A Cost-Driven Fracture Heuristics to Minimize Sliver Length

Xu Ma, Shangliang Jiang and Avideh Zakhor, Fellow, IEEE

Department of Electrical Engineering and Computer Sciences
University of California at Berkeley, CA, 94706, USA
Email: {maxu,sjiang,avz}@eecs.berkeley.edu

ABSTRACT

In optical lithography industry, mask pattern is first fractured into basic trapezoids, and then fabricated by the variable shaped beam mask writing machine. Ideally, mask fracture tools aim at suppressing the trapezoid count in order to speed up the writing time, and minimizing the external sliver length in order to improve CD uniformity. However, the increasing transistor density, smaller feature sizes, and the aggressive use of resolution enhancement techniques (RET) pose new challenges to trapezoid count and external sliver length. This paper proposes a fracture heuristics to improve the performance of currently commercially available fracturing tools. In the proposed approach, the mask layout is first decomposed into elemental rectangles by the rays emitted from each concave corner. Then, a rectangle combination technique is applied to search and eliminate the external slivers from the polygon boundaries by moving them to the center. This approach guarantees that the resulting trapezoid count approaches the theoretical lower bound. Compared to a currently commercially available fracturing tool, our proposed approach effectively reduces the external sliver length by 8% to 13%.

Keywords: Fracture, mask data preparation, Variable Shaped Beam mask writing, sliver

1. INTRODUCTION

In optical lithography, light emitted from the illumination system is transmitted through the mask, and replicates the mask pattern on the wafer.[1] Mask writing is a significant step affecting the fidelity of the printed image on the wafer, and critical dimension (CD) control. During mask data preparation process, the mask pattern is initially fractured into numerous trapezoids. Subsequently, these trapezoids are exposed by the Variable Shaped Beam (VSB) mask writing machine. There are two independent metrics to be optimized during the fracturing process: trapezoid count and total external sliver length. Each trapezoid corresponds to at least one shot in the VSB mask writing machine. Thus, lower trapezoid count is desired, to reduce the writing time. Fig. 1(a) and 1(b) show two different fracture solutions for the same layout with three and four trapezoids, respectively. The solution in Fig. 1(a) leads to shorter mask writing time and lower cost. A trapezoid with lateral dimension w smaller than a prescribed threshold ϵ is referred to as a sliver, as illustrated in Fig. 1.[2,3] In Fig. 1, there are two different kinds of sliver: external and embedded. The external sliver is located on the boundary of the layout, while the embedded sliver is surrounded by other trapezoids. In Fig. 1(a), the top rectangle is an external sliver, and the middle rectangle is an embedded sliver. External slivers introduce inaccuracies in the mask writing process[4] by adversely affecting CD uniformity, while the embedded slivers do not.[2] Total external sliver length is the sum of the boundary length with its lateral dimension smaller than the sliver threshold ϵ. In Figs. 1(a) and 1(b), the external sliver lengthes are indicated in red. Fig. 1(a) shows that the embedded sliver may also contribute to the total external sliver length. Total external sliver length should preferably be minimized, while the effects of the internal sliver length can be ignored.

A number of fracturing algorithms have been proposed over these years in the literature.[4–9] Ohtsuki, et al. developed an algorithm to decompose the rectilinear polygons into minimum number of rectangles by searching the set of maximum independent degenerate lines.[5] A rectilinear polygon only has horizontal and vertical edges. Subsequently, Asano, et al. generalized Ohtsuki's algorithm to be capable of decomposing a polygon with slant edges into minimum number of trapezoids.[6] Moriizumi, et al. proposed a fracturing algorithm to reduce the number of slivers and to maintain CD without splitting.[4] Nakao, et al. introduced a fracturing approach based

Correspondence: avz@eecs.berkeley.edu, 510-643-6777

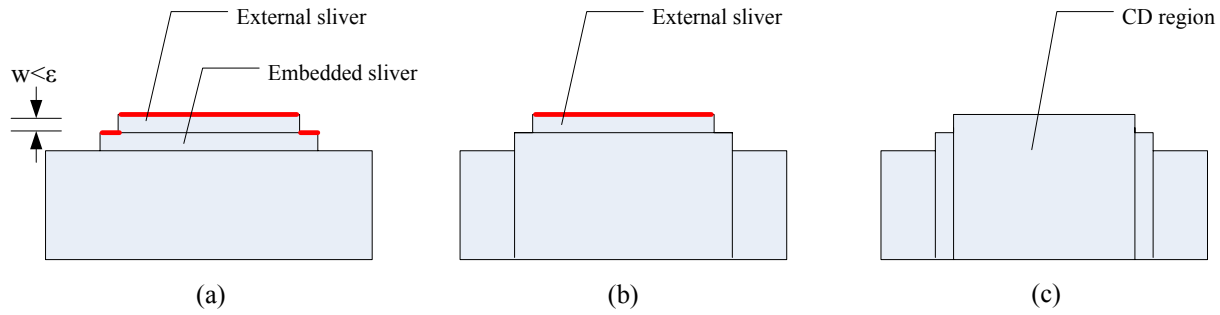

Figure 1. Examples of trapezoid count, sliver and CD preservation. (a) Fracture solution with three trapezoids, one external sliver and one embedded sliver; (b) fracture solution with four trapezoids and one external sliver; (c) fracture solution keeping the intact CD region.

on "relation graph" to suppress sliver and CD partition, and to minimize the trapezoid count.[7] Kahng, et al. suggested a fracturing method relying on integer linear programming (ILP) and a set of heuristics to speed it up.[8] The ILP method took into account all specified requirements, such as trapezoid count, sliver length, keeping intact slant edges and CDs, and maximum shot size limitation. However, this algorithm is prohibitively slow for polygons with large number of vertices, and heuristic partitioning of large polygons may severely degrade the solution quality.[9] As such, Kahng, et al. proposed a fast ray-segment selection heuristics to find a near-optimal fracture solution.[9]

The main challenge in the current fracturing tools include increasing transistor density, smaller feature sizes, and aggressive use of resolution enhancement techniques (RET). Resolution limit of optical lithographic systems has forced the electronics industry to rely on RET to compensate and minimize mask distortions as they are projected onto semiconductor wafers.[1,10] Optical proximity correction (OPC) is a RET that modifies the mask by adding sub-resolution assist features (SRAF) to the mask pattern such that the output pattern is as close to the desired pattern as possible.[10] These SRAFs introduce numerous additional vertices and corresponding edges, thus dramatically increasing the trapezoid count and total external sliver length in the resulting fractured pattern. In this paper, we propose a cost-driven fracturing heuristics to simultaneously reduce the trapezoid count and the total external sliver length. In our proposed approach, the resulting trapezoid count may reach the theoretical lower bound described by Kahng et al.[9] In addition, rectangle combination technique (RCT) is introduced to search, find and move the external slivers from the polygon boundaries to its centers. The proposed algorithm effectively distinguishes between the harmful external slivers and harmless embedded slivers.

The remainder of this paper is organized as follows. The fracturing problem is formulated in Section 2. The cost-driven fracture heuristics based on rectangle combination is described in Section 3. Simulations are included in Section 4, where the effects of the proposed algorithm on mask fracturing are shown. Conclusions are in Section 5.

2. FRACTURING PROBLEM FORMULATION

Current VSB mask writing machines fabricate masks by sequential exposure of the basic trapezoid obtained from the fracturing step. The requirements of the fracture pattern can be summarized as follows.[7,8]

(1) Mask pattern must be fractured into a set of trapezoids without overlap. Overlapping regions on the mask are overexposed and scatter more electrons to the periphery, which lead to variation of the desired trapezoid dimension.

