Integration, the VLSI Journal 70 (2020) 32–42

Contents lists available at ScienceDirect

Integration, the VLSI Journal

journal homepage: www.elsevier.com/locate/vlsi

DSA guiding template assignment with multiple redundant via and dummy via insertion

Xingquan Li [a,*], Bei Yu [b], Jianli Chen [c], Wenxing Zhu [c]

[a] School of Mathematics and Statistics, Minnan Normal University, China
[b] Department of Computer Science and Engineering, The Chinese University of Hong Kong, China
[c] Center for Discrete Mathematics and Theoretical Computer Science, Fuzhou University, China

ARTICLE INFO

Keywords:
Directed self-assembly
Redundant via
Guiding template cost
Unconstrained nonlinear programming
Line search

ABSTRACT

As an emerging manufacture technology, block copolymer directed self-assembly (DSA) is promising for via layer fabrication. Meanwhile, redundant via insertion is considered as an essential step for yield improvement. For better reliability and manufacturability, in this paper, we first concurrently consider DSA guiding template cost assignment with multiple redundant via and dummy via insertion. Firstly, by analyzing the structure property of guiding templates, we propose a building-block based solution expression to discard redundant solutions. Then, honoring the compact solution expression, we construct a conflict graph with dummy via insertion, and then formulate the problem to an integer linear programming (ILP). In addition, to optimize the guiding template cost, we incorporate it into the objective of ILP by introducing vertex weight and edge weight in conflict graph. To make a good trade-off between solution quality and runtime, we relax the ILP to an unconstrained nonlinear programming (UNP). Finally, a line search optimization algorithm is proposed to solve the UNP. Experimental results verify the effectiveness of our new solution expression and the efficiency of our proposed algorithm. Specifically, our guiding template cost optimization method can save 18% total guiding template cost.

1. Introduction

In an integrated circuit (IC) layout, a via provides the connection between two net segments from adjacent metal layers. Due to various reasons such as random defects, cut misalignment, electro migration and thermal/mechanical stress [1,2], a single via may fail partially or completely. Via failure heavily impacts on the functionality and yield of a design [3,4]. Up to now, redundant via (RV) insertion has been considered as an essential step to reduce via failure, and then improves circuit reliability and yield [5,6]. The redundant via insertion technique is that redundant via should be inserted next to every single via [7]. In addition, an inserted redundant via should not cause any circuit short, that is, an inserted redundant via should not overlap with any metal wire from other nets of wires. As the layout shown in Fig. 1(a), we can find all possible redundant via candidates (RVCs). In Fig. 1(b), c_1s are two RVCs of via v_1 and c_2 is a RVC of via v_2. Conventionally, we only need insert one redundant via for a via, and then two possible one redundant via insertion plans are shown in Fig. 1(c) and (d) and for the layout in Fig. 1(a).

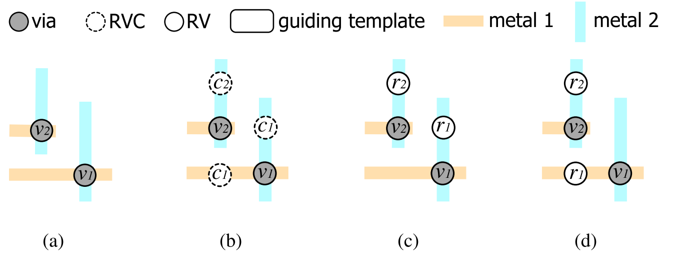

Fig. 1. (a) A layout; (b) Redundant via candidates; (c)(d) Two possible one redundant via insertion plans.

With the size of feature continually shrinking to beyond 7 nm, conventional optical lithography technology patterning becomes more and more expensive. As an emerging VLSI manufacture technology, block copolymer directed self-assembly (DSA) is considered as the most promising for the via layers [8,9]. Furthermore, previous work has made many significant improvements on manufacturing, model-

* Corresponding author.
E-mail addresses: xqli@mnnu.edu.cn (X.), byu@cse.cuhk.edu.hk (B.), jlchen@fzu.edu.cn (J.), wxzhu@fzu.edu.cn (W.).

https://doi.org/10.1016/j.vlsi.2019.09.011
Received 14 June 2019; Received in revised form 26 August 2019; Accepted 24 September 2019
Available online 10 October 2019

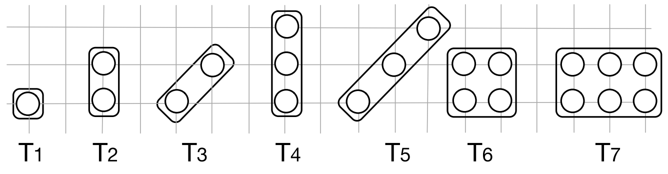

Fig. 2. Seven useable types of guiding templates.

ing, simulation and graphoepitaxy of DSA [10,11]. These improvements enable DSA technology to pattern vias. In DSA, block copolymers with right proportion would form cylinders, and the remainder material can be used to fabricate contacts/vias after removing cylinders. To generate irregularly distributed vias using DSA, guiding templates including vias are required [12,13]. Since irregular guiding template has a higher probability of generating overlay error, to guarantee the overlay accuracy, we only use some regular guiding templates with few holes. In this paper, we follow [14] designing seven types of useable guiding templates T_1, T_2, … , T_7 as shown in Fig. 2. These guiding templates are manufactured by the conventional optical lithography, and thus the resolution is limited by the pattern pitch. The spacing between neighboring guiding templates should not be less than the optical resolution limit spacing d_s. Thus, only one guiding template between two close guiding templates can be patterned. To form the shape of useable guiding templates, adjacent vias and redundant vias may be gathered together and put into a matched guiding template [15–17]. Thus, we need to decide the assignment of guiding templates such that more vias and redundant vias can be surrounded by guiding templates.

Figs. 3(a), (b) and (c) show three different guiding template assignments for the redundant via insertion plan in Fig. 1(c). In Fig. 3(a), via v_1 and redundant via r_1 are assigned to a 1 × 2 hole template t_1, and via v_2 and a redundant via r_2 are guided by another 1 × 2 hole template t_2. However, since t_1 and t_2 are too close, only one of them can be patterned (suppose t_1). For the result in Fig. 3(a), only a via can be manufactured and a redundant via can be inserted. The same as for the result in Fig. 3(b). Similarly, for the result in Fig. 3(c), two vias can be manufactured but two redundant via cannot be inserted.

In the traditional design process, the redundant via insertion and the manufacture of via layers are handled in two independent stages. Fang et al. [14] first concurrently considered the redundant via insertion and DSA guiding template assignment problem. For this concurrent consideration, both the number of insertable vias (insertion rate, IR) and the number of manufacturable vias (manufacture rate, MR) are increased. Recently, several techniques are presented to improve both of the insertion rate and the manufacture rate. The mainstream techniques in previous works [18–21] can be summarized in the following two types: increasing resolution space by multiple patterning [19,20]; improving the degree of freedom of redundant vias and guiding templates [18,21]. For the second way, Fang et al. [21] investigated the redundant via insertion and DSA guiding template assignment problem with wire bending. By local perturbing some metal wires, it inserts redundant vias at the cost of increasing the wirelength. But in the advanced 1-D metal layer design, wire bending are unwarrantable. To avoid this issue, Hung et al. [18] studied the problem with dummy via insertion, in which some dummy vias are inserted for assisting formation of guiding templates. In a circuit, dummy via does not connect to any wire, which is only used for filling the guiding template. For the redundant via insertion plan in Fig. 1(c), if we insert a dummy via (DV) as in Fig. 3(d), then the via v_2 and redundant vias r_2 and r_1 can be guided by a regular 2 × 2 template t_2, but via v_1 still cannot be patterned due to the small space between template t_1 and t_2.

Traditionally, designers only insert one redundant via for a single via. To further improve the insertion rate and the manufacture rate, in this paper, we consider another technology, namely multiple redundant vias insertion. As shown in Fig. 3(d), v_1 cannot be patterned due to space limitation. But if v_1 can be included by a guiding template with other vias or redundant vias, then v_1 also can be patterned properly. In Fig. 4(a), a guiding template t_1 include two vias, two redundant vias and a dummy via. Regretfully, this guiding template t_1 is irregular and is not our useable type of guiding template. However, if we insert another redundant via r_1 of via v_1 in lower-left, then the formed distribution of holes match a T_7 guiding template as Fig. 4(b). Thus, multiple redundant vias insertion can improve the insertion rate and manufacture rate.

After using multiple redundant vias insertion and dummy via insertion, layouts become more free for inserting redundant vias and using multi-hole guiding templates. And then the insertion rate and manufacture rate can be improved. In Ref. [18], the authors generated all guiding template candidates for all the redundant via candidates, dummy via candidates, and immediate neighbor vias. The generated guiding template candidates are utilized to express solution space, which is extremely large. And then, to solve the problem, the authors of [18] proposed an ILP formulation. As we know, solving ILP can obtain the exact result, but it is very time consuming for the large scale and dense circuit layouts since the NP complexity of ILP. The experimental comparisons in Section 5 verify this. Hence, it is important to derive a more compact solution space and a fast solving method for the DSA guiding template assignment with multiple redundant via and dummy via insertion problem.

