Generating Layouts of Standard Cells by Implicit Learning on Design Rules for Advanced Processes

1st Aaron C.-W. Liang
National Chiao-Tung University
Hsinchu, Taiwan (R.O.C.)
chiawei.eed05g@g2.nctu.edu.tw

2nd Hsuan-Ming Huang
MediaTek Inc.(MTK)
Hsinchu, Taiwan (R.O.C.)
hsuan-ming.huang@mediatek.com

3th Charles H.-P. Wen
National Chiao-Tung University
Hsinchu, Taiwan (R.O.C.)
opwen@g2.nctu.edu.tw

***Abstract*—For the advanced process technologies (e.g. finFET with EUV), the design rules (DRs) are the most challenging issue to the generation of cell layouts and all DR violations must be solved in a legal cell layout. However, most of previous works apply explicit encoding on the selected DRs into the routing engine and cannot accommodate the rapid growth on the size and complexity of DRs as the processes continue to advance. Therefore, in this paper, we propose two implicit-learning techniques, (1) experience-guidance learning (EGL) and (2) constraint-driven learning (CDL) for effectively solving such two problems of DRs, and meanwhile develop an automatic cell-layout generation (ACLG) framework for efficiently generating legal cell layouts. The experimental results show that in a finFET-EUV process [1], EGL and CDL successfully reduce all DR violations on eight target cells where each case takes averagely three minutes. As a result, without manual effort, ACLG is capable of generating legal layouts of standard cells by implicit learning on DRs of advanced processes.**

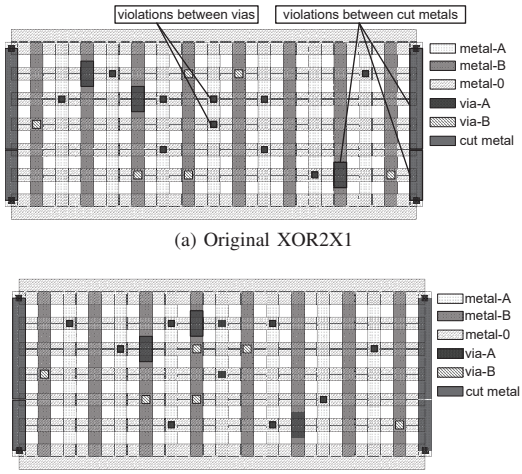
***Index Terms*—cell layout, finFET, EUV, design rules, deep learning, neural network**

I. Introduction

Standard cells play an important role for digital designs of VLSI circuits. Typically, a foundry company will supply the customers (e.g. IC design houses) with a library of standard cells with the guaranteed performance and yield, as well as the design rules, under a specific process [2], [3]. However, a design house may still need to build some additional standard cells to fulfill different requirements of projects.

During the generation of standard cells, performance, power, area (PPA), runtime, and design rules (DRs) are five key factors that need to be considered. For those processes before FinFET technology, many researches focus on optimizing the PPA and runtime of the automatic cell layout generation (ACLG) since the DRs are easily satisfied [4]–[6]. However, with the rapid progress on the process technologies, the complexity and the number of DRs grow drastically. The most critical challenge to layout designers is shifted to how to generate a cell layout that perfectly meet all the DRs.

Fig. 1a shows an example to illustrate how DRs become the most critical problem for the generation of standard cells where a generated layout of XOR2_X1 is optimized for some routing constraints in the finFET-EUV process. Although the area and metal length of the cell are highly optimized, such layout contains 10 DR violations. Even for an experienced physical designer, it is not easy to fix all violations between vias and cut metals since vias are the connecting points of nets and cut metals are used to separate net signals. To fix these 10 DR violations, we must re-route some net signals to widen the spaces between vias and cut metals. In the case, an optimized cell layout is useless before all the DR violations can be cleaned.