(2) External slivers result in distortion of the prescribed trapezoid geometries; thus, total external sliver length should preferably be minimized.

(3) In order to reduce shot count and cost of mask writing, the number of trapezoids in the resulting fracture pattern should preferably be minimized.

(4) In VSB mask writing machine, slant edges are harder to control than axis-parallel edges. Thus, partition of the slant edges is prohibited.

(5) Regions including elements with critical dimension (CD) should be preserved in one shot without being split. Am example of this is gate of a transistor, where special fabrication accuracy is required. When a CD region is separated and exposed by different shots, CD uniformity is degraded. For example, in Fig. 1(c), the middle rectangle is a CD region without any splits, and hence the fabrication accuracy is preserved. However, in Figs. 1(a) and 1(b), the CD region is separated into several rectangles, which is undesirable.

(6) The maximum linear size of each shot produced by current VSB mask writing machine is between $2\mu m$ and $3\mu m$ on the mask scale. Thus, the linear size of each trapezoid on the fracture pattern should be restricted to this limit.

In this paper, we only consider decomposition of the rectilinear polygons with axis-parallel edges into disjoint rectangles. Thus, requirements (1) and (4) are inherently satisfied, and therefore all trapezoids in the fracture pattern are rectangles. In addition, the heuristics described in next section is applied to simultaneously reduce the trapezoid count and the total external sliver length. Thus, requirements (2) and (3) are the foci of our algorithm. Requirements (5) and (6) and mask patterns with 45-degree slant edges are part of our future work.

Let P be the mask pattern, and F be the fracture pattern. The mask fracturing problem is to find an optimal fracture pattern $\hat{F}$, satisfying

$$\hat{F}(P) = \arg\min_{F \in \mathcal{C}} \{\#(\text{trapezoid}) + \gamma L(\text{external sliver})\}, \tag{1}$$

where $\mathcal{C}$ is the set of all fracture patterns, $\#$(trapezoid) is the number of trapezoid shots, L(external sliver) is the total length of external slivers, and γ is the weight coefficient for total length of external slivers.

3. PROPOSED METHOD

A mask pattern with only horizontal and vertical edges can be represented by a set of rectilinear polygons. We use the term polygon to mean rectilinear polygon in the remainder of this paper for convenience. Our proposed cost-driven fracture heuristics applies three steps to each polygon on the mask. In the first step, an ILP method is applied to choose the maximum independent set of the chords to divide the polygon into several sub-polygons, subject to the condition that the external sliver length induced by these chords is minimized. In the second step, each sub-polygon is decomposed into elemental rectangles by all potential ray-segments emanating from all concave vertices. Rectangle combination technique is then applied to search and move the external slivers from the polygon boundaries to its centers. In the third step, a clean-up processing is introduced to remove the redundant ray-segments, and preserve the convexity constraints for the internal nodes in the fracture pattern. These three steps are iterated until there are no rectangles to be combined any more. Next, the above three steps are described in detail.

3.1. Choosing the Maximum Independent Set of Chords

We start by decomposing a given a polygon P into smaller rectangles by a set of internal rays. The introduction of internal rays increases the trapezoid count. In order to form a set of non-overlapping trapezoids, it has been shown that a polygon should be fractured by the rays emitted from its concave corners.[9] To minimize the number of trapezoids, exactly one ray should be sent from each concave corner.[9] Assuming there are N concave corners in P, then N rays need to be generated. However, two rays may coincide if they share the same two concave vertices; this is defined as a chord.

Fig. 2(a) shows an example of a polygon and its four chords. As seen, B, D, G, H, O, S, V, W, and Z are the concave corners, and $\overline{ZW}$, $\overline{BG}$, $\overline{DS}$, and $\overline{SO}$ are the chords. Choosing a chord effectively removes two concave vertices from the set of concave corners in a polygon, while the ray connected to only one concave vertex removes only one concave vertex. Thus, if we choose M chords in the fracture pattern F, and if the fracture pattern satisfies the convexity constraints described shortly, then the total trapezoid count becomes

$$\#(\text{trapezoid}) = N - M + 1. \tag{2}$$

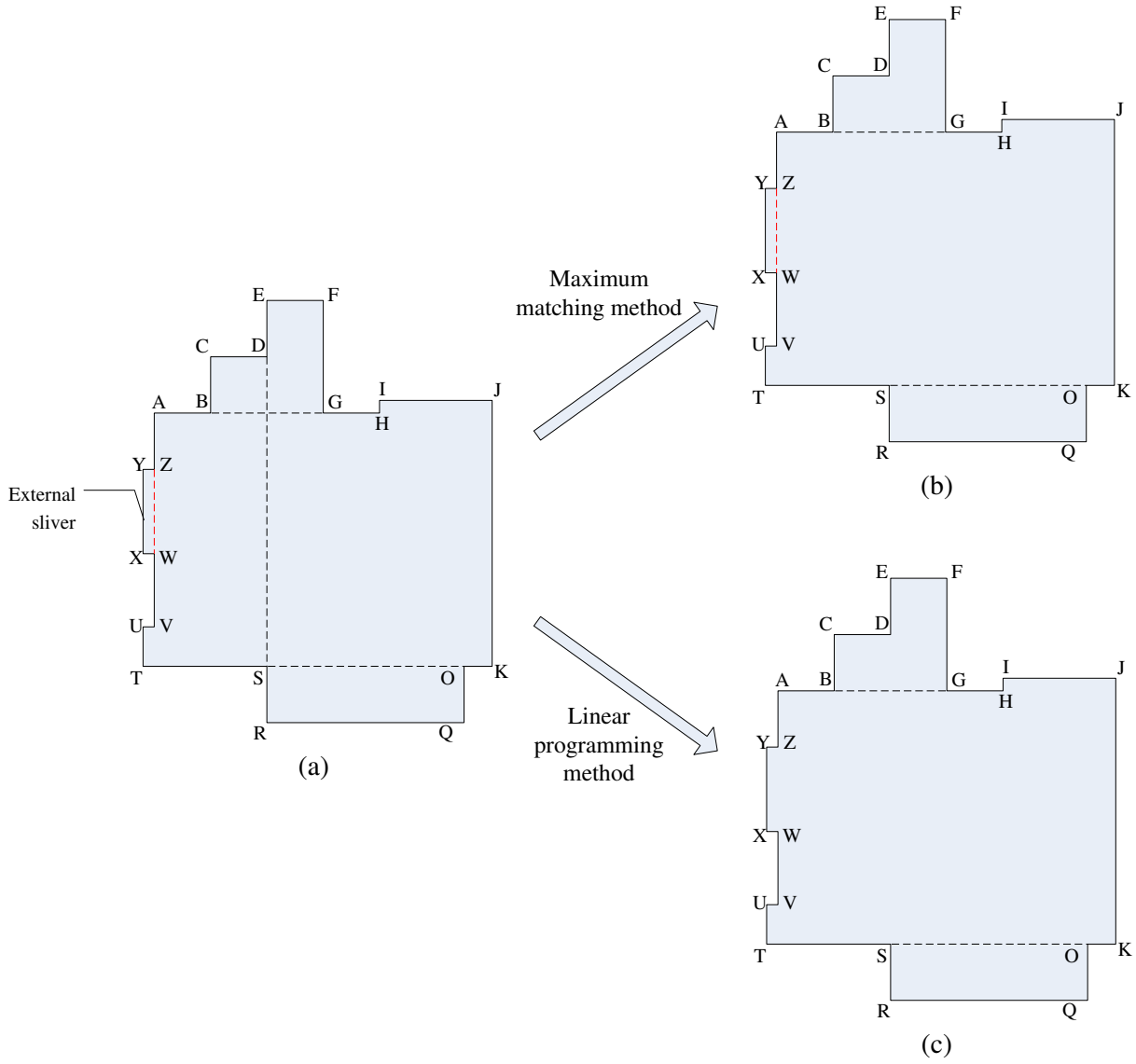

Figure 2. Linear programming method to choose chords; (a) a polygon and four chords shown by dashed lines; (b) chords chosen by the maximum matching method; (c) chords chosen by the linear programming method.