In addition, grouping more than one contacts in a multi-hole guiding template may introduce overlays. For different guiding templates with different shapes or sizes, the overlays are different. Specifically, complex guiding templates with more holes may introduce large overlays and the contained vias may not be patterned correctly. Hence, to achieve a better guiding template assignment result with less total overlays, the costs of guiding templates should be considered during guiding template assignment. An example is shown in Fig. 5. Given a layout with two vias as Fig. 5(a), we can find two guiding template assignment results as Figs. 5(b) and (c) and , in which two vias can be patterned and a redundant via can be inserted. There is a T_4 guiding template in Fig. 5(b), and there are a T_1 and a T_2 in Fig. 5(c). Suppose the costs of guiding templates T_1, T_2 and T_4 are $w_{T_1} = 0$, $w_{T_2} = 1$ and $w_{T_4} = 3$, respectively. Obviously, the guiding template assignment result in Fig. 5(c) is better than in Fig. 5(b).

In this paper, we consider DSA guiding template assignment with multiple redundant via and dummy via insertion. Our main contributions are summarized as follows.

- We first introduce multiple redundant via technique to improve the insertion rate and manufacture rate.
- We first optimize the guiding template cost, and achieve guiding template assignment results with less overlays.
- We first prove the NP-complexity of the DSA guiding template assignment and redundant via insertion problem.
- We introduce a building-block based manner instead of guiding template candidate to compactly express solution. With the help of building-blocks, we model the DSA guiding template assignment with redundant via and dummy via insertion problem to a new ILP formulation based on a conflict graph.
- With a sigmoid function, we relax the ILP to an UNP to make a good trade-off between solution quality and runtime. We develop a line search optimization algorithm to solve the UNP, and prove the local converegence of the proposed algorithm.

The rest of this paper is organized as follows. In Section 2, we introduce the related concepts and the problem formulation. In Section 3, we discuss the proposed graph model. In Section 4, we detail our IP formulation, local optimal algorithm and guiding template cost optimization method for the problem. In Section 5, we list experimental results, followed by conclusion in Section 6.

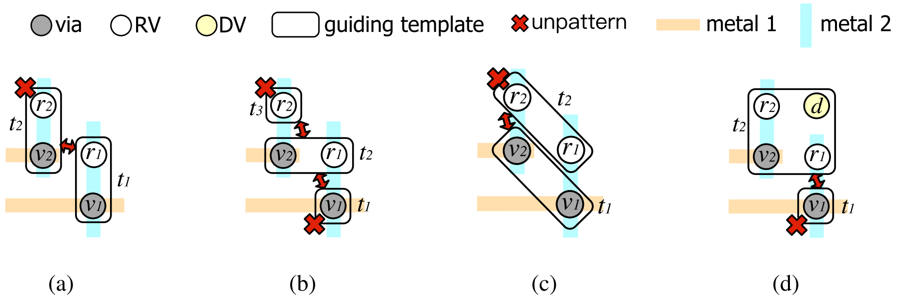

Fig. 3. (a)(b)(c) Three guiding template assignments for redundant via insertion plan in Fig. 1(c); (d) A result with dummy via insertion.

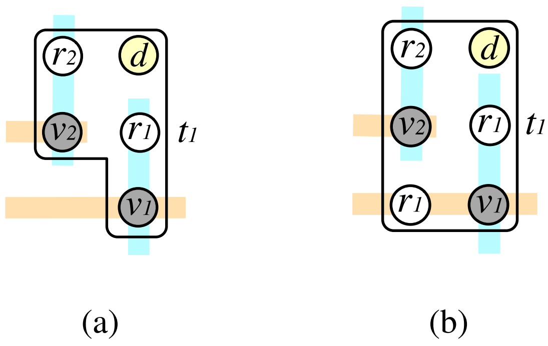

Fig. 4. (a) An irregular guiding template; (b) A result with multiple redundant via and dummy via insertion.

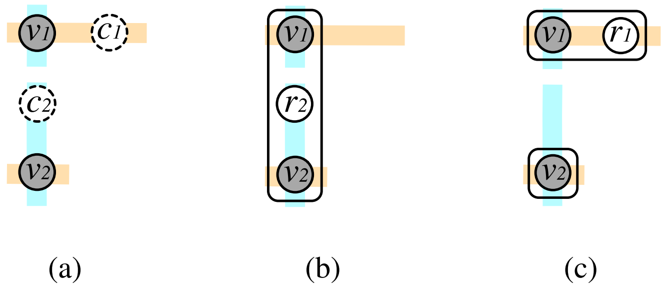

Fig. 5. (a) A layout with two redundant via candidates; (b)(c) Two possible guiding template assignment results.

2. Preliminaries

In this section, we first briefly introduce the details of multiple redundant via and dummy via insertion, respectively. Then, the problem formulation and solution flow will be presented.

2.1. Multiple redundant via insertion

In this paper, we consider the multiple redundant via insertion on a grid graph. Suppose the grid coordinate of a single via v_i is (x_i, y_i). For the neighbor four grid coordinates $(x_i - 1, y_i)$, $(x_i + 1, y_i)$, $(x_i, y_i - 1)$ and $(x_i, y_i + 1)$, a grid coordinate is called a redundant via candidate (RVC) of v_i if its position is not occupied by other vias or metal wires from other nets.

Conventionally, the objective of redundant via insertion problem is to insert a redundant via for a via. To match more possible useable guiding template shape, in this paper, a via is allowed to insert multiple redundant vias (one or more than one redundant vias). As shown in Fig. 4(b), to match a T_7 guiding template, two redundant vias r_1s are inserted for via v_1. Naturally, the insertion of two or more redundant vias for a via should satisfy the following two conditions: i) the insertion can make up a multi-hole (not less than three holes) guiding template with other vias or redundant vias; ii) it can improve the insertion rate or manufacture rate.

2.2. Dummy via insertion

As the technique of multiple redundant via, another effective trick is adding some dummy vias (DV), such that the structure of vias matches a special useable guiding template. As the effect of dummy via d in Fig. 3(d). In a circuit, dummy via does not connect to any wire, which is only used for filling the guiding template. The same as multiple redundant via, the insertion of dummy vias also should satisfy the following two conditions: i) the insertion can make up a multi-hole (not less than three holes) guiding template with other vias or redundant vias; ii) it can improve the insertion rate or manufacture rate. After finding the possible redundant via candidates (RVC), we should find all potential guiding template assignments for every via. If these needed dummy vias on the empty grid points satisfy the above two conditions, then these empty grid points are marked as dummy via candidates (DVC).

2.3. Problem and framework

The problem aims at inserting at least a redundant via for every via, and manufacturing all vias and their redundant vias by the DSA technique with the help of dummy via insertion. The redundant via insertion rate and the manufacture rate are considered as evaluation indicators in Refs. [14,19]. The insertion rate (IR) is defined as the ratio of the number of vias with redundant vias to the number of vias. And the manufacture rate (MR) is the ratio of the number of manufacturable vias to the number of vias. The DSA guiding template assignment with multiple redundant via and dummy via insertion (DMRD) problem is formulated as follows:

Problem 1. **[DMRD].** *Given a post-routing* via *layers layout, the usable types of guiding templates, and the optical resolution limit spacing d_s, insert at least a redundant* via *for every* via*, assign guiding templates for vias, redundant vias and dummy vias, such that: i) the inserted redundant vias are legal; ii) the spacing between neighboring guiding templates should not be less than d_s. The objective is maximizing $MR + \beta \cdot IR$, where β is a weighting parameter.*

To show the complexity of the DMRD problem, we first simplify the DMRD problem by restricting the useable types of guiding templates to only T_1. Then the simplified-DMRD problem can be formulated as follows: Given a layout with vias and redundant via candidates (they can be pre-detected), in which every via or redundant via candidate is included into a T_1 guiding template, we need find a subset V_1 of all T_1s such that the distance between every two T_1s in V_1 is larger than d_s. The target of simplified-DMRD problem is to maximize the size of V_1, if $\beta = 1$. Obviously, the simplified-DMRD problem is a special case of the DMRD problem. On the other hand, we introduce the unit disk

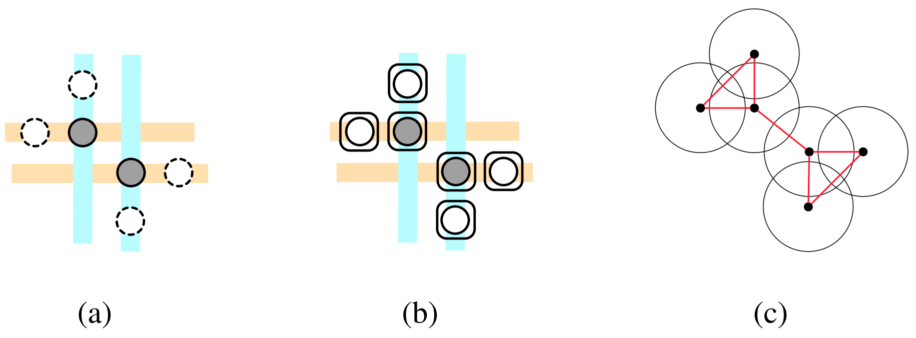

Fig. 6. (a) Layout after detecting redundant via candidates; (b) The simplified-DMRD problem; (c) Unit disk graph.

maximum independent set (UDMIS) problem, which is solving the maximum independent set on unit disk graph. The unit disk graph is a graph, where vertices corresponds to the equal-sized circles in the plane, and an edge appears between two vertices when the corresponding circles intersect [22].