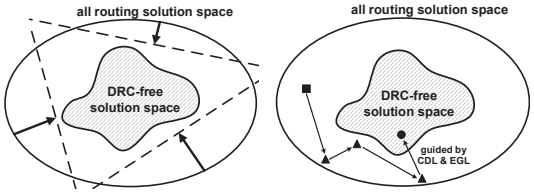


(a) Original XOR2X1

(b) DRC-free XOR2X1

Fig. 1: Example of Critical Design Rule Problem

To deal with the complicated DR problem in the advanced processes, some representative works encoded the conditional DRs explicitly during their ACLG. In [7], the Diff-net DRs and Same-net DRs were considered. In [8], the DRs related to LISD, LIG, SDT layers were encoded in the routing algorithm. Jo et al. in [3] took more DRs such as active (RX), gate poly (PC), and contacts (CA) into account to synthesize layouts with fewer DR violations. Most recently, the work [9] even considered multi-pattern-aware DRs such as parallel run length rule and step height rule to prevent the complicated DR violations. All the previous works aimed at finding the solution space of DR violation-free region where each explicit DR was encoded as a constraint in the routing engine to rule out partial solution space as shown in Fig. 2a.



(a) explicit encoding (b) implicit learning

Fig. 2: Concept of Explicit Encoding and Implicit Learning

However, it is almost impossible to enumerate all DRs for the advanced processes into the routing engine since the total number of DRs is intimidating and some of DRs are highly complicated. Take the finFET-EUV process technology for example, there are more than ten thousands of DRs. Even if

we consider all DRs for an advanced process, the number of variables and constraints may increase explosively, leading to an intimidatingly long runtime to derive a solution. Moreover, the DRs are highly process-dependent. Different processes have significant differences on their DRs. As a result, it is hard to re-apply the existing routing engine for one process to another. The manual effort to explicitly encode the DRs needs to be paid again once a different process is employed. According to all the arguments that mentioned above, we need a more cost-effective solution to replace explicitly encoding on DRs.

From another different point of view, the generation of a standard cell aims at finding "one" solution with certain targets rather than finding all solutions. Do we really need to explicitly encode all DRs? When a layout designer starts to work on a new layout of one cell, it is natural to derive an initial version on basis of the layouts that he/she has seen before rather than considering all the DRs at once. With the help of commercial design rule checker (DRC), the layout designer needs to fix the DR violations that only involve a small potion of all the DRs.

Such concept can be illustrated by Fig. 2b, which starts from an initial layout (as the square in Fig. 2b) from the functional routing model and then derives a violation-free cell layout (as the circle in Fig. 2b) at the end. Once a DR violation occurs during the exploration of the layout space, experience-guidance learning (EGL) and constraint-driven learning (CDL) will be applied implicitly to deal with these violations which guides ACLG to avoid the violations (as the triangle in Fig. 2b). Therefore, a novel ACLG framework is developed in this paper to deal with DR violations by implicit learning, rather than encoding all DRs explicitly into the routing engine.

For generating cell layouts, the DRs can be categorized into two classes: (1) **geometry-related** DRs and (2) **routing-related** DRs. **Geometry-related** DRs refer to the set of rules that specify the feasible coordinates that vias and cut metals can be placed in the cell layout, where **routing-related** DRs denote the other set of rules that constrain the relationships between routed nets. If two routed nets are placed too close, the components like vias and cut metals of the nets may fail in DRC.

To consider **geometry-related** DRs, experience-guidance learning (EGL) is proposed. EGL builds a neural network model to learn coordinate information for a via or a cut metal from the existing cell layouts and places the components at the correct position. On the other hand, to handle the **routing-related** DRs, constraint-driven learning (CDL) is applied and uses the information of DR violations to implicitly encode those DRs into the routing model as additional constraints. Last but not the least, EGL and CDL are integrated into a iterative flow for violation refinement, as shown in Fig. 3, to minimize the total number of DR violations during the cell-layout generation. With such violation-refinement flow, the cell layout containing several DR violations as shown in Fig. 1a can be transformed into a violation-free cell layout as shown in Fig. 1b automatically in a short time.

To further prove the effectiveness of the proposed EGL/CDL techniques, eight representative cells are used in the experiment. The result shows that both the area and metal length of each cell before and after applying the proposed ACLG framework are comparable while all DR violations of eight cells are solved (100% reduction rate of DR violations) automatically in minutes (i.e. 191.23 (61.71) seconds on average (best case)).

The remaining sections are organized as follows. Section II describes the technical details of the proposed implicit learning techniques, including EGL and CDL. Section III shows the experimental result and provides discussion. Last, Section VI concludes the paper.

