DRE's virtual cells and produces less number of larger layouts than those produced by the layout experts. However, our advantage is that we are able to quickly generate layouts comparable to the manual ones. Furthermore, it is observed that there is a positive correlation between the M2 usages for our layouts and the industry partner's manual ones, which also holds true in the layout area. This trend supports that our layout generator can be used as a reference cell layout generator of the DTCO framework.

TABLE IV

NUMBER OF CELLS AND NET SPECIFICATION FOR CELL TYPES ADOPTED FOR 14-nm TECHNOLOGY EXPERIMENTS

Cell type	#Cells	#Nets	#Nets per cell	Avg. #DR violations
AND/OR	10	68	6.8	0
AOI/OAI	12	100	8.3	1.33
BUF/INV	4	18	4.5	0
MXIT/XOR/XNOR	3	25	8.3	1.33
NAND/NOR	10	67	6.7	0.5
DFF/SDFF	4	62	15.5	7

2) Performance Analysis of 14-nm FinFET Technology Migration: The experiments are conducted using PDK and standard cell library with 43 cells migrated to 14-nm technology referencing information opened in public [26]. For cell architectures, M1 of nine tracks and single diffusion break (SDB) is used, which have also been assumed in the 28-nm experiments in Section II-D1. We adopted, for experiments, a subset of the full standard cell library, as summarized in Table IV.

We produced the layouts of 43 cells in Table IV by using techniques 1–5 described in Sections II-C1–II-C5 targeting 14-nm FinFET technology (denoted by 14 nm for the layouts). However, due to the confidential agreement of the industry partner's 14-nm FinFET technology, we used, as the second-best option, the virtual 14-nm FinFET technology that was migrated from 28 nm one and evaluated the results by comparing to ours of 28-nm planar technology (denoted by 28 nm for the layouts).

Assessment of DR Violations, Routing Completion, and Runtime: For 43 cells, we generated cell layouts with 100% completion of internal net routing. The total runtime is less than 1 h. The avg. # DR violations in Table IV shows the number of DR violations for each of the cell types. The DR violations on some cells occur at the compaction stage of complex DR. Currently, we do not resolve DR violations completely in advanced process technologies, where compaction is necessary in order to resolve complex DR. DR violations can reflect the effect of changing DRs and can perform the DRs evaluation for DTCO.

Assessment of Layout Area: Since 14-nm technology's DRs are shrunk further over 28-nm technology, a direct comparison on the area between 14 and 28 nm is not acceptable. Instead, we use the number of gate polys as a metric for the area comparison, It is a close-to-fair comparison because cell architecture and net connectivity are almost identical for both technologies, and the amount of the difference between the numbers of gate polys tells us how much our layout generator for 14-nm technology is effective.

Fig. 14(a) shows the comparison of the number of gate polys between 28- and 14-nm layouts. On average our 14-nm layouts use 0.4 more gate polys over the 28-nm layouts. Note that for AND/OR cells, our 14-nm layouts use 1.6 fewer gate polys, which is caused by the replacement of (string driving strength) X4 cells with (weak driving strength) X2 cells. Fig. 14(b) shows the cell count for the difference of the numbers of

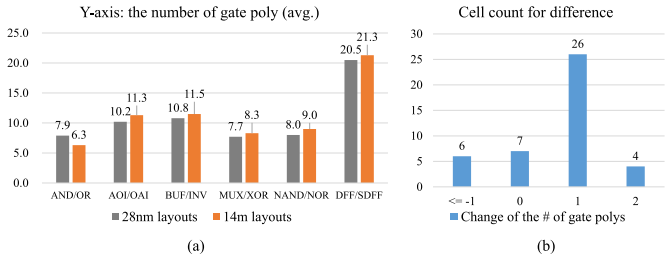

Fig. 14. Comparison of area in terms of the number of gate polys. (a) Numbers of (avg.) gate polys per cell, categorized by the type, of 28- and 14-nm layouts. (b) Cell count for the change of the number of gate polys for the migration from 28 to 14 nm.