Figs. 6(a) and (b) show an example of the simplified-DMRD problem. Fig. 6(c) is a unit disk graph.

Lemma 1. *The UDMIS problem is equivalent to the simplified-DMRD problem.*

Proof. *The UDMIS problem can be further described as follows: Given a unit disk graph, we need find a subset V_c of centers of all circles, such that the distance of every two centers of circles in V_c is larger than the diameter of unit circle. The target of UDMIS problem is to maximize the size of V_c. If we set the diameter of unit circle as d_s, then the UDMIS problem is equivalent to the simplified-DMRD problem.*

Since the UDMIS problem is NP-hard [22], the simplified-DMRD problem is a special case of the DMRD problem, we have following theorem.

Theorem 1. *DMRD problem is NP-hard.*

Furthermore, to achieve a better patterned result, we integrate the DSA guiding template cost into our objective, i.e., we need to concurrently minimize the total guiding template cost *Tcost*. Then the redundant via insertion and DSA guiding template (cost-aware) assignment with multiple redundant via and dummy via insertion (DMRD-cost) problem is formulated as follows:

Problem 2. [DMRD-cost]. *Given a post-routing* via *layers layout, the usable types of guiding templates, and the optical resolution limit spacing d_s, insert at least a redundant* via *for every* via, *assign guiding templates for vias, redundant vias and dummy vias, such that: i) the inserted redundant vias are legal; ii) the spacing between neighboring guiding templates should not be less than d_s. The objectives are maximizing $MR + \beta \cdot IR$ and minimizing Tcost.*

In this work, we propose a local optimal method to concurrently optimize the redundant via insertion and guiding template assignment. Fig. 7 shows our solution flow, which is composed of preprocessing and solver. First, in the preprocessing stage, we first find all redundant and dummy via candidates. Based on these candidates, we

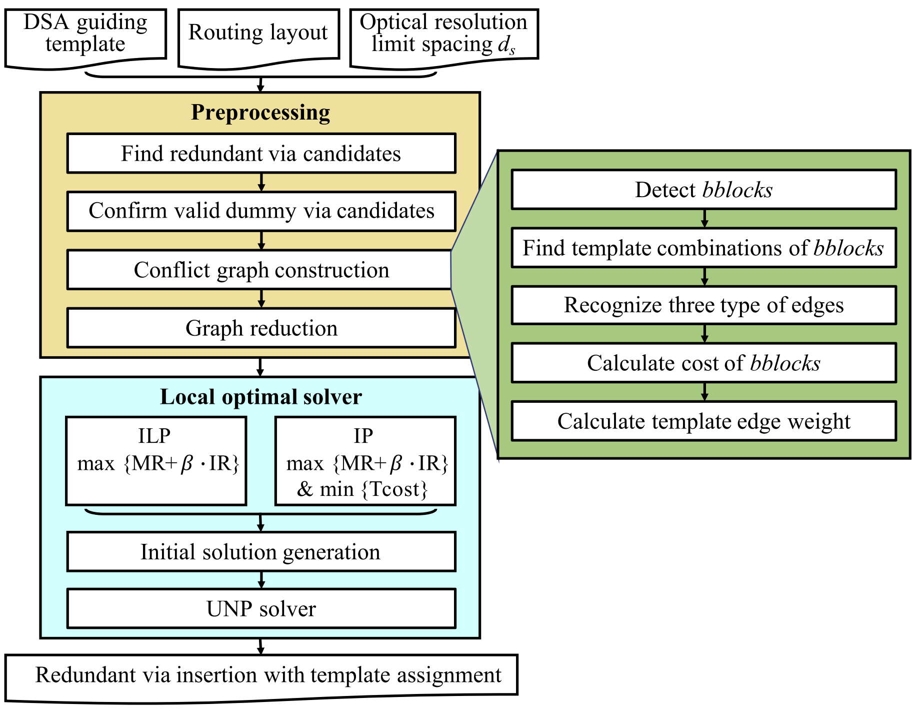
DSA guiding template
Routing layout
Optical resolution limit spacing d_s
Preprocessing
Find redundant via candidates
Confirm valid dummy via candidates
Conflict graph construction
Graph reduction
Detect *bblocks*
Find template combinations of *bblocks*
Recognize three type of edges
Calculate cost of *bblocks*
Calculate template edge weight
Local optimal solver
ILP max {MR+$\beta \cdot$IR}
IP max {MR+$\beta \cdot$IR} & min {Tcost}
Initial solution generation
UNP solver
Redundant via insertion with template assignment

Fig. 7. Solution flow.

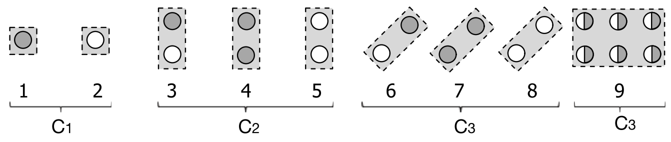

Fig. 8. Building-blocks.

detect all building-blocks, and further construct a conflict graph on building-blocks. Then, some vertices deletion techniques are introduced to reduce the size of conflict graph. Second, in the local optimal solver, we first formulate the DMRD problem into two integer programmings (IPs): IP without guiding template cost consideration, and IP with guiding template cost consideration. Then, we design an unconstrained nonlinear programming (UNP) algorithm to solve our IPs. Before that, we propose a method to generate a good initial solution for our UNP method.

3. Conflict graph construction

3.1. Building-block

After finding all possible redundant via candidates and dummy via candidates, previous work directly checks all the possible guiding template assignments. However, the solution space based on guiding template assignments would be extremely large. And then, solving the DMRD problem with extremely large number of guiding template assignments is time consuming.

To obtain a compact solution expression, we introduce a concept of building-block (*bblock*). A *bblock* is composed of some vias and some candidates, and *bblocks* can be used to compose various types of guiding templates. Nine types of *bblocks* are shown in Fig. 8, where *bblock*_1 includes a via; *bblock*_2 includes a redundant via; *bblock*_3 includes a via and a redundant via; *bblock*_4 includes two vias; *bblock*_5 includes two redundant vias; *bblock*_6 includes a via and a redundant via in diagonal; *bblock*_7 includes two vias in diagonal; *bblock*_8 includes two redundant vias in diagonal; and *bblock*_9 includes six vias or redundant vias, which can be covered by a six-hole guiding template. These types of *bblocks* compose a dictionary. *bblock*_1 and *bblock*_2 are grouped as C_1; *bblock*_3, *bblock*_4 and *bblock*_5 are grouped as C_2; *bblock*_6, *bblock*_7 and *bblock*_8 are grouped as C_3, *bblock*_9 is grouped as C_4.

3.2. Conflict graph

Then the seven types of useable guiding templates in Fig. 2 can be formed by grouping some *bblocks* in the dictionary, as shown in Fig. 9. Here we only list the vertical cases and skip the horizontal cases due to similarity.

A dummy via candidate (DVC) must belong to a guiding template. At the DVC finding stage, we can easily find out which guiding template a DVC belongs to. Given a result of finding redundant via candidates as in Fig. 10(b), we can identify all possible *bblocks* as in Fig. 10(c), and these *bblocks* are regarded as vertices in the conflict graph. Based on these vertices, we construct a conflict graph [20], as shown in Fig. 10(d). There are three types of edges in conflict graph:

Definition 1. [Overlap Edge [20]] . *There exists an overlap edge e_{ij} between bblock s i and j if they overlap each other. Let E_O be the set of overlap edges.*

Definition 2. [Conflict Edge [20]] . *There is a conflict edge e_{ij} between bblock s i and j if the distance between them is not larger than d_s and $e_{ij} \notin E_O$. Let E_C be the set of conflict edges.*

Definition 3. [Template Edge [20]] . *For two bblock s i and j, suppose that at least one of them is not S_1. If i and j can be assigned simultaneously to a guiding template without any design error, and between i and j there exists a conflict edge $e_{ij} \in E_C$, then e_{ij} is also called a template edge between i and j. Let E_T be the set of template edges. Obviously, $E_T \subseteq E_C$.*

Then, a conflict graph $CG(V, E)$ is constructed.

Definition 4. [Conflict Graph CG **[20]] .** *The conflict graph is an undirected graph $CG(V, E)$, where vertex $v \in V$denotes a bblock, $e_{ij} \in E$ is an edge and $E = (E_C - E_T) \cup E_O$. E_C, E_T and E_O are the sets of conflict edges, template edges and overlap edges, respectively.*

In Fig. 10(d), the black edges are the overlap edges, the red edges are the conflict edges, and the green dotted edges are the template edges. From Fig. 10(b), we know *bblocks e* and *d* are overlapped with each other at r_1, hence there is an overlap edge between them as in Fig. 10(d). The distance between *bblocks a* and *e* is not larger than d_s, hence there is a conflict edge between them. Since *bblocks b* and *d* can be assigned to a 3×1 hole guiding template as in Fig. 10(d), there is a template edge between *b* and *d*.