II. Methodology

Two key problems await to be solved in the automatic cell-layout generation (ACLG): (1) to generate the cell layout with the correct functionality and (2) to minimize the total number of DR violations in the generated cell layout as many as possible. To achieve the correct functionality during the synthesis of the cell layout, an ILP-based functional routing (IFR) model is developed to derive the legitimate cell layout that can pass the commercial layout versus schematic (LVS) check. For minimizing the DR violations, experience-guidance learning (EGL) and constraint-driven learning (CDL) are incorporated to deal with geometry-related and routing-related issues, respectively. Last, our ACLG combines IFR, EGL, and CDL together to generate a legitimate cell layout with the minimized number of DR violations.

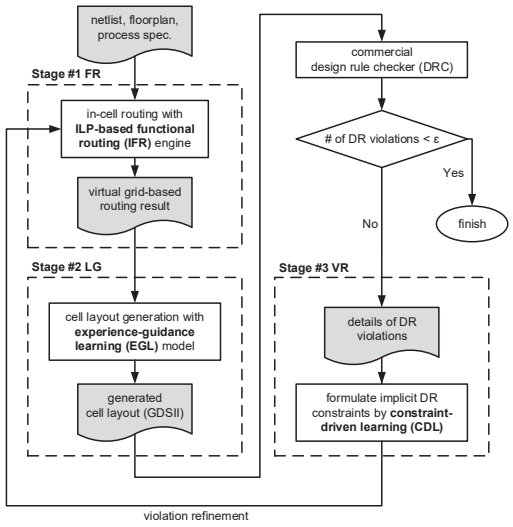


Fig. 3: Automatic Cell-Layout Generation (ACLG) Framework

As shown in Fig. 3, the proposed ACLG framework consists of three core stages, including Stage #1: Functional Routing (FR), Stage #2: Layout Generation (LG), and Stage #3: Violation Refinement (VR). The inputs of ACLG are the netlist and the floorplan of the target cell as well as the specification of the advanced process technology. The output of the ACLG engine is the cell layout in the GDSII format with the minimized number of DR violations. After the information of the target cell and the process is fed into the ACLG engine, the IFR model will accordingly generate a virtual grid-based layout in Stage #1: Functional Routing. All the nets, vias, and cut metals will be assigned to a specific row and column on the virtual grid

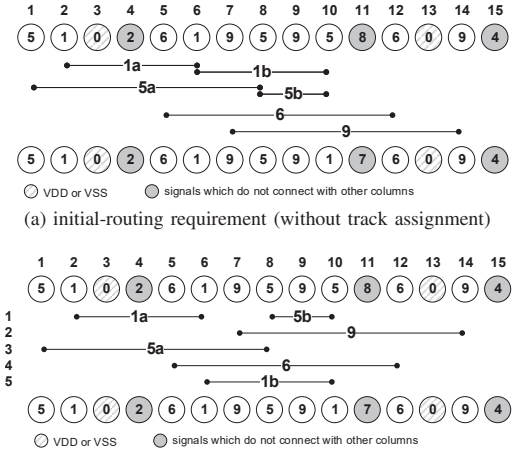
map so that the correctness of functionality of the target cell can be guaranteed.

Stage #2: Layout Generation (LG) is to generate the physical layout in the GDSII format based on the virtual grid-based routing result. The most critical task of the LG stage is to place the vias of the target cell at the accurate coordinates subject to the geometry-related DRs. Experience-Guidance Learning (EGL) is applied to learn the geometry-related DRs implicitly according to the hidden patterns from the existing cell layouts into multiple neural network models for determining the (x, y) locations of the vias of the target cell.

The last stage of our ACLG is Violation Refinement (VR). For most of the cases, the generated cell layouts remain to contain several routing-related DR violations since there is no corresponding DRs explicitly encoded in our IFR model in Stage #1. Therefore, in Stage #3, we apply the constraint-driven learning (CDL) to analyze violations from the report of the commercial design rule checker (DRC) and then add these implicit DR constraints for avoiding the violated DRs to derive a new cell layout with fewer number of DR violations. Violation Refinement (VR) will gradually reduce the DR violations until the total number of violations is fewer than a user-defined threshold (default as 0). Combining two learning techniques (i.e. experience-guidance learning (EGL) and constraint-driven learning (CDL)) in the proposed ACLG framework, those complicated DRs can be learned in an implicit way and the cell layout can be generated with the minimized number of DR violations efficiently.