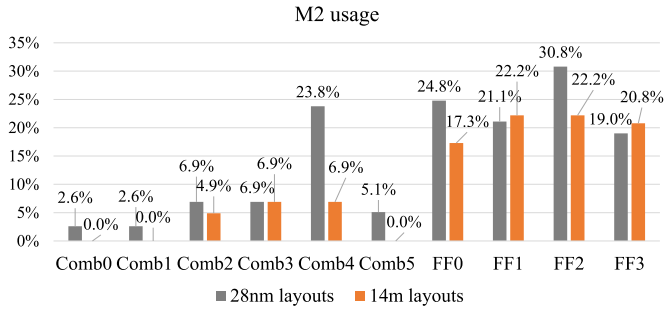

Fig. 15. M2 usage by 28- and 14-nm layouts.

gate polys. The 26 14-nm cells among 43 have one more gate poly, while 4 cells have two more each. The increase in the number of gate polys is mainly caused by the dummy poly insertion due to the insufficient space for gate V0 and MOL spacing rules.

Assessment of M2 Usage: Fig. 15 shows a comparison of M2 usage in percent, showing that on average, our 14-nm layouts use 10% of M2 resource, while 28-nm layouts use 14%. The reason for less usage of M2 is the alleviation of the demand of M2 layer for routing completion, which is caused by the replacement of some of the routing in M1 layer with routing in the MOL layer.

III. FRAMEWORK FOR DESIGN AND TECHNOLOGY CO-OPTIMIZATION

A. Overview

The goal of our DTCO framework, called DT-comp, is to build a completely integrated infrastructure to fully automate the conventional time-consuming back and forth process of evaluating DRs. We achieve the acceleration by enabling our DT-comp to automate the exploration and evaluation of various DRs.

The middle part in Fig. 16 depicts the inputs/outputs and internal structure of DT-comp. The framework receives as inputs the netlist of cells, a set of DRs, and the specification of cell architecture. Then, it explores various layouts by repeatedly generating cell layouts while changing the DRs and measures the layout qualities in terms of the number of DR violations and cell/chip area. Currently, we perform three sets of exploration. The results will be shown in Section IV, each using different tuning parameters: 1) T2S spacing and T2T

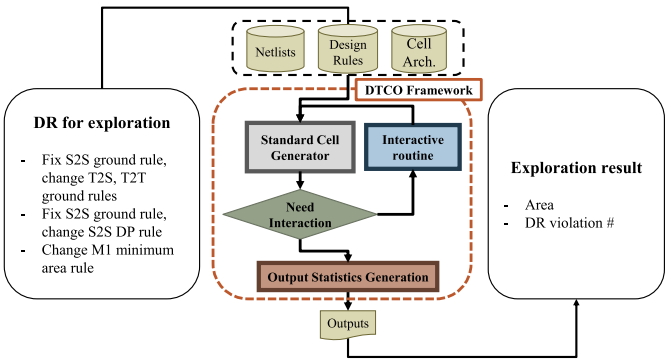

Fig. 16. Conceptual view (i.e., I/O and internals) and flow of DT-comp.

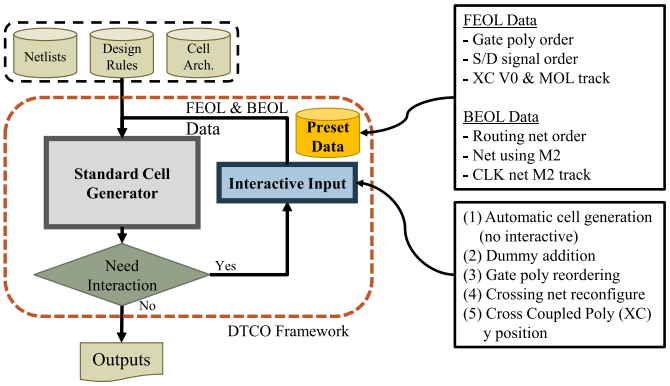

Fig. 17. Interactive flow of DT-comp with supplemental items.

spacing GRs with fixed S2S spacing GR; 2) S2S DP rule with fixed S2S spacing GR; and 3) M1 minimum area rule, as listed in Fig. 16 (left). However, more DRs can be added for DR exploration using this framework easily.

DT-comp can completely automate the full process of generating cell layout and explore the effect of DR changes with two features: interactive routine and output statistics generation, while taking into account the DR constraints for both 28 and 14 nm. Sections III-B and III-C describe the additional features that facilitate the applicability of our DT-comp.