Given M, Eqn. (2) is the theoretical lower bound of the total trapezoid count.[9] It implies that we should choose as many chords as possible to reduce the trapezoid count.[11] We refer to intersecting chords as the "conflicting chords", e.g., $\overline{BG}$ and $\overline{DS}$ are conflicting; $\overline{DS}$ and $\overline{SO}$ are also conflicting. According to the convexity constraints introduced by Kahng, et al.,[8] among the four edges incident to the intersection of two rays called internal node, there could be zero fracture edges, two fracture edges along the same ray, or three fracture edges forming a T-shape. These are illustrated in Fig. 7(b). Clearly, conflicting chords introduce a cross shape as shown in Fig. 7(c), which is prohibited by the convexity constraints. Thus, conflicting chords cannot both be chosen in the final fracture pattern F. Kahng, et al. represented the relationship among the chords by a bipartite graph as illustrated in Fig. 3(a). As seen, the nodes in left and right columns indicate the horizontal and vertical chords, respectively. There is a connection between conflicting chords. The chord choosing problem is to select as many non-conflicting chords as possible; it corresponds to seeking the maximum independent set in the bipartite graph, which can be solved by maximum matching algorithm in polynomial time.[8] Maximum independent set means the maximum set of chords without any conflicting chords. Assume there are M' nodes corresponding to M' chords in the bipartite graph. Let α_m be the indicator of the nodes in the bipartite graph, such that

$$\alpha_m = \begin{cases} 1 & \text{if the } m\text{th chord is chosen} \\ 0 & \text{elsewhere} \end{cases}, \tag{3}$$

where $m = 1, \ldots, M'$. $\beta_{i,j}$ is the indicator of the edges in the bipartite graph, such that

$$\beta_{i,j} = \begin{cases} 1 & \text{if the } i\text{th and } j\text{th chords are conflicting} \\ 0 & \text{elsewhere} \end{cases}, \tag{4}$$

where $i, j = 1, \ldots, M'$. The chord choosing problem defined by Kahng, et al. is formulated as[8]

$$\max\{\sum_{m=1}^{M'} \alpha_m\}, \text{ s.t., } \alpha_i + \alpha_j \leq 1, \text{ if } \beta_{i,j} = 1. \tag{5}$$

The solution to the maximum matching algorithm is shown in Fig. 3(b), and the corresponding fracture pattern is shown in Fig. 2(b).

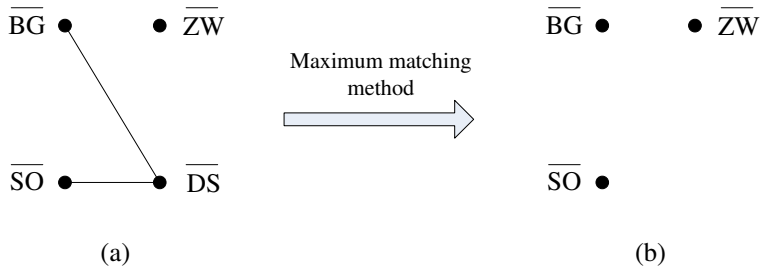

Figure 3. Maximum matching method to choose chords; (a) bipartite graph representing the relationship among chords; (b) chords chosen by the maximum matching method.

The bipartite graph and the maximum matching method do not take into account the external slivers induced by the chords. Thus, this approach selects the chord $\overline{ZW}$ in Fig. 2(b), which increases the total external sliver length in the final fracture pattern. In order to overcome this limitation, we modify the chord choosing problem so as to choose as many non-conflicting chords as possible, subject to minimizing the induced external sliver length. Specifically, we apply the weighed bipartite graph to represent the relationship among chords and the potential induced external slivers as shown in Fig. 4(a).

In Fig. 4(a), we assign a weight $w(\cdot)$ for each chord. For the chords inducing external sliver, such as $\overline{ZW}$, $w(\overline{ZW}) = -1$. For the chords not inducing external sliver, w is the distance from the chord to the nearest parallel boundary of the polygon in units of nanometers. Thus, $w(\overline{BG}) = |\overline{CB}|$, $w(\overline{SO}) = |\overline{SR}|$, and $w(\overline{DS}) = |\overline{CD}|$,

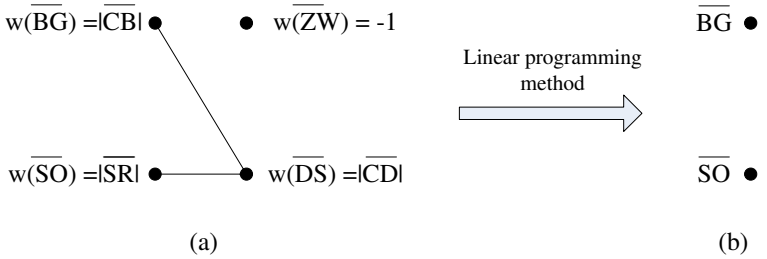

Figure 4. Linear programming method to choose chords; (a) weighed bipartite graph representing the relationship among chords and the potential induced external slivers; (b) chords chosen by the linear programming method.

where $|\cdot|$ means the length of the argument. In other words, the weights are set, such that chosen chords do not induce sliver and are as far from the boundaries as possible. The ILP problem to choose the chords can be formulated as

$$\max\{\sum_{m=1}^{M'} w(m)\alpha_m\},\ \text{s.t.},\ \alpha_i + \alpha_j \leq 1,\ \text{if}\ \beta_{i,j} = 1. \tag{6}$$

Compared with Eqn. (5), the modified chord choosing method uses the weight coefficients to suppress the induced external sliver length. The solution to the above ILP problem in Eqn. (6) for the bipartite graph in Fig. 4(a) is shown in Fig. 4(b), and the corresponding fracture pattern is shown in Fig. 2(c). Using the selected chords, we can divide the entire polygon into several independent child-polygons (P_c), such as: B-C-D-E-F-G, A-B-G-H-I-J-K-O-Q-R-S-T-U-V-W-X-Y-Z, and S-O-Q-R. The original polygon A-B-C-D-E-F-G-H-I-J-K-O-Q-R-S-T-U-V-W-X-Y-Z is referred to as parent-polygon. In the next step, the rectangle combination technique (RCT) is applied to fracture each of these child-polygons, while suppressing the external sliver length.