Stage #1: Functional Routing (FR) By ILP

The ILP-based functional-routing model named IFR in the proposed ACLG is similar to [5], but formulates a virtual grid map for the cell layout without specifying any design rule. The input of the IFR model is the floorplan of the layout that specifies signals at each terminal lying on two ends of the cell. At first, the IFR model will create routing nets in order to connect all the same signals together as shown in Fig. 4a. The terminals of signals that belong to PMOS are placed on the upper part and other terminals which belong to NMOS lie on the lower part of the cell. The terminals that lie on the same column with the same value are connected, accordingly. For example, the signals with value 1 on two-ends of column 2 are connected. And other terminals with the same value that placed on the different columns need to be connected with the routing nets. Take the routing net with value 6 for example, it connects two signals at column 5 and column 12. The terminals with value 0 refer to the VDD or VSS signals of the cell which are connected inherently and need not to be connected through any routing net. The terminals in the gray background are signals which do not connect to other terminals on different columns. Therefore, no routing net will be created for these terminals. After the routing nets are created, the columns of starting points and ending points of nets are also determined. The last problem in IFR is the row assignment to place those net signals. An ILP engine can be used to solve such problem since the net signals and row assignments can be represented by the routing constraints.



(a) initial-routing requirement (without track assignment)

(b) after-routing result (with track assignment)

Fig. 4: Example of ILP-based Functional Routing

Typically, the routing constraints can be classified into four types: (1) track constraints, (2) vertical constraints, (3) horizontal constraints, and (4) via constraints for nets. The track constraints restrict the range of one track that the target net can be placed according to the underlying cell architecture. The vertical constraints define the order of nets if they are overlapped at some columns. The horizontal constraints ensure that enough space can be reserved between the nets which lie on the same track. Last the via constraints check if the space between vias (start points or end points of nets) that lie on the same column are enough or not.

After developing all the routing constraints, ILP solving can be performed to obtain a feasible solution, which properly assigns the row number for each net signal. An example for the output of the IFR model is shown in Fig. 4b. As one can see, every net signal will be assigned to a specific row on the virtual grid map. Take signals with value 5 for example, they are connected with two routing nets 5a and 5b which are assigned to row-3 and row-1, respectively.

Stage #2: Layout Generation (LG) By Experience-Guidance Learning (EGL)

Lack of knowledge or information of the geometry-related DRs, it is impossible to place the vias correctly in the cell layout. Even for a experienced layout designer, he/she needs to observe many prior legitimate cell layouts and consider the relative positions between vias and the metals to be connected for realizing where to place the vias. Besides, more than one design rule may apply to one via placement in the cell layout. Take via-A for example, via-A is used to connect metal-A and metal-0. As Fig. 5 shows, there are several different combinations of placement between metal-A and metal-0 in a single cell. For each kind of combination, the placement rule of via-A will be different based on the positions of metal-A and metal-0. For other cells, there may exist more different combinations of placement between vias and metals. Furthermore, the rules of placing a via can be different for different processes. As a result, it is not easy to explicitly

encode all rules for every possible kind of relationship between metals and vias in ACLG.

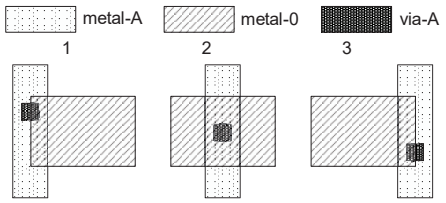


Fig. 5: Example of Different Relative Placement of Via-A

To sum up, the proposed ACLG needs to achieve two goals. The first one is to properly handle all kinds of relative placement between metals and vias without explicitly encoding the geometry-related DRs. The second one is to easily reuse the layout-generating method to other processes without much engineering effort. Therefore, we propose a experience-guidance learning (EGL) technique for the aforementioned goals. The main concept of EGL is to learn the relationship of metals and vias automatically from the legitimate cell layouts existing in the cell library (about 1000 cells) that provided by the foundry companies and then derive the corresponding neural network (NN) models. The well-trained NN model (also called the EGL model) will behave like an experienced layout designer to place the vias at the correct coordinates based on the prior experiences.

