E-Beam Lithography Stencil Planning and Optimization with Overlapped Characters

Kun Yuan and David Z. Pan
ECE Dept. Univ. of Texas at Austin, Austin, TX 78712
{kyuan, dpan}@cerc.utexas.edu

ABSTRACT

Electronic Beam Lithography (EBL) is an emerging maskless nanolithography technology which directly writes the desired circuit pattern into wafer using e-beam, thus it overcomes the diffraction limit of light in current optical lithography system. However, low throughput is its key technical hurdle. In conventional EBL system, each rectangle in the layout will be projected by one electronic shot, through a Variable Shaped Beam (VSB). This would be extremely slow. As an improved EBL technology, Character Projection(CP) shoots complex shapes, so called characters, by putting them into a pre-designed stencil to increase throughput. However, only a limited number of characters can be put on the stencil due to its area constraint. For those patterns not in the stencil, they still need to be written by VSB. A key problem is how to select an optimal set of characters and pack them on the CP stencil to minimize total processing time. In this paper, we investigate a new problem of EBL stencil design with overlapped characters. Different from previous works, besides selecting appropriate characters, their placements on the stencil are also optimized in our framework. Our experimental results show that compared to conventional stencil design methodology without overlapped characters, we are able to reduce total projection time by 51%.

Categories and Subject Descriptors

B.7.2 [**Hardware, Integrated Circuit**]: Design Aids

General Terms

Algorithms, Design, Performance

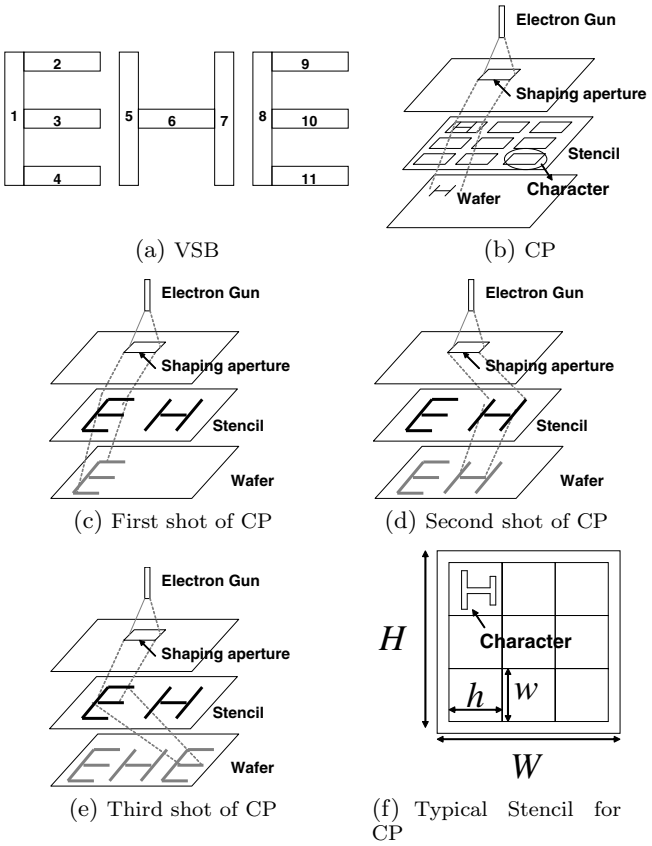
Keywords

Electronic Beam Lithography, Stencil Design

1. INTRODUCTION

As aggressive scaling continues, the conventional 193nm optical photolithography technology is facing the great challenge of printing sub-32nm. For near future, double/multiple patterning lithography has been developed as temporary solution for 32nm, 22nm, even 16nm, technology [1–3]. In the longer future, the semiconductor industries and researchers have been actively pushing on alternative emerging nanolithography to print finer feature size below 16nm, such as Electronic Beam Lithography (EBL), Extreme Ultra Violet (EUV) and nanoimprint.

ISPD'11, March 27–30, 2011, Santa Barbara, California, USA.



(a) VSB (b) CP (c) First shot of CP (d) Second shot of CP (e) Third shot of CP (f) Typical Stencil for CP

Figure 1: Electron Beam Lithography

EBL [4–6] is a maskless technology which shoots desired patterns directly into a silicon wafer, with charged particle beam. The primary advantage is that it is one of the ways to beat the diffraction limit of light of current well-adopted optical lithography [7]. However, the key limitation of electron beam lithography is low throughput.

The conventional type of EBL system is Variable Shaped Beam (VSB). In VSB, the layout is usually decomposed into a set of rectangles, and each one would be shot into resist by dose of electron sequentially. As Fig. 1 (a) shows, the pattern of "EHE" is divided into eleven rectangles and needs total eleven shots. The whole processing time of this technique increases with number of beam shots. This makes its throughput very low for modern complicated design, which is commonly composed of significant number of small rectangles.