3.2. Rectangle Combination Technique (RCT)

The definitions used in Section 3.2 are summarized in Table 1. Given a child-polygon P_c, we first emit both possible rays from each concave corner, and grid P_c into elemental rectangles, forming the initial grid pattern. These emanating rays are called grid lines. In Fig. 5, the child-polygon A-B-G-H-I-J-K-O-Q-R-S-T-U-V-W-X-Y-Z is an example of the initial grid pattern of P_c, and Z, H, V and W are the concave corners. We send $\overline{ZE_1}$ and $\overline{ZW}$ from Z, $\overline{HE_1}$ and $\overline{HE_5}$ from H, $\overline{VE_4}$ and $\overline{VE_6}$ from V, and $\overline{WZ}$ and $\overline{WE_3}$ from W. A, I, J, K, T, U, X and Y are convex corners, E_1, E_2, E_3, E_4, E_5, and E_6 are line ends, I_1, I_2, and I_3 are the internal nodes, and B, G, S, and O are the concave vertices in the parent-polygon. A ray is a line starting and ending on polygon boundary, such as $\overline{ZW}$ and $\overline{ZE_2}$. A segment is a portion of a ray, starting and ending on polygon boundary or internal nodes, such as $\overline{ZI_1}$ and $\overline{I_1I_2}$. So far, the child-polygon is divided into 11 elemental rectangles. The main idea behind RCT is to delete the intermediate segment between two adjacent rectangles, thus merging them. For example, if we delete the segment $\overline{ZW}$, then the rectangles Y-Z-W-X and Z-I_1-I_2-W are merged. The final fracture pattern F_c should be a combination of these elemental rectangles in the initial grid pattern.

A grid row or column corresponds to where the grid lines exist. For example, there are 6 grid rows and 4 grid columns in Fig. 5, which are denoted by R_i and C_i, respectively. Any two grid rows or columns compose a grid-pair, such as (R_1, R_4) and (C_1, C_2). The distance of a grid-pair, denoted by $d(\cdot,\cdot)$ is defined as the distance between the two grid rows or columns composing that pair. For instance, $d(R_1, R_4) = |\overline{JE_3}|$ and $d(C_1, C_2) = |\overline{TE_6}|$. A segment on the grid line is referred to as external segment, if it overlaps with the boundary of the parent-polygon, e.g., $\overline{WV}$. Otherwise, the segment is called internal segment, e.g., $\overline{ZW}$ and $\overline{SO}$. Although $\overline{SO}$ is on the boundary of the child-polygon, it is inside the parent-polygon. Thus, $\overline{SO}$ is an internal segment. If an external segment is parallel to an internal segment in a grid-pair, it is called a coupled external segment with respect to this grid-pair. For example, $\overline{YX}$ is a coupled external segment with respect to (C_1, C_2). However, $\overline{IH}$ is not coupled with any grid-pair, because its parallel segment $\overline{JE_1}$ is an external segment, not an internal one. Based on the above definitions, we can describe RCT step as follows:

Step 1: Distance calculation: the distance of all grid-pairs are calculated.

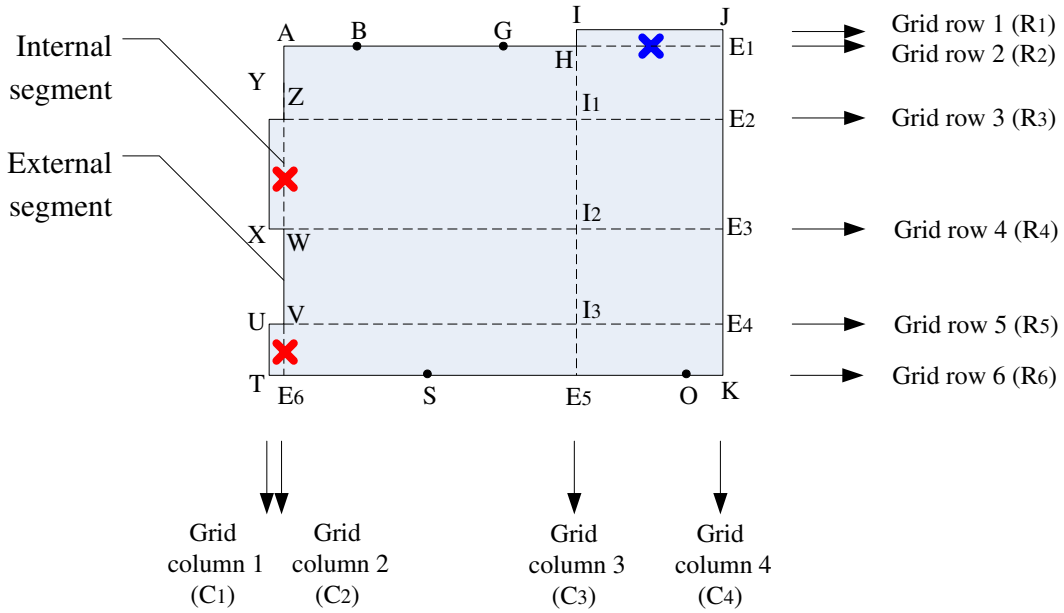

Figure 5. The initial grid pattern of P_c.

Table 1. Summary of definitions.

Terminology	Definitions
Grid row/column	The row/column corresponding to where the grid lines exist.
Grid-pair	The combination of two parallel grid rows/columns.
Distance of grid-pair	The distance between the two grid rows/columns composing that pair.
External segment	The segment of a grid line overlapping on the boundary of the parent-polygon.
Internal segment	The segment of a grid line inside the parent-polygon.
Coupled external segment	If an external segment is parallel to an internal segment in a grid-pair, it is called a coupled external segment with respect to this grid-pair.
Modified segment end	The segment end of a deleted segment.
Modified concave corner	The modified segment end that locates at a concave corner.
Modified line ends	The modified segment end that locates on the boundary of parent-polygon, and not locates at any concave corner.
Modified internal node	The modified segment end locating at a internal node.
Singular segment	The segment not emanating from any concave corner.
Singular concave corner	The concave corner that does not send any rays.

The goal of RCT is to eliminate an external sliver by merging it with its neighboring rectangles inside the child-polygon. First, we need to find the location of slivers. To do so, we calculate the distance for all grid-pairs. If the distance is smaller than the sliver threshold ϵ, we set it to -1. So every grid-pair with distance equal to -1 corresponds to one or more slivers, either embedded or external, depending on whether or not they are located on the boundary of the parent-polygon. For example, in Fig. 5, the distance for grid pairs (R_1, R_2) and (C_1, C_2) are both set to -1, since (R_1, R_2) corresponds to external sliver I-J-E_1-H, and (C_1, C_2) corresponds to external slivers Y-Z-W-X and U-V-E_6-T. In the following steps, the grid-pairs with smaller distances have higher priority to merge.

Step 2: Choose the grid-pair with the minimum distance: the grid-pair with the minimum distance and highest external sliver length are chosen.

The grid-pairs are ordered based on the distances calculated in *Step 1*. The minimum distance value is -1, corresponding to slivers. Smaller positive distance values correspond to narrower rectangles, while larger positive

distance values correspond to wider rectangles. So, in the second step, we select the grid-pair with the minimum distance to merge in *Step 3*. If several grid-pairs have the same minimum distance, then we choose the grid-pair with the longest total length of coupled external segments, because it results in the longest external sliver length. For example, in Fig. 5, (R_1, R_2) and (C_1, C_2) have the same minimum distance of -1. For (R_1, R_2), the total length of coupled external segments is $l_1 = |\overline{IJ}|$. For (C_1, C_2), the total length of coupled external segments is $l_2 = |\overline{YX}| + |\overline{UT}|$. Since $l_2 > l_1$, (C_1, C_2) is selected to be processed in the next step.

Step 3: Block merging: the external slivers are merged to their neighboring polygons.