The inputs to the EGL model are the (x,y)-coorinates of points from the relative metals and the predicted outputs are the (x,y)-coorinates as the center point of the vias. Take via-A for example, Fig. 6 shows the inputs and outputs to the EGL model. There are total four points contained in one input pattern. Two points from metal-A and the other two points from metal-0. The EGL model then predicts the center point of via-A based on the relationship of input points.

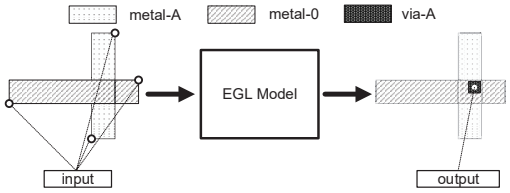


Fig. 6: The Inputs and Output of EGL Model

The architecture of the EGL model is simple as shown in Fig. 7 and only consists of four full-connected layers. Since the geometry-related DRs in advanced processes often restrict the placement of vias, the patterns corresponding to the relationship between metals and vias become more regular. Therefore, such simple model can achieve a satisfactory accuracy.

With the EGL model, the coordinates for the centers of the vias can be predicted according to the relative placement between the fundamental metals. Therefore, there is no need to explicitly encode all geometry-related DRs into our ALCG. Moreover, the EGL model learns the rules implicitly. As we intend to apply our ACLG to another new process, we only need to re-train the EGL model through the legitimate cell layouts in such new process.

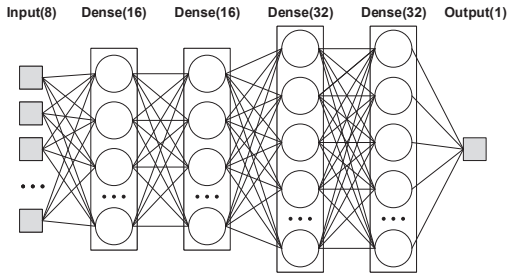


Fig. 7: The Architecture of EGL Model

TABLE I: Notation of Routing Constraints for DRs

Term	Description
N	Total number of routing nets to be constrainted
i	Index of the constrainted nets
net_i	Row variable of the constrainted net
row_i	The original row number that net_i was assigned
M	A large positive number set to 100000
L	Assignment Variance
j	Index of the selection variables
Q_j	The selection variable of absolute value

Stage #3: Violation Refinement (VR) By Constraint-Driven Learning (CDL)

Once the physical layout is generated by the IFR and the EGL model, the generated cell layout needs to be examined by the commercial DRC tool. Since the IFR model do not explicitly encode the complicated DRs, for most of the cases, the generated cell layout still contains few routing-related DR violations. These violations are usually caused by vias or cut metals that are placed too close to each other as Fig. 1a shows. However, the placement of vias and cut metals are determined by the routing nets because the vias are the connecting points of nets and cut metals are used to separate the net signals which lie on the same track. Therefore, to avoid DR violations caused by vias and cut metals, the routing nets need to be re-arranged to widen the space between the related vias and cut metals.

To tame the routing-related DR violations, constraint-driven learning (CDL) is applied. The concept of CDL is to encode the routing-related DRs implicitly by formulating the additional routing constraints on net signals whenever a routing-related DR violation is reported by the DRC tool. The additional constraints will cut off the illegal solution space corresponding to the constrained nets and then guide the IFR model to regenerate a new routing result to avoid such DR violation.