According to our conflict graph construction process, the constructed conflict graph may be very dense. Specially, there may exist some vertices with local full-degree, i.e., these vertices connected to all vertices (except themself) by conflict edges or overlap edges in a local subgraph. For example, vertex *e* in Fig. 10(d) is a local full-degree vertex since it connects to all other vertices by conflict edges or overlap edges. In this part, we propose a method to reduce the size of conflict graph by removing these local full-degree vertices. Firstly, we perform our initial solution generation algorithm in Algorithm 1, then we obtain a solution with greedy best value. We compare the greedy best value with the weights of all local full-degree vertices one by one. If the weight of local full-degree vertex *v* is not greater than the greedy best value, then vertex *v* and its connected conflict edges and overlap edges will be removed from the conflict graph. After performing the graph reduction, the conflict graph in Fig. 10(d) is simplified as Fig. 10(e).

3.3. Guiding template cost, building-block cost, and template edge weight

Suppose the costs of seven guiding templates $T_1, T_2, \ldots, T_7$ are w_{T_1}, w_{T_2}, …, w_{T_7}, respectively. And suppose the costs of four groups of building-blocks $C_1, C_2, C_3 and\ C_4$ are w_{C_1}, w_{C_2}, w_{C_3}, and w_{C_4}, respectively. Four groups of building-blocks can be used to generate seven guiding templates by some template edges. From Figs. 4 and 5, it can be seen that: Guiding template T_1 can be composed of a C_1; Guiding

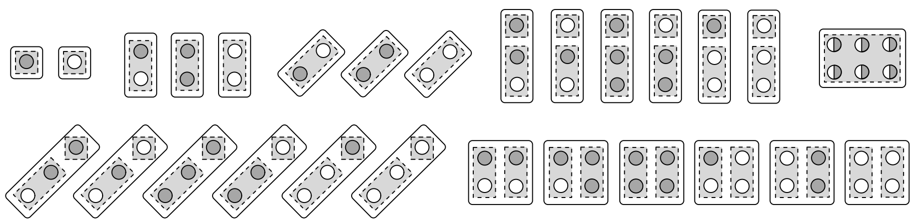

Fig. 9. All possible combinations of *bblocks* to form the seven types of guiding templates.

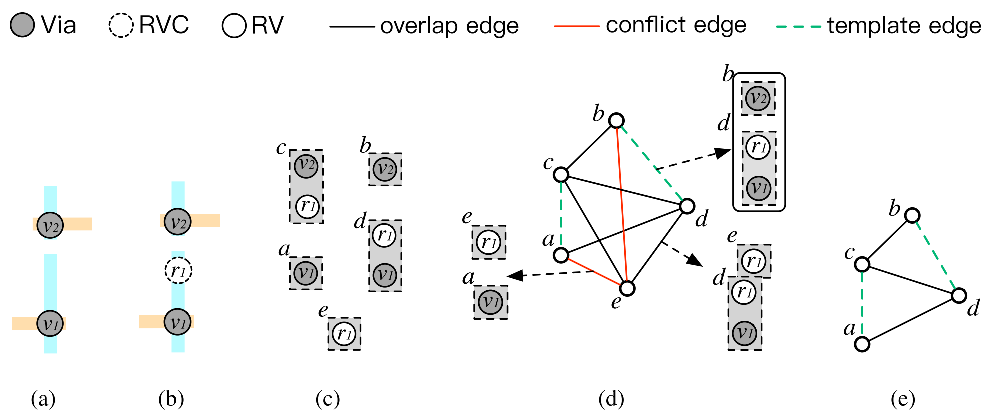

Fig. 10. (a) A layout; (b) Redundant via candidates; (c) All *bblocks* of the layout in Fig. 10 (a); (d) Conflict graph; (e) Reduced conflict graph.

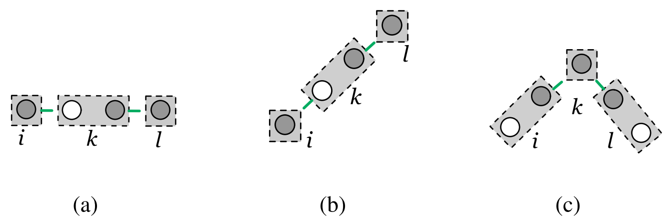

Fig. 11. Three kinds of conflict structures.

template T_2 can be composed of a C_2; Guiding template T_3 can be composed of a C_3; Guiding template T_4 can be composed of a C_1 and a C_2; Guiding template T_5 can be composed of a C_1 and a C_3; Guiding template T_6 can be composed of two C_2s; Guiding template T_7 can be composed of a C_4. Guiding templates T_1, T_2, T_3 and T_7 compose of only one building-block, guiding templates T_3, T_4 and T_5 compose of two building-blocks and a template edge. Furthermore, we divide the set of template edges according to the types of guiding templates. That is $E_T = E_{T_4} \cup E_{T_5} \cup E_{T_6}$, where E_{T_4} is the set of template edges in guiding template T_4, E_{T_5} is the set of template edges in guiding template T_5, and E_{T_6} is the set of template edges in guiding template T_6. Suppose the weight of template edges in E_{T_l} as $w^e_{T_l}$, $(l = 4, 5, 6)$, then we have follow relationships:

$$\begin{cases} w_{T_1} = w_{C_1}; \\ w_{T_2} = w_{C_2}; \\ w_{T_3} = w_{C_3}; \\ w_{T_4} = w_{C_1} + w_{C_2} + w^e_{T_4}; \\ w_{T_5} = w_{C_1} + w_{C_3} + w^e_{T_5}; \\ w_{T_6} = 2w_{C_2} + w^e_{T_6}; \\ w_{T_7} = w_{C_7}. \end{cases} \quad (1)$$

And then, we have

$$\begin{cases} w_{C_1} = w_{T_1}; \\ w_{C_2} = w_{T_2}; \\ w_{C_3} = w_{T_3}; \\ w_{C_4} = w_{T_7}; \\ w^e_{T_4} = w_{T_4} - w_{T_1} - w_{T_2}; \\ w^e_{T_5} = w_{T_5} - w_{T_1} - w_{T_3}; \\ w^e_{T_6} = w_{T_6} - 2w_{T_2}. \end{cases} \quad (2)$$

That is, given the guiding template costs of seven useable guiding templates, we can obtain the costs of four groups of building-blocks and the weight of template edges.

4. Algorithms

4.1. ILP formulation

4.1.1. Constraints

In conflict graph, if two *bblocks* i and j are overlapped with each other, i.e., $e_{ij} \in E_O$, then only one of the them can be patterned. In addition, if two *bblocks* are within the optical resolution limit spacing d_s, i.e., $e_{ij} \in E_C$, then only one of them can be patterned due to limitation of resolution, unless the two *bblocks* are assigned into the same guiding template, i.e., $e_{ij} \notin E_T$.

If two *bblocks* i and j are connected by a template edge, then they may be assigned to the same guiding template, but not necessarily. Specially, for the structure shown in Fig. 11(a), *bblocks* i and l are connected to k by two template edges, but i, k and l cannot be simultaneously assigned to a same guiding template, since we do not have a guiding template with four holes aligned in a line (same as the structures in Fig. 11(b) and (c)). We call these unordered triplets (i, k, l) as conflict structures, and denote CS as the set of conflict structures, whose formal definition are described as follow:

Definition 5. [Conflict Structure]. *The conflict structure is a structure composed of three bblock s i, k and l, in which e_{ik} and e_{kl} are template edges and there does not exist any edge between i and l.*

4.1.2. Objectives

The main objective of DMRD problem is intend to maximize $MR + \beta \cdot IR$, i.e., maximizing the weighted sum of the number of manufacturable vias and the number of inserted redundant vias. Suppose the weights of a via and a redundant via is 1 and β, respectively. Since different *bblocks* include different vias and redundant vias, we jointly consider MR and IR by assigning weight w_i to every *bblock* i as

$$w_i = N_v + \beta \cdot N_r, \quad (3)$$

where N_v and N_r are the numbers of included vias and redundant vias by *bblock* i, respectively. Let W be the set of weights, then the conflict graph $CG(V, E)$ is weighted and written as $CG(V, E, W)$.

Let binary variable $x_i = 1$ denote that building-block i is selected. Then, $\sum_{i \in V} w_i x_i$ denotes the main objective. We formulate the DMRD problem as following ILP.

$$\max_{\mathbf{x}} \sum_{i \in V} w_i x_i \quad (4)$$

$$\text{s.t.} \quad x_i + x_j \le 1, \quad \forall e_{ij} \in E; \tag{4a}$$

$$x_i + x_k + x_l \le 2, \quad \forall (i,k,l) \in CS; \tag{4b}$$

$$x_i \in \{0,1\}, \quad \forall i \in V. \tag{4c}$$

In above ILP, Constraint 4(a) indicates that, if there exists $e_{ij} \in E_O$ or $e_{ij} \in E_C - E_T$ between vertices i and j, then at most one of them can be patterned; Constraint 4(b) ensures that, if i, k and l compose an conflict structure, then at most two of them can be patterned.

4.2. A local optimal algorithm

It is time consuming to solve the ILP by commercial solver for a large scale layout. In this subsection, we develop a fast algorithm to obtain a local optimal solution of DMRD problem.