B. Feature 1 (Interactive Flow)

This feature allows users to supplement designer's experience to theoretical knowledge, thereby compensating the limits of automatic cell generation and helping them draw accurate conclusions. The interactive flow and the supplemental items are shown in Fig. 17.

The interactive flow is operated by two options: 1) interruption and 2) preset data.

Option 1: The designers can intervene in the cell generation process by modifying the items, such as the dummy poly insertion, gate poly reorder, crossing net reconfiguration, and y-coordinate reposition of a special constructor that connects XC poly.

Option 2: The designers can choose layouts that they are best on their personal basis and store the layout information in the form of the preset data. Then, DT-comp converts the preset data to be used as an initial layout. The FEOL information in the preset data includes gate poly order, source/drain order, and y-coordinate position of a special constructor that connects XC poly, whereas the BEOL information includes the routing order of internal nets, M2 usage for routing internal nets, and track position occupied by M2 to connect CLK signal.

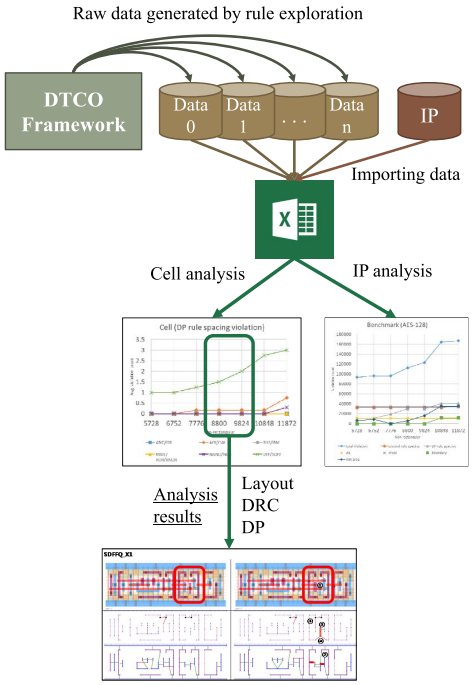

Fig. 18. Graphical view of automatic generation of output statistics by DT-comp.

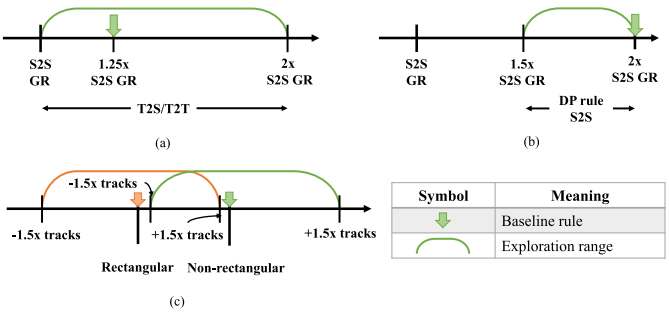

Fig. 19. Exploration range of DR parameters. (a) T2S/T2T GRs with fixed S2S. (b) S2S DP rule with fixed S2S. (c) Minimum area of M1.

C. Feature 2 (Output Statistics)

This feature of generating a statistical data set of cell layouts explored by DT-comp provides users with an easy trace of close-to-Pareto-optimal DTCO points. Fig. 18 shows a graphical view of the generation of layout statistics, which includes the information of cell layout area, number of DR violations, DP data, DR changes, GDSII layout, and the like. In addition, the statistical data on cell layouts can be used in analyzing the impact of cell changes (through the use of a graph form with excel macros) on the IP block level, thus enabling to search for DRs that best matches the target of the IP block.