Table I shows the notations of the routing constraints for routing-related DRs and Eq. 1 shows the formulation of constraints for all the target nets.

$$\sum_{i=0}^{N-1} |net_i - row_i| >= L \tag{1}$$

Take Fig. 8 for example, two nets violate certain routing-related DRs in the original placement, in which net_1 was placed at row_1 and net_2 was placed at row_2. To prevent such violations from occurrence, we need to restrict net_1 not be placed at row_1 and net_2 not be placed at row_2, simultaneously. The corresponding constraint is developed as shown in Eq. 2.

$$|net_1 - row_1| + |net_2 - row_2| >= L \tag{2}$$

However, the absolute value representation is typically not supported in the LP file format for an ILP engine. Eq. 2 needs to be transformed into Eq. 3 which imposes the same restrictions on net_1 and net_2. After the transformation, the number of the constraints may grow exponentially, leading to the curse of complexity. Therefore, we only set constraints on the nets which cause violations reported by the DRC tool. According to our real experiences, the total number of nets is at most 4 and thus the proposed IFR model remains to work.

$$
\begin{aligned}
& L >= 1 \\
& 0 <= Q_0, Q_1, Q_2, Q_3 <= 1 \\
& Q_0 + Q_1 + Q_2 + Q_3 = 3 \\
& L >= (net_1 - row_1) + (net_2 - row_2) \\
& L <= (net_1 - row_1) + (net_2 - row_2) + M * Q_0 \\
& L >= (net_1 - row_1) + (-net_2 + row_2) \\
& L <= (net_1 - row_1) + (-net_2 + row_2) + M * Q_1 \\
& L >= (-net_1 + row_1) + (net_2 - row_2) \\
& L <= (-net_1 + row_1) + (net_2 - row_2) + M * Q_2 \\
& L >= (-net_1 + row_1) + (-net_2 + row_2) \\
& L <= (-net_1 + row_1) + (-net_2 + row_2) + M * Q_3
\end{aligned}
\tag{3}
$$

By adding Eq. 3 into our IFR model, the case of placing net_1 and net_2 at row_1 and row_2 will no longer be generated. CDL keeps generating the necessary routing constraints whenever a DR violation is reported by the commercial DRC tool. If there is no DR violation reported (or the total number of violations are smaller than a user-defined threshold (defaulted by 0)), the entire ACLG will stop and output the final cell layout with the minimized number of DR violations.

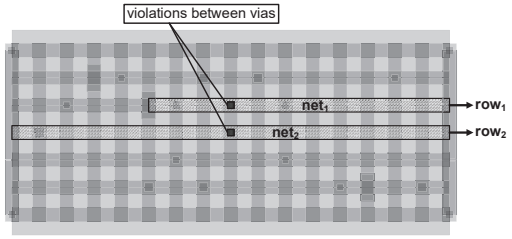


Fig. 8: Example of Routing-Related Design Rule Violation

To sum up, in the proposed automatic cell layout generation (ACLG), the ILP-based functional routing (IFR) model guarantees the functional correctness of the target cell. Critical design rules such as geometry-related DRs and routing-related DRs are solved by experience-guidance learning (EGL) and constraint-driven learning (CDL), respectively. Combining all the IFR, EGL and CDL techniques iteratively, the proposed ACLG framework can automatically generate the desired cell layout with the minimized number of DR violations. Thanks to two implicit learning techniques (EGL and CDL), we do not need to encode any DRs explicitly into the functional routing and the layout generation. Moreover, since IFR, EGL and CDL only rely on the cell layouts in the existing library the proposed ACLG engine can still work on other cell designs in different processes easily.

III. Experimental Result

In the experiments, the ACLG framework is implemented in Python 3.0 and validated on a linux workstation with 3.5GHz Intel Xeon E3-1240 CPU and 8G memory. Eight standard cells with different functions under a finFET-EUV process are used to demonstrate the effectiveness of our ACLG engine. Experimental results will describe the performances of our ACLG on (1) reduction rate of DR violations [1] and (2) runtime breakdown of different stages.

A. Reduction Rate of DR Violations

Table II first shows the reduction rate of DR violations by our ACLG where column 1-3 denote the specifications of the target cells. Column 1 shows the cell names that denote the functions and size of the cells. Column 2 and 3 shows the numbers of transistors and net connections in each cell. The numbers of variables and constraints used in the ILP-based routing formulation are showed in column 4-7. Column 4 and 5 denote the numbers of variables and constraints used in the functional routing formulation, which guarantees the functional correctness of each cell. Column 6 and 7 show the additional routing variables and constraints generated in the DR refinement with CDL to deal with the routing-related DR violations. The numbers of the variables and constraints in our formulation are much smaller compared to those[2] from [9]. Note that the necessary variables and constraints are formulated for the routing correctness in the IFR model. As a result, a large number of variables and constraints that used to explicitly encode DRs can be saved.