Firstly, the ILP formulation of Problem (4) is equivalent to:

$$\max_{\mathbf{x}} \sum_{i \in V} w_i x_i \tag{5}$$

$$\text{s.t.} \quad x_i x_j = 0, \quad \forall e_{ij} \in E; \tag{5a}$$

$$x_i x_k x_l = 0, \quad \forall (i,k,l) \in CS; \tag{5b}$$

$$x_i \in \{0,1\}, \quad \forall i \in V, \tag{5c}$$

where $\mathbf{w} = (w_1, w_2, \dots, w_n)^\top \in \mathbb{R}^n$, $\mathbf{x} = (x_1, x_2, \dots, x_n)^\top \in \{0,1\}^n$, $n = |V|$. Since Constraints (5a) and (5b) are equality, they can be directly incorporated in the objective function. That is, Problem (5) can be further rewritten as following integer nonlinear programming:

$$\max_{\mathbf{x}} \sum_{i \in V} \{w_i x_i \prod_{\substack{j \in V \\ e_{ij} \in E}} (1 - x_j) \prod_{\substack{k,l \in V \\ (i,k,l) \in CS}} (1 - x_k x_l)\} \tag{6}$$

$$\text{s.t.} \quad x_i \in \{0,1\}, \quad \forall i \in V. \tag{6a}$$

In (6), $e_{ij} \in E$ and $(i,k,l) \in CS$ are used to describe the relationship among vertices. Let $\mathbf{B} = (B_{ij}) \in \{0,1\}^{n \times n}$ be the adjacency matrix of the conflict graph CG. If $e_{ij} \in E$, then $B_{ij} = 1$ and $(1 - x_j)^{B_{ij}} = (1 - x_j)$; and if $e_{ij} \notin E$, then $B_{ij} = 0$ and $(1 - x_j)^{B_{ij}} = 1$. Moreover, we use $\mathbf{C} = (C_{ikl}) \in \{0,1\}^{n \times n \times n}$ to represent the conflict structures in layout. If $(i,k,l) \in CS$, then $C_{ikl} = 1$ and $(1 - x_k x_l)^{C_{ikl}} = (1 - x_k x_l)$, otherwise $C_{ikl} = 0$ and then $(1 - x_k x_l)^{C_{ikl}} = 1$. The objective of Problem (6) can be more conveniently written using adjacent matrix and tensor of CG as Problem (7).

$$\max_{\mathbf{x}} \sum_{i \in V} \{w_i x_i \prod_{j \in V} (1 - x_j)^{B_{ij}} \prod_{k,l \in V} (1 - x_k x_l)^{C_{ikl}}\} \tag{7}$$

$$\text{s.t.} \quad x_i \in \{0,1\}, \quad \forall i \in V. \tag{7a}$$

Problem (7) is equivalent to Problem (4), and still falls to the catogory of discrete formulation. To design a more efficient solution, we further relax this problem into a continuous domain. First, we introduce an auxiliary vector $\mathbf{y} = (y_i) \in \mathbb{R}^n$, and approximate the constraint $x_i \in \{0,1\}, \forall i \in V$ with the sigmoid function

$$x_i \approx \sigma(y_i) = (1 + e^{-\gamma y_i})^{-1}. \tag{8}$$

Above sigmoid function is used to approximate function

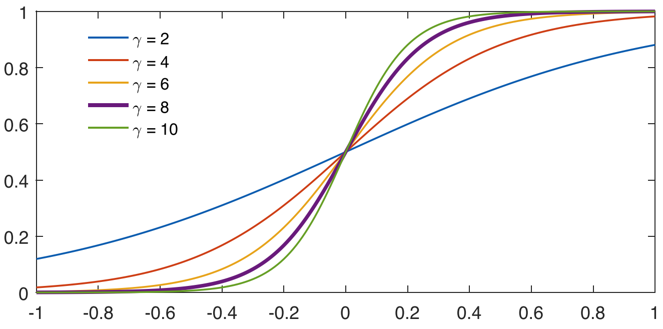

Fig. 12. Sigmoid function $\sigma(y_i)$ on different γ.

$$x_i = \begin{cases} 0, & y_i \le 0; \\ 1, & y_i > 0, \end{cases} \tag{9}$$

where γ is set to 8 in this paper for a sharper sigmoid function. The detailed curves of sigmoid function with different γ are plotted in Fig. 12, where γ is set to 8 in this paper for a sharper sigmoid function.

Then Problem (7) is approximated as

$$\max_{\mathbf{y}} f(\mathbf{y}) = \sum_{i \in V} \{w_i \sigma(y_i) \prod_{j \in V} (1 - \sigma(y_j))^{B_{ij}} \prod_{k,l \in V} (1 - \sigma(y_k)\sigma(y_l))^{C_{ikl}}\}. \tag{10}$$

If we obtain a solution $\mathbf{y}^*$ of Problem (10), then the final solution $\mathbf{x}^*$ is obtained by rounding the sigmoid function value $\sigma(y_i^*)$ to the nearest integer, $\forall i \in V$. Problem (10) is an unconstrained nonlinear programming (UNP). Let

$$g_i(\mathbf{y}) = \sigma(y_i) \prod_{j \in V} (1 - \sigma(y_j))^{B_{ij}} \prod_{k,l \in V} (1 - \sigma(y_k)\sigma(y_l))^{C_{ikl}} \tag{11}$$

and $g_i = g_i(\mathbf{y})$, then the objective of Problem (10) is

$$f(\mathbf{y}) = \sum_{i \in V} w_i g_i.$$

We aim at finding a maximal solution $\mathbf{y}^* \in \mathbb{R}^n$ of Problem (10). At each iteration t, the solution is updated by the following gradient direction of $f(\mathbf{y})$:

$$\mathbf{y}^{(t+1)} = \mathbf{y}^{(t)} + \alpha \nabla f(\mathbf{y}^{(t)}), \tag{12}$$

where α is the step length, which is obtained by the Wolfe-Powell inexact line search method in Ref. [23]. Besides, $[\nabla f(\mathbf{y}^{(t)})]_i = \partial f(\mathbf{y}^{(t)}) / \partial y_i$ is calculated by

$$[\nabla f(\mathbf{y}^{(t)})]_i = \gamma w_i g_i^{(t)} \{(1 - \sigma(y_i^{(t)})) - \sum_j B_{ij} \sigma(y_j^{(t)}) - \sum_k \sum_l C_{ikl} \frac{\sigma(y_k^{(t)})(1 - \sigma(y_k^{(t)}))\sigma(y_l^{(t)})}{1 - \sigma(y_k^{(t)})\sigma(y_l^{(t)})}\}, \tag{13}$$

where $g_i^{(t)} = g_i(\mathbf{y}^{(t)})$. It can be shown that first order dynamic in Equation (12) increases $f(\mathbf{y}^{(t)})$ at every iteration t, and will converge to a local optimum.

However, since the objective function of Problem (10) is highly nonlinear and non-concave, the above iteration is highly dependent on the initial solution $\mathbf{y}^{(0)}$ and may converge to an undesirable local optimum. Hence, in order to obtain a better solution, the iteration would be better starting from a good initial solution $\mathbf{y}^{(0)}$. We propose an $\mathcal{O}(|V|)$ complexity algorithm to obtain a desirable initial solution, as detailed in Algorithm 1.

Table 1
Comparison of four methods for DMRD problem.

Benchmarks	#V	TCAD/17 [14]			ASPDAC/17 [18]			Our-ILP			Our-UNP		
		MR(%)	IR(%)	CPU(s)	MR(%)	IR(%)	CPU(s)	MR(%)	IR(%)	CPU(s)	MR(%)	IR(%)	CPU(s)
Benchmarks	#V	MR(%)	IR(%)	time(s)	MR(%)	IR(%)	time(s)	MR(%)	IR(%)	time(s)	MR(%)	IR(%)	time(s)
struct	12551	99.86	99.42	30.00	99.96	99.79	40.47	99.96	99.81	19.72	99.96	99.81	14.78
primary 1	8764	99.80	99.21	28.00	99.92	99.33	29.14	99.88	99.64	21.08	99.87	99.55	15.42
primary2	32684	99.54	98.97	72.00	99.92	99.49	111.22	99.81	99.49	58.83	99.81	99.49	37.10
s5378	8649	81.10	61.78	11.00	96.41	75.27	34.41	97.35	80.89	9.61	97.34	80.49	0.36
s9234	6874	80.07	59.54	9.00	96.21	74.51	38.63	97.23	80.85	7.26	97.14	80.76	0.33
s13207	18780	84.35	66.93	23.00	97.13	79.28	99.96	97.99	83.61	21.31	97.89	83.44	1.39
s15850	22694	82.70	64.41	28.00	96.63	77.35	121.94	97.71	82.65	22.21	97.77	82.22	1.70
s38417	54225	84.00	65.71	65.00	96.83	78.13	320.35	96.92	83.34	67.71	96.92	83.23	16.42
s38584	74155	81.53	63.01	88.00	96.37	76.83	416.11	97.84	82.54	99.72	97.21	82.21	15.37
dma	34697	97.85	95.29	55.00	99.61	97.49	208.75	99.50	97.62	31.80	99.48	97.61	4.34
dsp 1	30317	99.05	97.57	53.00	99.74	98.45	176.42	99.75	98.84	36.72	99.73	98.81	6.97
dsp 2	31301	98.50	96.68	52.00	99.76	98.74	179.37	99.68	98.95	33.99	99.72	98.91	6.47
risc1	43858	98.77	96.93	75.00	99.70	98.04	216.61	99.74	98.56	51.69	99.74	98.56	12.78
risc2	44385	98.79	96.91	77.00	99.70	97.98	229.22	99.69	98.45	50.86	99.69	98.45	13.07
Avg.	30281	91.85	83.03	47.57	98.42	89.33	158.76	98.79	91.80	38.04	98.73	91.68	10.46
Ratio		0.93	0.91	4.55	1.00	0.97	15.17	1.00	1.00	3.63	1.00	1.00	1.00

```
Algorithm 1 Initial Solution Generation
Input: A connected component of $CG(V, E, W)$;
Output: Initial solution $\mathbf{x}^{(0)}$ of ILP (4);
1:  repeat
2:    $S \leftarrow \{k \mid k \in \mathrm{argmin}_{l\in V} w_n(l)\}$;
3:    repeat
4:      $\forall k \in S$, compute $w_s(k)$;
5:      $x_i^{(0)} \leftarrow 1$, where $i = \mathrm{argmin}_{k\in S} w_s(k)$;
6:      forevery $j$ in $V$ with $e_{ji} \in E$ do
7:        $x_j^{(0)} \leftarrow 0$, and $V \leftarrow V - \{j\}$;
8:        if $j \in S, S \leftarrow S - \{j\}$;
9:      end for
10:     $S \leftarrow S - \{i\}$, and $V \leftarrow V - \{i\}$;
11:   until $S = \emptyset$
12: until $V = \emptyset$
```