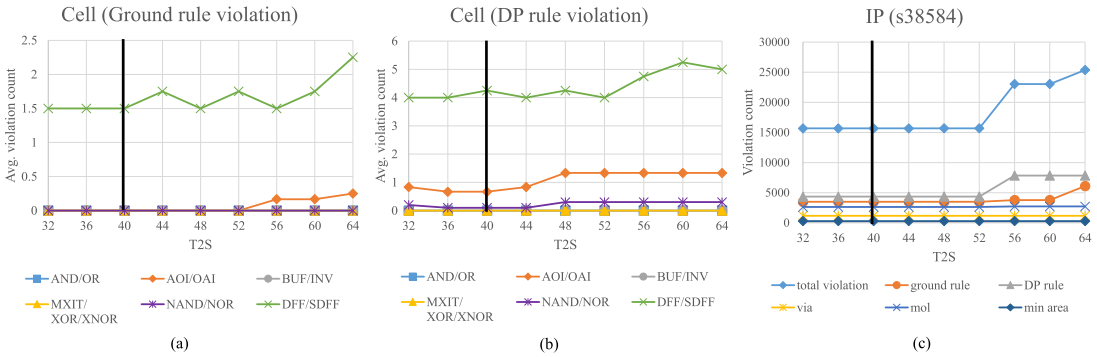

Fig. 20. Changes of the number of DR violations as the T2S/T2T GR changes. Vertical heavy lines: baseline rule. (a) Change of the number of GR violations in the cell level. (b) Change of the number of DP rule violations in the cell level. (c) Change of the total number of violations for each DR in the IP block level.

TABLE V
NUMBER OF CELLS FOR EACH CELL TYPE

Cell type	#Cells
AND/OR	10
AOI/OAI	12
BUF/INV	4
MXIT/XOR/XNOR	3
NAND/NOR	10
DFF/SDFF	4

TABLE VI
DRs USED AS A BASELINE FOR COMPARISON

DR Name	#baseline
S2S GR	32nm
T2S/T2T GR	40nm
S2S DP rule	64nm
T2S/T2T DP rule	80nm
M1 minimum area(rect)	5040nm^2
M1 minimum area(non-rect)	8064nm^2

TABLE VII
NUMBER OF CELL INSTANCES FOR EACH IP

IP	#Instances
s38584	∼7k
Nova	∼109k
AES-128	∼134k
openMSP430	∼6k
USB	∼9k

IV. EXPLORATION RESULTS

We used the 43 cells produced by DT-comp, as listed in Table V. Due to the confidentiality, the 14-nm FinFET DRs cannot be uncovered without the permission of the industry partner. As an alternative, we used the virtual 14-nm FinFET DRs listed in Table VI, as the baseline for comparison. Precisely, the S2S GR is taken from [26] and the S2S DP rule is set to 2 times of the S2S GR, and the tip rule is set to 1.25 times of that of S2S, for both GR and DP rules.

For the IP block-level analysis, we used five benchmarks, each specification of which is listed in Table VII. We synthesized the IP blocks using 45-nm Nangate Library [25] and mapped them to the cells produced by DT-comp. If an IP block has a cell, whose type is not available in Table V, we construct the cell type by using the existing cells. The number of DR violations in an IP block is then the total number of violations of all cell instances.

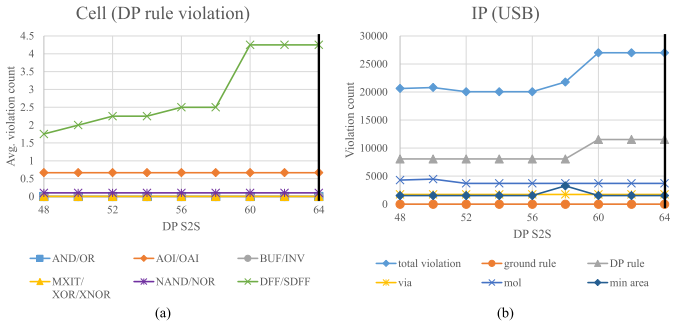

Fig. 21. Changes of the number of DR violations as the S2S DP rule changes. Vertical heavy lines: baseline rule. (a) Change of the number of double pattern rule violations in the cell level. (b) Change of the total number of violations for each of the design rules in the IP block level.

A. Exploration of T2S and T2T Rules With Fixed S2S

Based on the fact that the tip spacing has to be equal to or wider than S2S spacing to take into account the lithography issue, we set the range of tip spacing for exploration to the one shown in Fig. 19(a). The curves in Fig. 20(a) and (b) show the changes in the number of GR violations and the number of DP rule violations, respectively, where each curve starts to increase from 56 nm, which is larger than the baseline rule for both the GR and DP rule. The increase in the number of DP rule violations is mainly due to the increase in the number of layouts that requires bigger T2S/T2T GRs. However, for such case, metals cause coloring conflicts with adjacent metals. Thus, the number of DP rule violations increases as T2S/T2T GRs increase. Fig. 20(c) shows the changes in the total number of violations for each DR in the IP block-level as the value of tip spacing rule changes. Through the analyses, we are able to consider relaxing the T2S and T2T GRs from the baseline.