After finding the locations of external slivers, we eliminate them by merging them with their neighboring rectangles inside the child-polygon. Thus, in this step, we search for the possibility to combine rectangles along the grid-pair selected in *Step 2*, and delete the intermediate segment between the rectangles. For example, along (C_1, C_2), we delete the intermediate segment $\overline{ZW}$, thus merging the two associated rectangles into a larger rectangle Y-I_1-I_2-X. We also delete $\overline{VE_6}$, and generate a larger rectangle U-I_3-E_5-T. The segment ends of the deleted segments, e.g., Z, W, V and E_6, are called modified segment ends, because we change the number of lines connecting to these ends through the RCT process. During the RCT process, modified segment ends may include modified concave corners, e.g., Z, W and V, modified line ends, e.g., E_6, and modified internal nodes. The modified internal nodes are not illustrated in this example. However, if we were to delete segment $\overline{HI_1}$, then I_1 would have been a modified internal node.

Another example is shown in Fig. 6. Fig. 6(a) shows the initial grid pattern of a polygon, where the segments

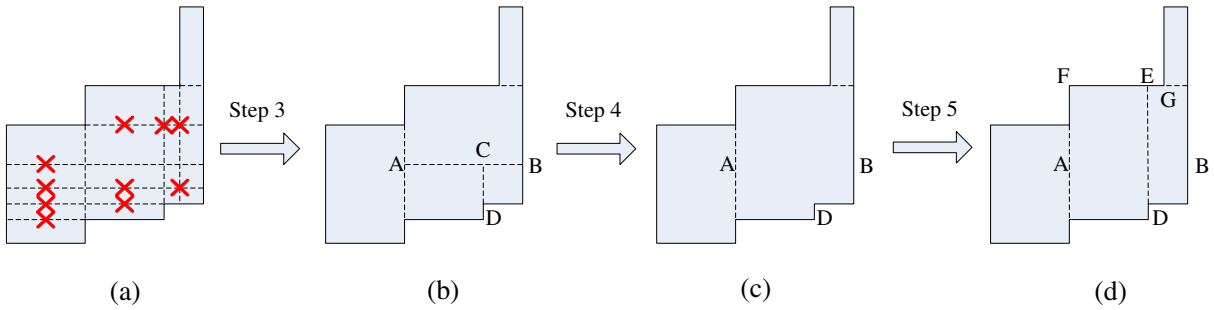

Figure 6. Complement a ray from the singular concave corner; (a) initial grid pattern; (b) result of *Step 3*; (c) result of *Step 4*; (d) result of *Step 5*.

with red cross are deleted in *Step 3* as shown in Fig. 6(b).

Step 4: Internal node checking: the connection configurations of internal nodes are checked and fixed.

Fig. 7(a) shows an example of internal node B. In order to reach the minimum trapezoid count during the rectangle combination process, the connection of an internal node can only have three possibilities as illustrated in Fig. 7(b).[8] In case I, no segment is connected to this internal node; in case II, there is a one-way connection, and in case III, there is T-shape connection. Fig. 7(c) shows another possible connection of an internal node, namely a cross. While the cross is allowed in the intermediate steps of the rectangle combination process, it should be removed after the rectangle combination step is completed, i.e. in the post-processing step described shortly in Section 3.3. In the intermediate steps, the prohibited connection configurations of internal nodes are shown in Fig. 8, where case I is "dead end", and case II is L-shape.

In *Step 3*, we delete some intermediate segments and introduce some modified segment ends, whose connection configurations are changed. In order to constrain their connection configurations to the four possibilities shown in Figs. 7(b) and 7(c), we "check" all modified internal nodes after each rectangle combination. By "check" we mean the following: If there are any internal nodes with connection configuration the same as the two cases listed in Figs. 8, we remove the incident segments to them and degrade them to case I. For instance, if the segments $\overline{AB}$ and $\overline{DB}$ are deleted in Fig. 7(a), then the modified internal node B becomes an L-shape connection; thus, we delete segments $\overline{BC}$ and $\overline{BE}$ to degrade B to case I.

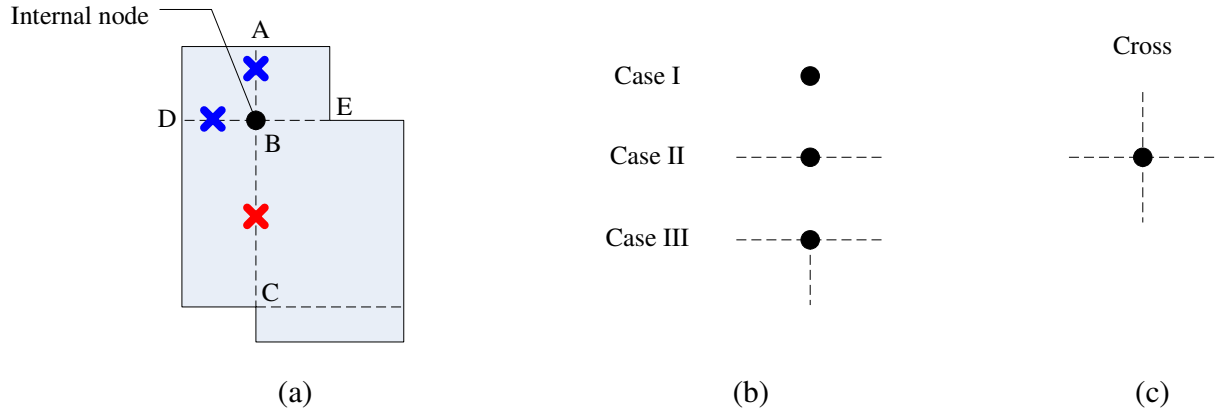

Figure 7. Connection configurations of the internal node; (a) example of an internal node; (b) three kinds of permitted connection configurations; (c) cross connection.

Figure 8. Prohibited connection configurations of internal nodes in the intermediate steps.

In addition, we delete any segment not emanating from a concave corner, which is referred to as singular segment; this is because singular segments increase the trapezoid count.[7] In Fig. 7(a), if the segment $\overline{BC}$ is deleted during the rectangle combination, then $\overline{AB}$ becomes a singular segment.

When we delete the incident segments connected to modified internal nodes and singular segments, we may introduce new modified internal nodes. Thus, we build a list of all of the modified internal nodes to be "checked". If new modified internal nodes are generated, they are pushed to the end of the list. *Step 4* is terminated when the list is empty.

In this step, we only check the modified internal nodes. The checking of the modified concave corners, which are defined in *Step 3* is described shortly in *Step 5*. Also we do not need to check the modified line ends, because in the initial grid pattern, all grid lines emanate from the concave corners. So, if the connection configurations of the concave corners and internal nodes are determined, the connection configurations of the modified line ends are also known. Consider the example in Fig. 6(b), where $\overline{AB}$ is a singular segment, and is therefore removed in *Step 4*. The absence of $\overline{AB}$ configures the internal node C as "dead end"; thus $\overline{CD}$ is removed in Fig. 6(c).

Step 5: Concave corner checking: singular concave corners are found and fixed.

We check all modified concave corners introduced in *Steps 3* and *4* to guarantee that at least one ray is sent from every modified concave corner. The singular concave corner is defined as the one that does not send any rays. If a singular concave corner is found, we emanate a ray from it until the ray reaches another segment. The emanated ray should be in the same direction as the last ray deleted from the singular concave corner, because the ray in this direction is guaranteed not to introduce new modified internal nodes. In other word, we do not want to disturb the current fracturing pattern, so we trace back in the direction of the last deleted ray from the singular concave corner. Consider the example in Fig. 6. In Fig. 6(c), the vertex D becomes the singular concave corner, and $\overline{CD}$ is the last ray deleted from it. So, in *Step 5*, we emanate a ray from D along $\overline{CD}$ as shown in Fig 6(d). As seen, the ray $\overline{DE}$ is stopped when it reaches another segment $\overline{FG}$.