In line 2 of Algorithm 1, $w_n(l)$ is the number of vias and redundant vias covered by *bblock l*. In line 4, the selection weight $w_s(k)$ of *bblock k* is calculated by

$$w_s(k) = d_c(k) - d_t(k), \tag{14}$$

where $d_c(k)$ is the number of conflict edges incident to *bblock k*, and $d_t(k)$ is the number of template edges incident to *bblock k*.

After obtaining the desirable initial solution, we present our optimization method for the ILP (4) in Algorithm 2, where the objective value is increased at every iteration, and is converged to a maximal solution. Experimentally, Algorithm 2 only takes a few iterations before achieving the convergence condition. Furthermore, we know from Equation (13) that every iteration of Algorithm 2 can be finished in $\mathcal{O}(\max\{|V| \cdot |E|, |V| \cdot \|\mathbf{C}\|_0\})$, where $\|\mathbf{C}\|_0$ is the number of nonzero elements in tensor $\mathbf{C}$.

```
Algorithm 2 UNP Solver
Input: A connected component of $CG(V, E, W)$,
convergence threshold $\delta = 10^{-4}$;
Output: Solution $\mathbf{x}^*$ of ILP (4);
1:  Initialize $t \leftarrow 0$;
2:  Generate $\mathbf{x}^{(0)}$;
3:  If $x_i^{(0)} = 1$, let $y_i^{(0)} \leftarrow 1$; otherwise, let $y_i^{(0)} \leftarrow$
    $-1$;
4:  repeat
5:    $\forall i \in V$, compute $g_i^{(t)}$;
6:    Obtain $\nabla f(\mathbf{y}^{(t)})$;
7:    $\alpha \leftarrow LineSearch(\mathbf{y}^{(t)})$;
8:    $\mathbf{y}^{(t+1)} \leftarrow \mathbf{y}^{(t)} + \alpha\nabla f(\mathbf{y}^{(t)})$;
9:    $t \leftarrow t + 1$;
10: until $\| \nabla f(\mathbf{y}^{(t)}) \| < \delta$
11: Get $x_i^*$ by rounding $\sigma(y_i^{(t)})$ to the nearest
    integer, $\forall i \in V$.
```

4.3. Algorithm analysis

Lemma 2. *Under* Equation (13), $\sum_i w_i \Delta g_i \geq 0$

Proof. *By* Equation (13),

$$\sum_i w_i \Delta g_i = \sum_i w_i \gamma g_i \{(1 - \sigma(y_i))\Delta y_i - \sum_j B_{ij}\sigma(y_j)\Delta y_j$$
$$- \sum_k C_{ikl} \sum_l \frac{\sigma(y_k)(1 - \sigma(y_k))\sigma(y_l)}{1 - \sigma(y_k)\sigma(y_l)} \Delta y_k\}.$$

Let $u_i = w_i g_i$, $\mathbf{u} = (u_i) \in \mathbb{R}^l$, and the auxiliary matrix $\mathbf{A} = (\mathbf{A}_{ij}) \in \mathbb{R}^{n\times n}$ has the following elements: $\mathbf{A}_{ij} = 1 - \sigma(y_i)$, if $i = j$;

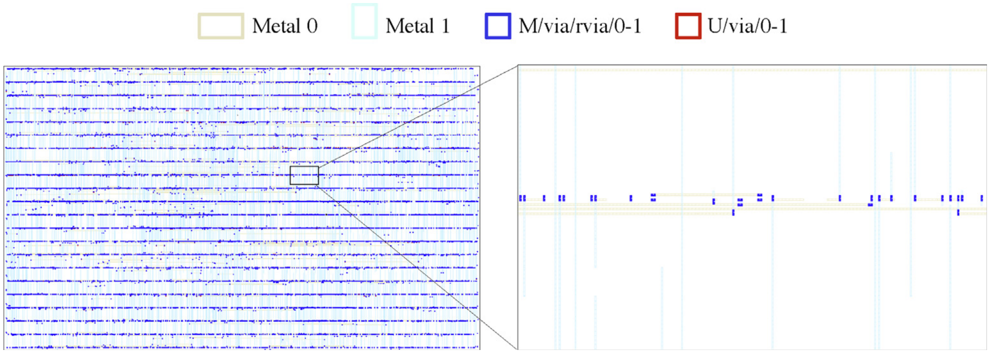

Fig. 13. The result of benchmark primary2 and its a partial layout on vias between Metal 0 and Metal 1.

$\mathbf{A}_{ij} = -\sigma(y_j)$, if $e_{ij} \in E$; $\mathbf{A}_{ik} = -\sum_{l\in V} \frac{\sigma(y_k)(1-\sigma(y_k))\sigma(y_l)}{1-\sigma(y_k)\sigma(y_l)}$, if $i,k,l \in CS$. Then $\sum_i w_i \Delta g_i = \gamma \mathbf{u}^\top \mathbf{A}\Delta\mathbf{y}$. From (12), we know $\Delta\mathbf{y} = \mathbf{y}^{(t+1)} - \mathbf{y}^{(t)} = \alpha\nabla f(\mathbf{y})$. According to the definition (13) of $\nabla f(\mathbf{y})$ and $\mathbf{A}_{ij}$, we have $\Delta\mathbf{y} = \alpha\gamma\mathbf{A}^\top\mathbf{u}$. Thus $\sum_i w_i \Delta g_i = \alpha\gamma^2 \mathbf{u}^\top \mathbf{A}\mathbf{A}^\top \mathbf{u} \geq 0$.

According to Lemma 2, $\sum_i w_i \Delta g_i \geq 0$ in every iteration, thus we have following Theorem 2.

Theorem 2. *Under* Equation (13), *$f(\mathbf{y})$ does not decrease.*

Corollary 1. *Strict inequality $\sum_i w_i \Delta g_i > 0$ cannot be achieved, since* $\mathbf{A}\mathbf{A}^\top$ *is not positive definite.*

Proof. We prove that $\mathbf{A}\mathbf{A}^\top$ is not positive definite. The ILP (4) contains at least one node, e.g., $x_i = 1$. It follows, $\forall j \in V, e_{ij} \in E$, $x_j = 0$. Then, all the elements of i-th row of $\mathbf{A}$ are zero, i.e., $\mathbf{A}$ does not have full rank. Consequently, at least one of the eigenvalues of $\mathbf{A}\mathbf{A}^\top$ is zero.

Theorem 3. *Algorithm 2 converges to a local maximum.*

Proof. Since $\forall i$, $x_i = \sigma(y_i) : \mathbb{R} \to [0,1]$, then from Equation (13), it can be easily deduced that $\forall i$, $g_i : \mathbb{R} \to [0,1]$. Consequently, $\sum_i w_i g_i \leq \mathbf{w}^\top \mathbf{1}$ holds. And by Theorem 2, $f(\mathbf{y}) = \sum_i w_i g_i$ always increases. Thus Algorithm 2 converges, and which stop when the gradient $\| \nabla f(\mathbf{y}) \| = 0$, i.e., in a local maximum.

We can achieve a local optimal result by performing Algorithm 2. In this paper, we skip the detailed Proof due to space limitation. In addition, as will be verified in experiments, if Algorithm 2 starts from a desirable initial solution $\mathbf{x}^{(0)}$ by Algorithm 1, it likely returns a near global optimal result.