Last but not least, column 8-10 show the effectiveness of the design rule refinement by our ACLG. By EGL, the violations due to the geometry-related DRs can be avoided so that the vias can be placed at the correct positions during the LG stage. Therefore, the generated cell layout will only contain the routing-related DR violations which will be dealt with by CDL during the VR stage. Column 8-9 in Table II shows the numbers of DR violations before and after applying our violation refinement. As one can see, all DR violations in each target cell are solved, leading to the 100% reduction rate. In other words, our proposed ACLG engine can generate the violation-free layouts for all eight target cells.

B. Runtime Performance

Table III shows the runtime (including breakdown for those used by three stages and design-rule checking (DRC)) of our ACLG framework on each target cell. Column 4 denotes the runtime of the FR stage which uses the IFR model to generate a gird-based routing result with correct functionality. Column 5 shows the runtime of the LG stage which applies the EGL model to predict the accurate positions of vias and to generate

[1]Prior works [5], [9] focus optimization on wire length and cell area, and cannot guarantee to fix all design rule violations. Therefore, no fair comparison on DR violations can be conducted between our ACLG and prior works.

[2]Unlike [9], transistor placement is not considered as an objective in our ACLG and all design rules are now implicitly learned by EGL and CDL. The total numbers of variables and constraints used in our IFR model are smaller by several hundreds of times.

TABLE II: Comparison of Numbers on DR Violations

Cell Specification			Routing Formulation		DR Formulation		Design Rule Refinement		
Cell Name	#FET	#Net	#Var	#Constraint	#Var	#Constraint	#DRV w/o DR	#DRV w/ DR	Reduction Rate
AOI22_X1	8	12	32	90	15	42	9	**0**	**100%**
NAND4_X1	8	12	18	46	5	14	8	**0**	**100%**
INVD_X1	2	6	6	14	19	52	2	**0**	**100%**
BUFF_X1	3	7	8	19	5	14	8	**0**	**100%**
XOR2_X1	14	12	45	127	10	28	10	**0**	**100%**
XNR2_X1	14	12	45	127	27	78	3	**0**	**100%**
MUX2_X1	12	12	44	128	5	14	8	**0**	**100%**
LHQ_X1	20	15	54	150	5	14	8	**0**	**100%**
Avg.	11.3	11.7	35.9	100.7	9.1	25.6	5.9	**0**	**100%**

TABLE III: Runtime of Proposed ACLG Framework

Cell Specification			Runtime (s)				
Cell Name	#FET	#Net	FR	LG	VR	DRC	Total
AOI22_X1	8	12	0.22	5.86	0.62	515.30	522.00
NAND4_X1	8	12	0.17	1.78	0.18	145.63	147.76
INVD_X1	2	6	0.29	1.93	0.10	145.61	147.93
BUFF_X1	3	7	0.17	1.48	0.15	114.90	116.70
XOR2_X1	14	12	0.56	2.67	0.27	202.29	205.79
XNR2_X1	14	12	0.19	0.81	0.06	60.65	61.71
MUX2_X1	12	12	0.16	1.54	0.16	148.47	150.33
LHQ_X1	20	15	0.19	2.03	0.23	175.17	177.62
Avg.	11.3	11.7	0.24	2.26	0.22	188.50	191.23

the physical layout in the GDSII format. Column 6 shows the runtime of the VR stage, in which CDL iteratively reduces the DR violations and generates the legal DRC-free layout. Column 7 shows the elapsed time for performing DRC to obtain the details of DR violations. Last, column 8 shows the total runtime of our ACLG for each target cell.

Although ILP is NP-hard, the numbers of variables and constraints used in FR are relatively small after applying CDL, all the optimized grid-based routing results for the target cells can be generated in one second, as shown in column 4 of Table III. Furthermore, the inference time of the EGL model in the LG stage takes only few seconds (as column 5 of Table III). In the VR stage, according to the DRC report, implicit DR constraints will be formulated by CDL and added into the IFR model which can be finished in one second (as column 6 of Table III). Last, as shown in column 7 of Table III, performing DRC takes most of the runtime in our ACLG because design-rule checking (DRC) will be executed iteratively for examining whether the generated cell layout is DRC-free or not. Typically, one round of DRC for one cell takes tens of seconds and there may be multiple runs in VR. As a result, the total runtime may go up to hundreds of seconds[3].