The Character Projection (CP) technology [4–6] has been invented for improving the throughput of VSB methods. The key idea is to print some complex shapes in one electronic beam shot, rather than writing multiple small rectangles. This reduces manufacturing time significantly. In detail, as the projection system of CP in Fig. 1 (b) illustrates, a library of layout configurations, called **Character**s, or **Template**s,

are prepared on a **stencil** first. During manufacturing, if any character exists in the targeted design, it will be chosen in the system and projected into the wafer. To print the example of Fig. 1 (a), suppose two characters "E" and "H" are pre designed for the stencil. By adjusting and aligning the shaping aperture and stencial, we can print the patterns of "E", "H", "E" in sequential manner, as Fig. 1 (c)-(e) shows. Totally, it only takes three shots.

Due to less beam shots for the same layout, CP system is much faster than VSB. However, the number of characters is limited due to the area constraint of the stencil. As in the example of Fig. 1 (f), there are only maximum $\lfloor W/w \rfloor \lfloor H/h \rfloor$ characters. For modern design, it is not practical to fully make use of CP, due to numerous distinct circuit patterns. Those patterns which do not match any character are still required to be written by VSB.

Several methodologies have been proposed to design and select group of circuit patterns as characters for minimizing total projection time of both CP and VSB. In [8], frequently-used standard cells are greedily chosen as characters, processed by CP technology. M.Sugihara at el. [9–12] employ integer linear programming to optimize the throughput, given a set of character candidates. Recently, EDA vendor D2S inc [4–6] proposes improving stencil design from a new point of view, but with no detailed algorithm presented. They show that, in practice, when individual character/template is designed, blanking area is usually reserved around its boundaries. By sharing blanks between adjacent templates, more characters can be placed on the stencil than the regular design of Fig. 1 (f), better improving the throughput.

The work of [4–6] implies that, to fully minimize the total projection time of EBL, besides selecting appropriate characters as [8–12], their relative locations on the stencil should be taken into account at the same time due to possible overlapping. In the paper, we will investigate on this new problem of electronic beam lithography stencil design with overlapped characters. One/two dimensional problem is researched separately, depending on whether the available overlapping space of characters is non-uniform in either horizontal or both directions. The main contributions of our work are stated as follows.

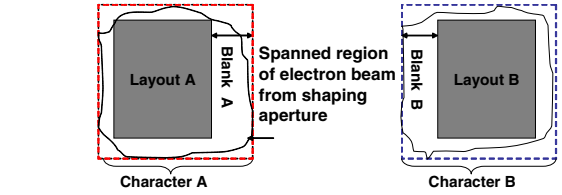
1. We co optimize the selection process of characters and their physical placements on stencil for effective EBL throughput improvement.
2. We propose a four-phase iterative refinement process to conduct one-dimensional stencil design optimization. A Hamilton-path based approach has been developed to solve single-row reordering efficiently and effectively.
3. We develop a Sequence Pair (SP) based simulated annealing framework to optimize general two-dimensional stencil design. Two SP-related techniques have been proposed to ensure correct and fast character placement evaluation, and two specialized perturbation methods have been developed for robust solution improvement of simulated annealing process.

2. PRELIMINARY AND PROBLEM FORMULATION

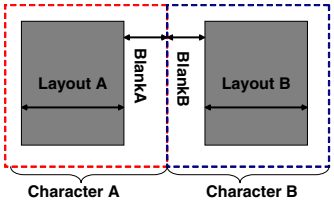
2.1 Overlapped Character

Electronic Beam Lithography (EBL) is a maskless technique, which shoots desired patterns directly into a silicon wafer, and can potentially combat device parameter variations [13–15]. Various investigation [9–12] have been conducted on the optimization of character selection for EPL technology, where no intersection is allowed between templates on the stencil, as shown by Fig. 1 (f). Recently, the work of [4–6] shows that the design of stencil can be further improved by overlapping adjacent characters, which allows more templates to be put and increase the throughput.

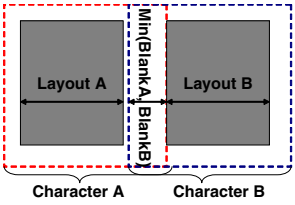
As pointed out by [4–6], when individual character is designed, blanking space is usually reserved around its enclosed rectangular circuit pattern, shown by Fig. 2 (a). The reason is that, when the electron beam is scatted from the shaping aperture of Fig. 1 (b), it could span larger area on the stencil than the layout to be printed. In order to avoid projecting any unwanted image, the white space should be preserved. These blanking areas offer great opportunity for character sharing.



(a) the layout of characters



(b) conventional stencil for (a)



(c) overlapped characters

Figure 2: Overlapped characters for improving the stencil densities.