4.4. Handle guiding template cost

To obtain a better DSA guiding template assignment result, guiding template cost should be considered. We impliedly calculate total guiding template cost by calculating the cost of building-blocks and the weight of connected template edges. The cost w_{bi} of building-block i can be calculated with Equation (2), i.e. the cost of building-block i:

$$w_{bi} = w_{C_k}, (k = 1,2,3,4),$$

if i belongs to the building-block group C_k. Similarly, the weight w^e_{ij} of template edge e_{ij} in guiding template T_l can be calculated with Equation (2), i.e.

$$w^e_{ij} = w^e_{T_l}, (l = 4,5,6),$$

if e_{ij} belongs to the building-block group T_l.

Let binary variable $x_i = 1$ denote that building-block i is selected. Then,

$$MR + \beta \cdot IR = \sum_{i \in V} w_i x_i$$

denotes the main objective, i.e., manufacture rate and insertion rate. And $\sum_{i\in V} w_{bi} x_i$ denotes the total cost of building-blocks, and $\frac{1}{2}\sum_{e_{ij}\in E_T} w^e_{ij} x_i x_j$ denotes the total weight of template edges. Thus

$$Tcost = \alpha\{\sum_{i\in V} w_{bi}x_i + \frac{1}{2}\sum_{e_{ij}\in E_T} w^e_{ij}x_i x_j\}$$

is the total template cost, where α is the weight parameter between the main objective and template cost objective. At last, we formulate the guiding template cost aware DMRD problem as the following integer programming (IP):

$$\max_{\mathbf{x}} \quad \sum_{i\in V} w_i x_i - \alpha\{\sum_{i\in V} w_{bi}x_i + \frac{1}{2}\sum_{i,j\in V} w^e_{ij}x_i x_j\} \tag{15}$$

$$\text{s.t.} \quad 4(a)\ 4(b)\ 4(c).$$

Table 2
Comparison between the TCAD′17 work and Our-UNP.

Benchmarks	#V	TCAD′17 [14]			Our-UNP		
		MR(%)	IR(%)	CPU(s)	MR(%)	IR(%)	CPU(s)
`des-perf-1`	736470	94.04	76.89	962.00	97.15	84.04	227.62
`des-perf-a`	765166	95.98	76.41	928.00	98.12	82.12	187.74
`des-perf-b`	720412	96.68	83.35	951.00	98.60	87.52	238.51
`fft-1`	238324	94.79	78.53	317.00	96.32	85.65	22.32
`fft-2`	255324	95.64	81.70	336.00	97.42	87.30	28.33
`fft-a`	234441	96.45	84.11	328.00	97.85	87.91	23.98
`fft-b`	247866	95.36	79.66	334.00	97.54	85.99	27.64
`pci-bridge32-a`	198376	96.80	82.88	258.00	98.43	87.77	19.77
`pci-bridge32-b`	172483	97.55	88.74	273.00	98.49	90.78	23.12
Avg.	396540	95.92	81.36	520.78	97.77	86.56	88.78
Ratio		0.98	0.94	5.87	1.00	1.00	1.00

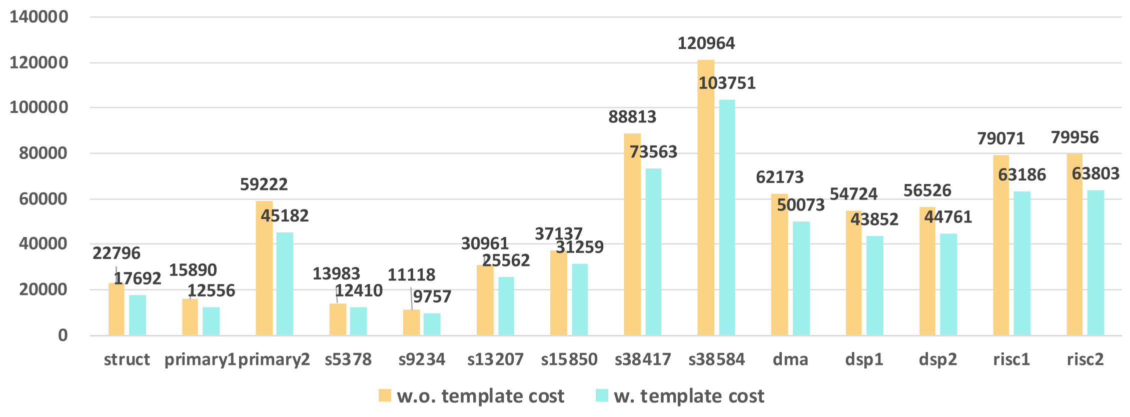

Fig. 14. Comparison on template cost between with and without guiding template optimization.

Let $w_i^v = w_i - \alpha w_{bi}$, then the IP formulation of Problem (15) is equivalent to:

$$\max_{\mathbf{x}} \quad \sum_{i \in V} w_i^v x_i - \frac{1}{2}\alpha \sum_{i,j \in V} x_i w_{ij}^e x_j \tag{16}$$

$$\text{s.t.} \quad 4(a)\ 4(b)\ 4(c).$$

Problem (15) can be directly solved by IP solver. Furthermore, as the process of Problem (5) to Problem (10), Problem (15) also can be formulated to a similar problem as (10). And then, our UNP solver in Algorithm 2 can be invoked to solve Problem (15).

5. Experimental results

Our proposed algorithms are implemented in C++ and run on a personal computer with 2.7 GHz CPU, 8 GB memory and Unix operating system. We test our method on MCNC benchmarks and an industry Faraday benchmarks, provided by Fang et al. [14]. As in Ref. [21], layouts of all benchmarks are transformed to grid models, where a grid size is one metal pitch, and the optical resolution limit spacing d_s of adjacent guiding templates is set to one metal pitch too. The user-defined parameter β and α are set to 1 and 0.01, respectively.

5.1. Effectiveness of ILP

To evaluate the performance of the proposed ILP in Section 4.1, we compare the obtained results with the ILP results in TCAD′17 [14] and ASPDAC′17 [18]. The experimental comparisons are reported in Table 1. Columns “TCAD’17 [14]” and “ASPDAC’17 [18]” are the results in Refs. [14,18], respectively. Column “Our-ILP” is obtained by performing the CPLEX solver [24] to solve our ILP (4) in Section 4.1. Moreover, in this table, column “#V” lists the numbers of vias, and column “CPU(s)” is the runtime in second. “MR(%)” and “IR(%)” are, respectively, the manufacture rate and the redundant via insertion rate.

$$\mathrm{MR} = \frac{\#\mathrm{MV}}{\#\mathrm{V}} \times 100\%, \qquad \mathrm{IR} = \frac{\#\mathrm{RV}}{\#\mathrm{V}} \times 100\%.$$

In the above equations, #MV is the number of manufacturable vias (excluding redundant vias), #RV is the number of vias with redundant vias.

The difference between the results in “Our-ILP” and in “TCAD’17 [14]” is that our work considers multiple vias and dummy via insertion but work TCAD′17 [14] does not. Compared with the computational results of “TCAD’17 [14]”, from the row “Ratio”, our ILP (4) improves MR and IR up to 7% and 9%, respectively. These improvements mainly result from the help of multiple vias and dummy via insertion. Naturally, considering multiple vias and dummy via insertion will extremely increase the size of solution space, which leads to more challenge for solving. In spite of this, our ILP (4) achieves less runtime than the ILP in TCAD′17 [14].

Both of the method in ASPDAC′17 [18] and our ILP (4) consider the dummy via insertion as the complementary technique for improving MR and IR. But our ILP (4) also consider multiple redundant vias insertion, which can further improve MR and IR. From the comparison in Table 1, our ILP achieves almost the same MR as the ILP in Ref. [18] But our ILP improves 3% IR than the ILP in Ref. [18]. It must be noted that, the average runtime of the ILP in Ref. [18] is 4.17 × slower than ours. The improvement in runtime owes to our compact solution expression, which greatly reduces the solution space.

5.2. Effectiveness of local optimal algorithm

In this subsection, we design another experiment to show the performance of our UNP algorithm in Algorithm 2. In Table 1, the data in “Our-UNP” are obtained by our UNP algorithm in Section 4.2. Compared with the results in column “Our-ILP”, “Our-UNP” achieves almost the same quality of results. However, the average runtime of our UNP algorithm is 3.63 × less than our ILP based method. In order words, compared with our ILP, our UNP algorithm can save 72% runtime. These comparisons show that our UNP algorithm is very effective and efficient.

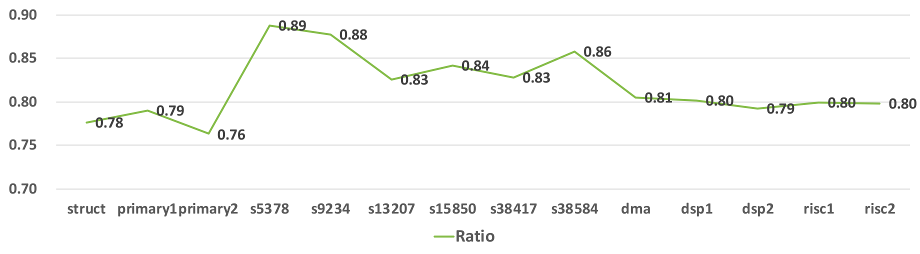

Fig. 15. The ratio on total guiding template cost between with and without guiding template optimization.