IV. Conclusion

Most of previous works apply explicit encoding on the selected DRs into the routing engine and cannot accommodate the rapid growth on the size and complexity of DRs as the processes continue to advance. To overcome the problems of explicit encoding, the ACLG framework is developed for generating legal layouts of cells automatically and applies two implicit learning techniques (i.e. experience-guidance learning (EGL) and constraint-driven learning (CDL)) to deal with the geometry-related and routing-related DR violations without any prior knowledge of DRs. The experimental results demonstrate that EGL and CDL jointly reduce all DR violations and achieve the 100% violation reduction rate for eight target cells in a finFET-EUV process. Moreover, without any manual effort, the layout of each target cell can be automatically generated in averagely three minutes. Therefore, the proposed ACLG framework is proven effective for solving DR violations and efficient for generating cell layouts for advanced processes.

[3]According to our industrial partner, manually drawing the DRC-free layout from scratch for a cell with the similar complexity as shown in Table II may take more than 25 minutes on average.

Acknowledgment

The authors would like to thank Wei-Min (Ryan) Hsu from MediaTek Inc. for constructive comments on engineering practice of the standard-cell designs. This work was partially supported by Ministry of Science and Technology, Taiwan, R.O.C. under grant no. MOST109-2221-E-009-141.

References

[1] D. Ha et al., "Highly manufacturable 7nm FinFET technology featuring EUV lithography for low power and high performance applications," 2017 Symposium on VLSI Technology, Kyoto, 2017, pp. T68-T69, doi: 10.23919/VLSIT.2017.7998202.

[2] R. S. Ghaida and P. Gupta, "DRE: A Framework for Early Co-Evaluation of Design Rules, Technology Choices, and Layout Methodologies," in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 31, no. 9, pp. 1379-1392, Sept. 2012, doi: 10.1109/TCAD.2012.2192477.

[3] K. Jo, S. Ahn, T. Kim and K. Choi, "Cohesive techniques for cell layout optimization supporting 2D metal-1 routing completion," 2018 23rd Asia and South Pacific Design Automation Conference (ASP-DAC), Jeju, 2018, pp. 500-506, doi: 10.1109/ASPDAC.2018.8297373.

[4] X. Xu, N. Shah, A. Evans, S. Sinha, B. Cline and G. Yeric, "Standard cell library design and optimization methodology for ASAP7 PDK: (Invited paper)," 2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), Irvine, CA, 2017, pp. 999-1004, doi: 10.1109/ICCAD.2017.8203890.

[5] H. Lu et al., "Practical ILP-based routing of standard cells," 2016 Design, Automation & Test in Europe Conference & Exhibition (DATE), Dresden, 2016, pp. 245-248.

[6] N. Ryzhenko and S. Burns, "Standard cell routing via Boolean satisfiability," DAC Design Automation Conference 2012, San Francisco, CA, 2012, pp. 603-612, doi: 10.1145/2228360.2228470.

[7] P. Van Cleeff, S. Hougardy, J. Silvanus and T. Werner, "BonnCell: Automatic Cell Layout in the 7nm Era," in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, doi: 10.1109/TCAD.2019.2962782.

[8] Y. Li et al., "NCTUcell: A DDA-Aware Cell Library Generator for FinFET Structure with Implicitly Adjustable Grid Map," 2019 56th ACM/IEEE Design Automation Conference (DAC), Las Vegas, NV, USA, 2019, pp. 1-6.

[9] D. Park, D. Lee, I. Kang, S. Gao, B. Lin and C. Cheng, "SP&R: Simultaneous Placement and Routing framework for standard cell synthesis in sub-7nm," 2020 25th Asia and South Pacific Design Automation Conference (ASP-DAC), Beijing, China, 2020, pp. 345-350, doi: 10.1109/ASP-DAC47756.2020.9045729.