After *Step 5*, we return to *Step 2* and select the grid-pair with the next minimum distance. This process is repeated until we have processed all of the grid-pairs, or there are no rectangles to be combined any more.

3.3. Post Processing

As discussed in Sections 3.1 and 3.2, there are three topography constraints for the fracture pattern. First, exactly one ray should be sent from each concave corner. Second, the connection configuration of any internal

node has to fall into the three cases listed in Fig. 7(b). Finally, any segment not connected to concave corner should be removed. In Section 3.2, The third topography constraint mentioned above is guaranteed by *Step 4*. *Step 4* also constrains the connection configurations of internal nodes to the four possibilities shown in Figs. 7(b) and 7(c). However, after the rectangle combination step, all crosses as shown in Fig. 7(c) should be removed. In addition, *Step 5* guarantees that every concave corner sends at least one ray. So, there may be some concave corners sending two rays. Based on this analysis, only the third topography constraint is fully satisfied. In this section, we introduce a post processing step to ensure that the first and second topography constraints are also satisfied. Specifically, in the post processing step, we check the configurations of all concave corners and internal nodes in the resulting fracture pattern obtained from the previous RCT stage.

3.3.1. Concave Corner Checking

The *Step 5* in the rectangle combination guarantees that each concave corner sends at least one ray. Thus, in the post processing, we are only concerned about the concave corners sending two rays. In this case, we compare the external sliver length induced by each ray. Then, we preserve the ray with shorter external sliver length, and remove the other. The external sliver length induced by a ray is defined as the difference between external sliver length generated by it and the external sliver length "saved" by preserving it. An example is shown in Fig. 9. Figure 9(a) shows the fracture pattern, where the dashed lines $\overline{CF}$, $\overline{CG}$ and $\overline{AB}$ are the grid lines. The

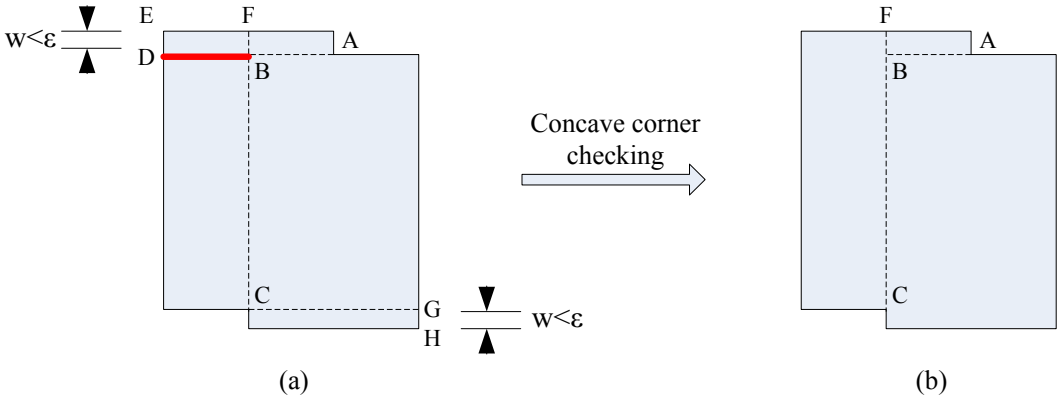

Figure 9. Concave corner checking; (a) fracture pattern with a concave corner sending two rays; (b) result of the concave corner checking process.

polygon is divided into four rectangles, with concave corner C sending two rays $\overline{CF}$ and $\overline{CG}$. In order to keep the topography constraints, one of them must be removed. Since $|\overline{GH}| = |\overline{ED}| = w < \epsilon$, ray $\overline{CG}$ generates an external sliver with length of $l_1 = |\overline{CG}|$. On the other hand, ray $\overline{CF}$ does not generate any external sliver. If $\overline{CF}$ is removed, the segment $\overline{AB}$ is extended to $\overline{AD}$ as illustrated by the red line. Thus, the external sliver length "saved" by preserving $\overline{CF}$ is $|\overline{BD}|$. Therefore, the effective external sliver length induced by $\overline{CF}$ is $l_2 = -|\overline{BD}|$. Since $l_1 > l_2$, we delete $\overline{CG}$ and preserve $\overline{CF}$. The resulting fracture pattern is shown in Fig. 9(b).

3.3.2. Internal Node Checking

The *Step 4* in the rectangle combination guarantees that the connection configuration of any internal node belongs to the four cases listed in Figs. 7(b) and 7(c). As described in Section 3.2, the cross is allowed in the intermediate steps of rectangle combination process, but is prohibited in the final fracture pattern. Thus, in the post processing step, we find the internal nodes in a cross configuration, and "fix" them. For the cross configuration, there are four incident segments to the internal node. Two of them from concave corners must be preserved in the fracture pattern, otherwise the corresponding ray becomes a singular segment as defined in *Step 4* of Section 3.2. Thus, one of the other two incident segments has to be removed to eliminate the cross configuration. In this case, we compare the external sliver length induced by each of the two remaining candidates. The external sliver length induced by a segment is defined similarly to that of a ray. We choose to preserve the segment with shorter induced external sliver length, and remove the other one, thus transforming

the cross configuration to the T-shape connection. An example of this is shown in Fig. 10, where the dashed lines are the grid lines. The polygon is divided into seven rectangles, and the internal node O has a cross connection.

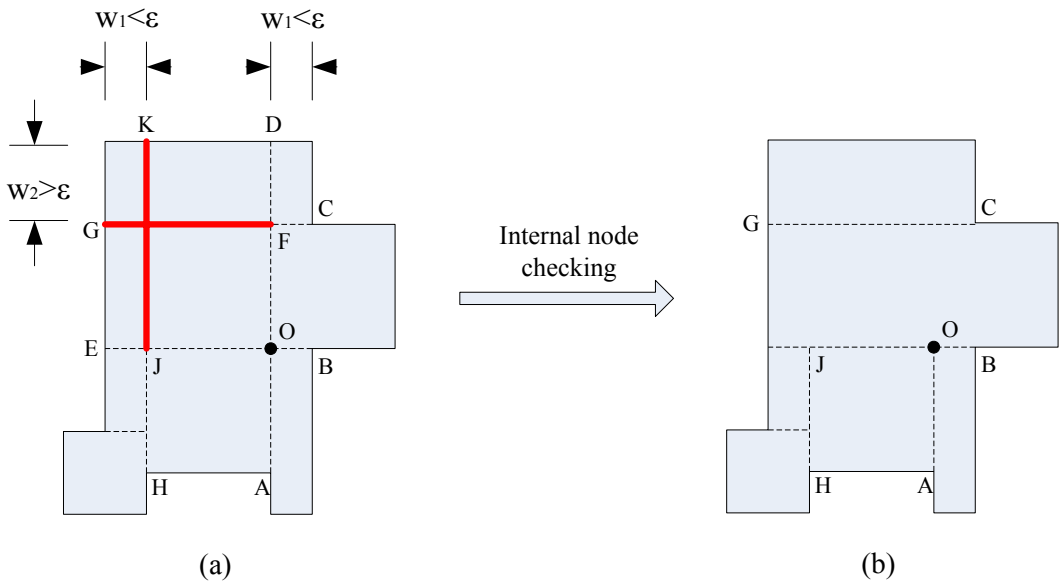

Figure 10. Internal node checking; (a) fracture pattern having a internal node with cross-shape connection; (b) result of the internal node checking process.