In Fig. 13, we plot the result generated by our optimization method on benchmark primary2. In the layouts, we show two metal layers, Metal 0, Metal 1 and two via layers V_{0-1}. In the legend of Fig. 13, M/via/rvia/0–1 (blue) denotes manufacturable via and redundant via on layer V_{0-1}, U/via/0–1 (red) denotes unmanufacturable via on layer V_{0-1}.

To further verify the efficiency of our UNP algorithm, we compare our UNP solver with the ILP solver in TCAD′17 [14] on nine much larger benchmarks in Ref. [14] modified from ISPD 2015 Placement Contest [25]. The experimental results are listed in Table 2. Compared with “TCAD’17 [14]”, “Our-UNP” improves the average manufacture rate by 2% and the average insertion rate by 6%. More important, from Table 2, it can be seen that the average runtime of ILP solver in Ref. [14] is up to 520 s. Specially, the runtime is up to 962 s for benchmark `des-perf-1`. This demonstrate that the ILP based method is seriously time consuming for the large scale cases. By comparison, “Our-UNP” achieves 5.87 $\times$ shorter runtime, i.e., it can save 83% runtime.

5.3. Guiding template cost

For the redundant via insertion with DSA guiding template assignment problem, exist works do not consider the cost of guiding template, we first concurrently optimize the guiding template cost. To evaluate the performance of the guiding template cost optimization, we compare our UNP solver with guiding template cost (objective (16)) and without guiding template cost (objective (4)). In this paper, we set the costs of seven useable guiding template $T_1, T_2, T_3, T_4, T_5, T_6$ and T_7 are $w_{T_1} = 0$, $w_{T_2} = 1$, $w_{T_3} = w_{T_4} = 3$, $w_{T_5} = w_{T_6} = 5$ and $w_{T_7} = 8$. This setting satisfies the rule that the cost of complex guiding template with more holes should be greater than the simple guiding template with less holes. The comparison result are ploted in Figs. 14 and 15. From Fig. 14, it can be seen that considering guiding template cost in our UNP solver can greatly reduce the total guiding template cost for every test circuit. From Fig. 15, we know that the average ratio of total guiding template cost between “with cost optimization” and “without cost optimization” is up to 82%. In other words, considering guiding template cost in our UNP can reduce 18% total guiding template cost.

6. Conclusion

In this paper, we have concurrently considered DSA guiding template assignment with multiple redundant via and dummy via insertion problem. In addition, we first optimize guiding template cost in this problem. Thanks to building-blocks, the vertices number in conflict graph can be effectively reduced. On the conflict graph, we model the problem as an ILP formulation, and relax it to an unconstrained nonlinear programming. We develop a line search optimization algorithm to obtain a local optimal solution. Experimental results demonstrate multiple redundant via insertion is effective for improving insertion rate and manufacture rate. In addition, our guiding template cost aware method greatly reduce the total guiding template cost.

Acknowledgements

This work was supported in part by the National Natural Science Foundation of China (NSFC) under Grants 61672005 and 61907024, and the Research Grants Council of Hong Kong SAR (Project No. CUHK24209017), and the Starting Research Fund from Minnan Normal University (Project No. KJ18009).

References

[1] S. Jeong, J.Y. Kim, B.H. Kim, H. Moon, S.O. Kim, Directed self-assembly of block copolymers for next generation nanolithography, Mater. Today 16 (12) (2013) 468–476.
[2] Y. Zorian, D. Gizopoulos, C. Vandenberg, P. Magarshack, Guest editors introduction: design for yield and reliability, IEEE Trans. Comput. Aided Des. Integr Circuits Syst. 21 (12) (2004) 2197–2208.
[3] G. Xu, L.D. Huang, D.Z. Pan, M.D.F. Wong, Redundant-via enhanced maze routing for yield improvement, in: Proc. Of IEEE/ACM Asia and South Pacific Design Automation Conference, 2005, pp. 1148–1151.
[4] J. Ou, B. Yu, X. Xu, J. Mitra, Y. Lin, D.Z. Pan, DSAR: DSA aware routing with simultaneous DSA guiding pattern and double patterning assignment, in: Proc. Of ACM International Symposium on Physical Design, 2017, pp. 91–98.
[5] H.-Y. Chen, M.-F. Chiang, Y.-W. Chang, L. Chen, B. Han, Full-chip routing considering double-via insertion, IEEE Trans. Comput. Aided Des. Integr Circuits Syst. 27 (5) (2008) 844–857.
[6] K.-Y. Lee, C.-K. Koh, T.-C. Wang, K.-Y. Chao, Fast and optimal redundant via insertion, IEEE Trans. Comput. Aided Des. Integr Circuits Syst. 27 (12) (2008) 2197–2208.
[7] S.-T. Lin, K.-Y. Lee, T.-C. Wang, C.-K. Koh, K.-Y. Chao, Simultaneous redundant via insertion and line end extension for yield optimization, in: Proc. Of IEEE/ACM Asia and South Pacific Design Automation Conference (ASP-DAC), 2011, pp. 633–638.
[8] H. Yi, X.-Y. Bao, J. Zhang, R. Tiberio, J. Conway, L. Chang, S. Mitra, H.-S.P. Wong, W.M. Tong, Contact-hole patterning for random logic circuits using block copolymer directed self-assembly, in: Proc. Of SPIE, vol. 8323, 2012.
[9] J. Ou, B. Yu, J.-R. Gao, D.Z. Pan, Directed self-assembly cut mask assignment for unidirectional design, J. Nanolithogr. MEMS, MOEMS 14 (3) (2015) 031211-1–031211-8.
[10] Y. Yang, Y.J. Choi, S.O. Kim, J.U. Kim, Directed self-assembly of cylinder-forming diblock copolymers on sparse chemical patterns, Soft Matter 11 (22) (2015) 4496–4506.
[11] H.-S.P. Wong, C. Bencher, H. Yi, X.-Y. Bao, L.-W. Chang, Block copolymer directed self-assembly enables sublithographic patterning for device fabrication, in: Proc. Of SPIE, vol. 8323, 2012.
[12] A. Latypov, T.H. Coskun, G. Garner, M. Preil, G. Schmid, J. Xu, Y. Zou, Simulations of spatial DSA morphology, DSA-aware assist features and block copolymer-homopolymer blends, in: Proc. Of SPIE, vol. 9049, 2014, San Jose, CA, USA.
[13] S. Shim, W. Chung, Y. Shin, Redundant via insertion for multiple-patterning directed-self-assembly lithography, in: Proc. Of ACM/IEEE Design Automation Conference, 2016, pp. 410–417.
[14] S.-Y. Fang, Y.-X. Hong, Y.-Z. Lu, Simultaneous guiding template optimization and redundant via insertion for directed self-assembly, IEEE Trans. Comput. Aided Des. Integr Circuits Syst. 36 (1) (2017) 156–169.
[15] Y. Badr, A. Torres, P. Gupta, Mask assignment and synthesis of DSA-MP hybrid lithography for sub-7nm contacts/vias, in: Proc. Of ACM/IEEE Design Automation Conference, 2015, pp. 1–6.
[16] Z. Xiao, Y. Du, H. Tian, M.D.F. Wong, H. Yi, H.-S.P. Wong, H. Zhang, Directed self-assembly (DSA) template pattern verification, in: Proc. Of ACM/IEEE Design Automation Conference, 2014, pp. 1–6.
[17] J. Kuang, J.-J. Ye, F.-Y. Young, Simultaneous template optimization and mask assignment for DSA with multiple patterning, in: Proc. Of IEEE/ACM Asia and South Pacific Design Automation Conference, 2016, pp. 75–82.
[18] C.-Y. Hung, P.-Y. Chou, W.-K. Mak, Optimizing DSA-MP decomposition and redundant via insertion with dummy vias, in: Proc. Of IEEE/ACM Asia and South Pacific Design Automation Conference, 2017.
[19] J. Ou, B. Yu, D.Z. Pan, Concurrent guiding template assignment and redundant via insertion for DSA-MP hybrid lithography, in: Proc. Of ACM International Symposium on Physical Design, 2016, pp. 39–46.
[20] X. Li, B. Yu, J. Ou, J. Chen, D.Z. Pan, W. Zhu, Graph based redundant via insertion and guiding template assignment for DSA-MP, IEEE Trans. Very Large Scale Integr. Syst. 26 (11) (2018) 2504–2517.
[21] S.-Y. Fang, Y.-X. Hong, Design optimization considering guiding template feasibility and redundant via insertion for directed self-assembly, in: Proc. Of IEEE Asia Pacific Conference on Circuits and Systems (APCCAS), 2016, pp. 526–529.
[22] B.N. Clark, C.J. Colbourn, D.S. Johnson, Unit disk graphs, Discrete Math. 86 (13) (1990) 165–177.
[23] J. Nocedal, S.J. Wright, Numerical Optimization, second ed., Springer, 2006.
[24] CPLEX, . [Online]. Available: http://www-01.ibm.com, 2016 July.
[25] I.S. Bustany, D. Chinnery, J.R. Shinnerl, V. Yutsis, ISPD 2015 benchmarks with fence regions and routing blockages for detailed routing-driven placement, in: Proc. Of ACM International Symposium on Physical Design, 2015, pp. 157–164.