B. Exploration of S2S Double Patterning Rule With Fixed S2S

We set the baseline of S2S DP rule to twice of the S2S GR. The exploration range of S2S rule is shown in Fig. 19(b).

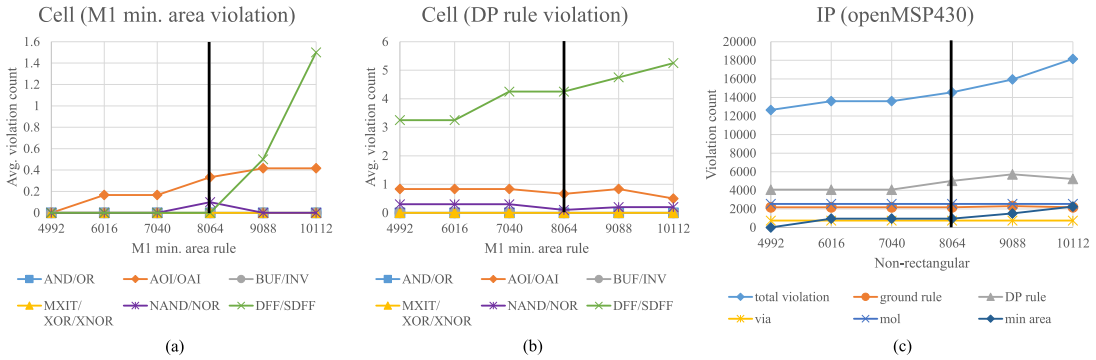

Fig. 22. Changes of the number of DR violations as the M1 minimum area rule changes. Vertical heavy lines: baseline rule. (a) Change of the number of M1 minimum area violations in the cell level. (b) Change of the number of DP rule violations in the cell level. (c) Change of the total number of violations for each DRs in the IP block level.

Fig. 21(a) shows the changes in the number of DP rule violations, in which a sharp drop occurs at 58 nm. Note that since S2S rule makes a little impact on layouts, merely affecting the coloring of metal patterns that have already been fixed, its impact on the other rules is little. Fig. 21(b) shows the changes in the total number of violations for each DR in the IP block level as the value of S2S rule changes. Through the analyses, we are able to consider relaxing the S2S DP rules from the baseline.

C. Exploration of Minimum Area Rule of M1

We set the exploration step size to the value of 0.5 track pitch multiplied by the metal width and set the exploration range of M1 minimum area rule to the interval shown in Fig. 19(c). Fig. 22(a) and (b) shows the numbers of M1 minimum area rule violations and the DP rule violations, respectively, in which the number of M1 minimum area rule violations increases, starting from the baseline rule (because M1 pattern cannot be larger due to an adjacent metal), while the number of DP rule violations increases gradually because the number of DP coloring conflict increases as the metal becomes large. Fig 22(c) shows the changes in the total number of violations for each DR in the IP block level as the value of M1 minimum area rule changes. Through the analyses, we are able to consider additional optimization of M1 minimum area rules from the baseline.

V. Conclusion

This paper first proposed a complete flow for the DTCO framework that was able to fully support an automatic evaluation of DRs. Precisely, as a core engine, we proposed key enabling techniques, namely, transistor chaining combined with transistor folding, netlist decomposition, gate poly ordering, and 2-D single-layer routing with congestion estimation. Then, we developed a standard cell layout generator for the process technology with planar transistors and migrated it to the advanced technology using FinFET transistors and complex DRs. For migrating to the advanced process technologies, we proposed a new technique for the compaction considering complex DRs. Also, we developed a complete and full automation flow of evaluating DRs to facilitate the process of DTCO. We showed that the tight integration of our automatic cell layout generator into the DR evaluation framework with diverse analysis features enabled DTCO process fast and efficient.

We analyzed the performance of our cell layout generator with respect to layout area, routing completion rate, M2 usage, number of DR violation, and runtime for both 28-nm planar transistors and 14-nm FinFET transistors technologies. In addition, we validated the usefulness of our DTCO framework by the exploration of several important 14-nm DRs.