Among the four incident segments to O, $\overline{OD}$ and $\overline{OE}$ are not connected to concave corners. In order to keep the topography constraints, one of them has to be removed. Since $|\overline{CF}| = |\overline{EJ}| = w_1 < \epsilon$, segment $\overline{OD}$ generates an external sliver with length of $l_1 = |\overline{DF}|$. If $\overline{OD}$ is removed, segment $\overline{CF}$ is extended to $\overline{CG}$ as illustrated by the red line. However, $|\overline{DF}| = w_2 > \epsilon$, thus $\overline{FG}$ is not an external sliver. On the other hand, segment $\overline{OE}$ does not generate any external slivers. If $\overline{OE}$ is removed, segment $\overline{HJ}$ is extended to $\overline{HK}$ as illustrated by the red line. Thus, the external sliver length "saved" by preserving $\overline{OE}$ is $|\overline{JK}|$. Therefore, the effective external sliver length induced by $\overline{OE}$ is $l_2 = -|\overline{JK}|$. Since $l_1 > l_2$, we delete $\overline{OD}$ and preserve $\overline{OE}$; the resulting fracture pattern is shown in Fig. 10(b).

The three topography constraints mentioned at the beginning of Secion 3.3 are enforced via the post processing step. After this, the fracture patterns for all of the child-polygon are stitched up to form the fracture pattern for the parent-polygon. As an example, the final fracture pattern for the parent-polygon in Fig. 2 is shown in Fig. 11.

4. SIMULATIONS

In this section, we present the simulation results of our proposed fracturing heuristics based on rectangle combination. We compare the trapezoid count, the total external sliver length and the total external sliver count between the proposed algorithm and Calibre^{TM}fracture package.[12] We consider two different sliver thresholds $\epsilon = 10nm$ and $\epsilon = 25nm$. All of the following simulations are based on the OPC layout of a Poly layer with critical dimension $CD = 90nm$.

First, we run pixel-based OPC on the Poly layer of a random logic circuit using PIXbar^{TM}software.[12] The prescribed sliver threshold is usually approximately half of the SRAF's width on the OPC layout. Thus, for the case of $\epsilon = 10nm$, we set the parameter in PIXbar^{TM}specifying the minimum allowed SRAF width to 20nm. Then, we select 30 polygons from the OPC layout to test our proposed algorithm. The performance comparison between Calibre^{TM}fracture package and the proposed algorithm for $\epsilon = 10nm$ is summarized in 2nd to 4th rows of Table 2. It is shown that our proposed algorithm reduces the trapezoid count by 1.61%, the total external sliver length by 12.87%, and the total external sliver count by 23.91%.

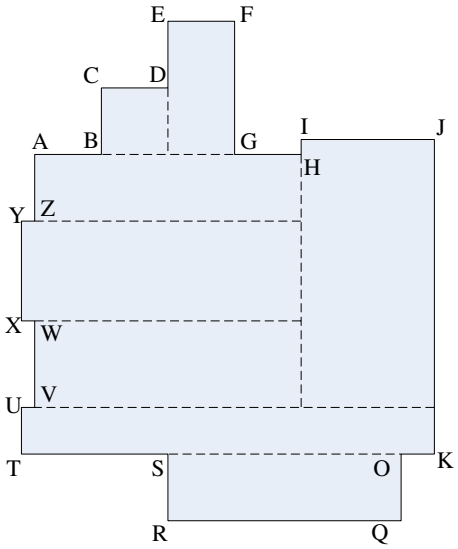

Figure 11. Final fracture pattern for the parent-polygon in Fig. 2.

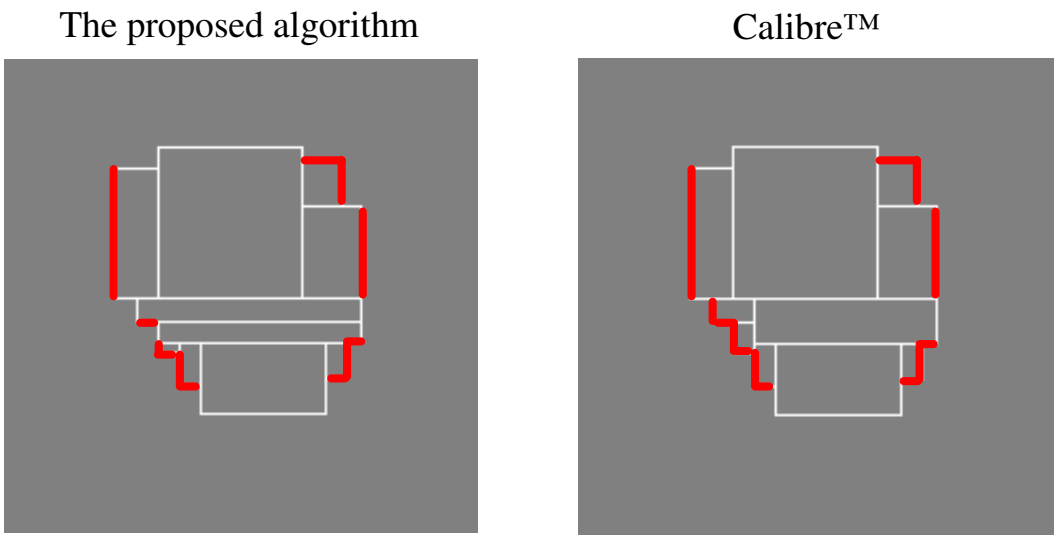

Figure 12. Performance comparison between the proposed algorithm and Calibre^{TM}software with $\epsilon = 25nm$; (a) fracture solution resulting from the proposed algorithm; (b) fracture solution from Calibre^{TM}. Red lines represent the external slivers.

Table 2. Comparison of trapezoid count, external sliver length, and external sliver count between Calibre^{TM}software and the proposed algorithm for $\epsilon = 10nm$ and $\epsilon = 25nm$.

	Calibre^{TM}software	**Proposed algorithm**	**Reduction (%)**
Trapezoid count with $\epsilon = 10nm$	867	853	1.61
Total external sliver length with $\epsilon = 10nm$ (nm)	1585	1381	12.87
Total external sliver count with $\epsilon = 10nm$ (nm)	46	35	23.91
Trapezoid count with $\epsilon = 25nm$	572	570	0.35
Total external sliver length with $\epsilon = 25nm$ (nm)	5572	5133	7.88
Total external sliver count with $\epsilon = 25nm$ (nm)	79	79	0

For the case of $\epsilon = 25nm$, we specify the minimum SRAF width in PIXbar^{TM}to be 50nm. We also select 30 polygons from the entire OPC layout. The performance comparison between Calibre^{TM}fracture package and the proposed algorithm for $\epsilon = 25nm$ is summarized in 5th to 7th rows of Table 2. Our proposed algorithm reduces the trapezoid count by 0.35%, the total external sliver length by 7.88%. However, our proposed algorithm generates the same sliver number as Calibre^{TM}. An example of the performance comparison between the proposed algorithm and Calibre^{TM}software is illustrated in Fig. 12. Fig. 12(a) shows the fracture solution of the proposed algorithm with 10 trapezoids and 188nm external slivers. Fig. 12(b) shows the fracture solution of Calibre^{TM}with 10 trapezoids and 205nm external slivers. Red lines represent the external slivers. The histograms of the trapezoid counts, external sliver lengths, and external sliver numbers for $\epsilon = 10$ and $25nm$ are shown in Figs. 13(a), 13(b) and 13(c), respectively. As expected, the histograms for our proposed algorithm are slightly more concentrated towards the left than that of Calibre^{TM}.