For adopting diverse process technologies, because we considered representative process technologies for planar (28 nm) and FinFET (14 nm) transistors, it could be migrated easily with minor modification. Also, as a future work, we plan to investigate the extended applicability of our DTCO framework: 1) to support 1-D two-layer internal routing of cells and 2) to include a machine learning mechanism on the interactive feature to enhance the evaluation quality.

Acknowledgment

The authors would like to thank the Design Enablement Team at Foundry Business, Samsung Electronics, for the fruitful discussions.

References

[1] K. Jo, S. Ahn, T. Kim, and K. Choi, "Cohesive techniques for cell layout optimization supporting 2D metal-1 routing completion," in *Proc. 23rd Asia South Pacific Design Automat. Conf.*, Jan. 2018, pp. 500–506.
[2] M. Chang, "Foundry future: Challenges in the 21st century," in *Proc. IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2007, pp. 18–23.
[3] D. James, "Moore's law continues into the 1x-nm era," in *Proc. 27th Annu. SEMI Adv. Semiconductor Manuf. Conf.*, May 2016, pp. 324–329.
[4] M. T. Bohr and I. A. Young, "CMOS scaling trends and beyond," *IEEE Micro*, vol. 37, no. 6, pp. 20–29, Dec. 2017.
[5] L. W. Liebmann and R. O. Topaloglu, "Design and technology co-optimization near single-digit nodes," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design*, Nov. 2014, pp. 582–585.
[6] R. S. Ghaida and P. Gupta, "DRE: A framework for early co-evaluation of design rules, technology choices, and layout methodologies," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 31, no. 9, pp. 1379–1392, Sep. 2012.
[7] R. S. Ghaida, T. Sahu, P. Kulkarni, and P. Gupta, "A methodology for the early exploration of design rules for multiple-patterning technologies," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, Nov. 2012, pp. 50–56.

[8] R. S. Ghaida, Y. Badr, M. Gupta, N. Jin, and P. Gupta, "Comprehensive die-level assessment of design rules and layouts," in *Proc. 19th Asia South Pacific Design Automat. Conf.*, Jan. 2014, pp. 61–66.
[9] B. Yu, X. Xu, S. Roy, Y. Lin, J. Ou, and D. Z. Pan, "Design for manufacturability and reliability in extreme-scaling VLSI," *Sci. China Inf. Sci.*, vol. 59, no. 6, Jun. 2016, Art. no. 061406.
[10] N. Ryzhenko and S. Burns, "Standard cell routing via Boolean satisfiability," in *Proc. DAC Design Automat. Conf.*, Jun. 2012, pp. 603–612.
[11] S. Hougardy, T. Nieberg, and J. Schneider, "BonnCell: Automatic layout of leaf cells," in *Proc. 18th Asia South Pacific Design Automat. Conf.*, Jan. 2013, pp. 453–460.
[12] C.-Y. Hwang, Y.-C. Hsieh, Y.-L. Lin, and Y.-C. Hsu, "A fast transistor-chaining algorithm for CMOS cell layout," *IEEE Trans. Comput.-Aided Design Integr.*, vol. 9, no. 7, pp. 781–786, Jul. 1990.
[13] R. Topaloglu, "Design with FinFETs: Design rules, patterns, and variability," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design*, Nov. 2013, pp. 569–571.
[14] M. Rashed *et al.*, "Innovations in special constructs for standard cell libraries in sub 28 nm technologies," in *Proc. IEEE Int. Electron Devices Meeting*, Dec. 2013, pp. 9.7.1–9.7.4.
[15] K. Ronse *et al.*, "Lithography options for the 32 nm half pitch node and beyond," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 56, no. 8, pp. 1884–1891, Aug. 2009.
[16] C. Y. Lee, "An algorithm for path connections and its applications," *IRE Trans. Electron. Comput.*, vol. EC–10, no. 3, pp. 346–365, Sep. 1961.
[17] C. E. Leiserson and F. M. Maley, "Algorithms for routing and testing routability of planar VLSI layouts," in *Proc. 17th Annu. ACM Symp. Theory Comput.*, May 1985, pp. 69–78.
[18] M. Barni, F. Bartolini, F. Buti, and V. Cappellini, "Optimum linear approximation of the Euclidean norm to speed up vector median filtering," in *Proc., Int. Conf. Image Process.*, Oct. 1995, pp. 362–365.
[19] R. S. Ghaida, K. B. Agarwal, S. R. Nassif, X. Yuan, L. W. Liebmann, and P. Gupta, "A framework for double patterning-enabled design," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, Nov. 2011, pp. 14–20.
[20] A. B. Kahng, C.-H. Park, X. Xu, and H. Yao, "Layout decomposition for double patterning lithography," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, Nov. 2008, pp. 465–472.
[21] K. Yuan, J.-S. Yang, and D. Z. Pan, "Double patterning layout decomposition for simultaneous conflict and stitch minimization," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 29, no. 2, pp. 185–196, Feb. 2010.
[22] WalkSAT. *Stochastic Local Search For Satisfiability*. Accessed: Sep. 13, 2016. [Online]. Available: https://www.cs.rochester.edu/u/kautz/walksat/
[23] *Gurobi Optimizer*. Accessed: Mar. 21, 2017. [Online]. Available: http://www.gurobi.com/products/gurobi-optimizer
[24] *Freepdk45 45 nm Process Design Kit is an Open-Source*. Accessed: Aug. 30, 2016. [Online]. Available: https://www.eda.ncsu.edu/wiki/FreePDK
[25] (2008). *Nangate 45 nm Open Cell Library*. [Online]. Available: http://www.nangate.com/?page_id=2325
[26] Wikichip. (2015). *14 nm Lithography Process*. [Online]. Available: https://en.wikichip.org/wiki/14_nm_lithography_process