5. CONCLUSION

This paper has developed a cost-driven fracture heuristics for VSB mask writing machine based on rectangle combination. First, a set of optimal chords are chosen using linear programming methods to divide the parent-polygon into several child-polygons. Subsequently, rectangle combination technique is applied to each child-polygon. Finally, the post processing step modifies the local connection configurations of the concave corners and internal nodes in the fracture pattern to guarantee the topography constraints. The proposed heuristics can reach the lower bound of the total trapezoid count, because we emanate exactly one ray from each concave corner, and constrain the internal nodes to the three possible connection configurations described in Section 3.2. In addition, our proposed algorithm effectively reduces the total external sliver length, thus improving the efficiency and accuracy of the mask writing. Simulations illustrate that our heuristics significantly reduces the total external sliver length of a currently commercially available fracturing tool by 8% to 13%. In the future, we plan to investigate the fracturing algorithm to deal with CD maintenance, maximum shot size, and slant polygon edges.

REFERENCES

1. X. Ma and G. R. Arce, *Computational lithography*, Wiley Series in Pure and Applied Optics, John Wiley and Sons, New York, 1 Ed., 2010.
2. M. Bloecker, R. Gladhill, P. D. Buck, M. Kempf, D. Aguilar, and R. B. Cinque, "Metrics to assess fracture quality for variable shaped beam lithography," in *Photomask Technology. Proceedings of the SPIE*, **6349**, pp. 63490Z, 2006.

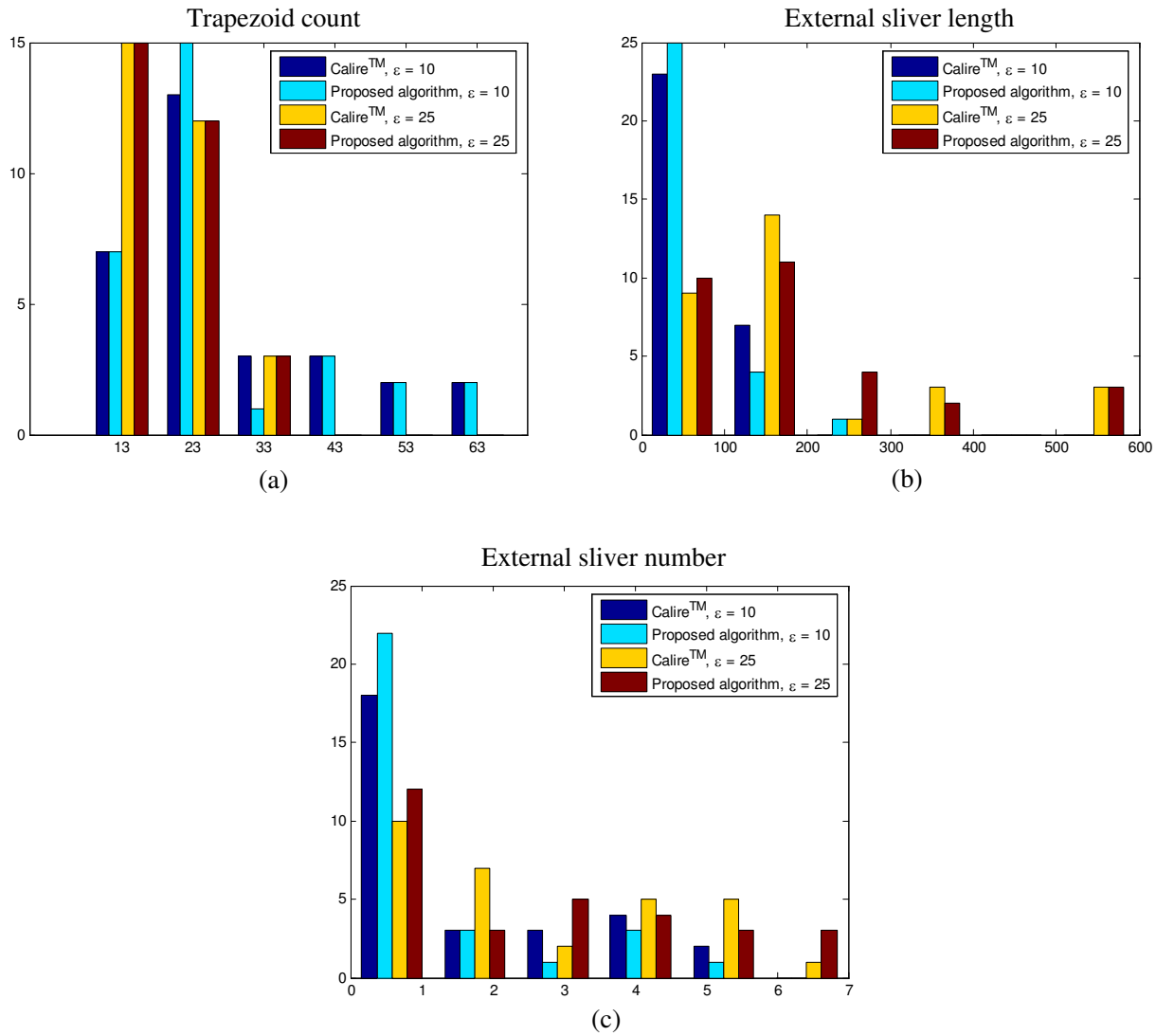

Figure 13. The histograms of (a) the trapezoid counts, (b) external sliver lengths, and (c) external sliver numbers.

3. Y. Zhang, R. Gray, S. Chou, B. Rockwell, G. Xiao, H. Kamberian, R. Cottle, A. Wolleben, and C. Progler, "Mask cost analysis via write time estimation," in *Design and Process Integration for Microelectronic Manufacturing III, Proceedings of the SPIE*, **5756**, pp. 313–318, 2005.
4. K. Moriizumi, H. Taoka, K. Ueyama, H. Morimoto, and T. Munakata, "High-quality drawing data preparation for variable-shape EB drawing systems (I)– fracturing algorithm," in *Proc. 55th Aut. Conv. Soc. Appl. Phys.*, pp. 565, 1994.
5. T. Ohtsuki, M. Sato, M. Tachibana and S. Torii, "Minimum fracturing of composite rectangular region," *Trans. Inf. Process* **24**, pp. 647–653, 1983.
6. T. Asano, T. Asano, and H. Imai, "Partitioning a polygonal region into trapezoids," *J. Assoc. Comput. Mach.* **33**, pp. 291–312, 1986.
7. H. Nakao, M. Terai, and K. Moriizumi, "A new figure fracturing algorithm for variable-shaped EB exposure-data generation," *Electronics and Communication in Japan, Part 3* **83**, pp. 87–102, 2000.
8. A. B. Kahng, X. Xu, and A. Zelikovsky, "Yield- and cost- driven fracturing for variable shaped-beam mask writing," in *Proc. 24th BACUS Symposium on Photomask Technology and Management*, pp. 360–371, Sep. 2004.
9. A. B. Kahng, X. Xu, and A. Zelikovsky, "Fast yield-driven fracture for variable shaped-beam mask writing," in *Photomask and Next-Generation Lithography Mask Technology XI, Proceedings of the SPIE* **6283**, pp. 62832R, Apr. 2006.
10. A. K. Wong, *Resolution enhancement techniques*, vol. 1, SPIE Press, 2001.
11. T. Ohtsuki, "Minimum dissection of rectilinear regions," in *Proc. ISCS*, pp. 1210–1213, 1982.
12. http://www.mentor.com/.