Kyeongrok Jo received the B.S. degree in electrical and computer engineering from Seoul National University, Seoul, South Korea, in 2016, where he is currently working toward the Ph.D. degree at the School of Electrical Engineering and Computer Science.

His current research interests include automatic standard cell generation and design technology co-optimization.

Seyong Ahn received the B.S. and Ph.D. degrees in electrical and computer engineering from Seoul National University, Seoul, South Korea, in 2013 and 2018, respectively, with a focus on resonant clock synthesis methodologies and design technology co-optimization (DTCO).

He is currently with the Memory Business, Samsung Electronics Co., Ltd., Hwaseong, South Korea. His current research interests include placement and routing automation and DTCO.

Jungho Do received the B.S. degree in electrical and electronic engineering from Kyungpook National University, Daegu, South Korea, in 2007, and the M.S. degree in electrical and electronic engineering from the Korea Advanced Institute Science and Technology, Daejeon, South Korea, in 2009.

He is currently with the Foundry Business, Samsung Electronics Co., Ltd., Hwaseong, South Korea.

Taejoong Song (M'05) received the Ph.D. degree in electrical and computer engineering from the Georgia Institute of Technology, Atlanta, GA, USA, in 2010.

He is currently with Samsung Foundry, Samsung Electronics as a vice president. He has authored over 12 journals and 14 conferences and holds 27 patents.

Taewhan Kim (SM'07) received the B.S. degree in computer science and statistics and the M.S. degree in computer science from Seoul National University, Seoul, South Korea, in 1985 and 1987, respectively, and the Ph.D. degree in computer science from the University of Illinois at Urbana–Champaign, Urbana, IL, USA, in 1993.

He is currently a Professor with the School of Electrical Engineering and Computer Science, Seoul National University. He has published over 200 technical papers in international journals and conferences. His current research interests include computer-aided design of integrated circuits ranging from the architectural synthesis through physical designs, specifically focusing on power, thermal, noise, reliability, and 3-D IC design issues.

Dr. Kim is serving as an Associate Editor for *Integration* (VLSI Journal).

Kyumyung Choi received the B.S. and M.S. degrees in electronic engineering from Hanyang University, Seoul, South Korea, in 1983 and 1985, respectively, and the Ph.D. degree in electrical and computer engineering from the University of Pittsburgh, Pittsburgh, PA, USA, in 1995.

He was with Samsung Electronics for 31 years, where he held several positions from an Engineer to the Senior Vice President, where he developed the design methodologies, PDKs, cell libraries, and mixed-signal and high-speed interface IPs for the semiconductor design as the Head of the Design Technology Team and the Infrastructure Design Centre. He was in charge of the semiconductor IC product development, such as CMOS image sensors, display driver ICs, security ICs, and power management ICs. He has been a Visiting Professor with the School of Electrical Engineering and Computer Science, Seoul National University, since 2016. He has authored or co-authored over 30 IEEE conference and journal papers.