Suppose the required white space around layout A and B are $BlankA$ and $BlankB$ respectively, in Fig. 2 (a). If the characters are conventionally aligned by edge as Fig. 2 (b), it results in a waste of area. The space between layout A and B is actually $BlankA + BlankB$, which is more than required for both patterns. By contrast, we would greatly reduce the total area of character A and B by sharing an amount of $min(BlankA, BlankB)$ space. In this case, $max(BlankA, BlankB)$ white width is still reserved between layout A and B, which is sufficient for ensuring correct printing image.

2.2 Stencil Design Challenge

The main challenge of stencil design with overlapped characters comes from the fact that, for each character, the amount of required blanking space is not uniform, strongly depending on its enclosed layout patterns. In consequence, for different placements of characters, the area reduction from template overlapping may vary a lot. Therefore, unlike the traditional design of Fig. 1 (f), the number of maximum allowable characters in the stencil is not fixed. To achieve high quality solution, the detailed physical placement information of all the characters must be taken into account. This makes the problem of stencil design with overlapped characters not only different from but also more difficult than conventional non-overlapping one, addressed in [9–12].

As the example of Fig. 3 (a) illustrates, suppose there are three character candidates A-C, and we would like to pack

them into a simple stencil of Fig. 3 (b) for minimum projection time. As easily seen, their blanking spaces are quite different. In conventional design where overlapping is not considered, at most two of them can fit. On the other side, when the blanking space is shared by adjacent characters, the result is correlated with the detailed physical implementation of stencil, and could be different from traditional design. If these three candidates are tried out by the order of A-B-C like Fig. 3 (c), only A and B can be put in. Patterns C is out of bound and has to be processed by VSB technique. This does not lead to higher throughput than conventional non-overlapped methodology. In contrast, if rearranged as C-B-A as Fig. 3 (d), all of these three patterns can be used as CP characters. Obviously, it is a better stencil optimization.

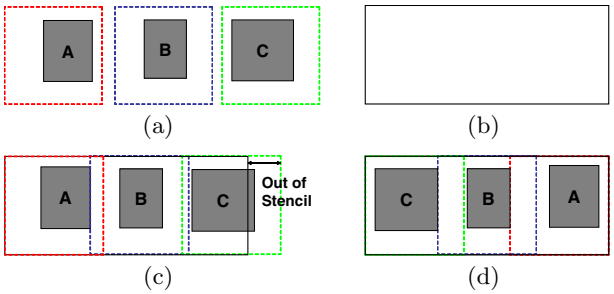


Figure 3: The main difficulty of stencil design with overlapped characters

2.3 Problem Formulation

In this subsection, we will formulate the problem of EBL stencil design with overlapped characters.

Similar to previous work [9–12], we assume a set of character candidates have already been given. To model overlapping information, as Fig. 4 (a) illustrates, assume the blanking spaces of each candidate c_i, from left, right, top and bottom boundaries, are l_i, r_i, t_i and b_i, respectively. The orientation of these candidates are not allowed to be flipped, since it actually becomes a different template, as explained in [10]. When two candidates c_i and c_j are put adjacent to each other horizontally, their maximum allowed overlap is set as o_{ij}^H, which is $min(r_i, l_j)$ as shown by Fig. 4 (b). Similarly, Fig. 4 (c) defines the maximum vertical overlapping margin o_{ij}^V. o_{ij}^H and o_{ij}^V vary for different i and j.

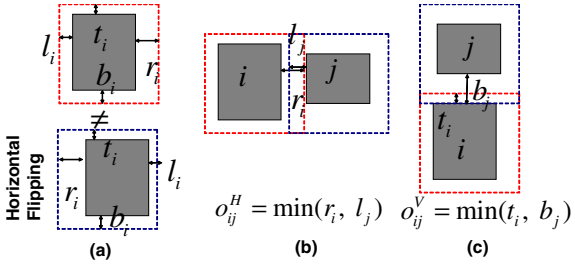


Figure 4: The dimensional variable of character candidates

Moreover, since the manufacturing time of EBL is dominantly determined by electronic beam shooting, in our work, we make use of total number of shots as the measurement of projection time. Suppose each candidate c_i is referred r_i^c times in the chip. For each of its appearance, the candidate c_i will be projected by either CP or VSB method, with a number of shots n_i^{CP} and n_i^{VSB}. The total processing time (number of shots) of the entire circuit is computed by following equation.

$$\sum_{c_i \in C^{CP}} r_i^c n_i^{CP} + \sum_{c_i \in (C^C \setminus C^{CP})} r_i^c n_i^{VSB} \quad (1)$$

C^C is the set of all the character candidates. C^{CP} is the union of selected candidates processed by CP method, which is a subset of C^C.

In our work, for simplification purpose, we only design and optimize the stencil for single design. The general case of multiple chips can be easily extended, where the characters would be reused by different designs. Based on above description, our optimization problem can be stated as below:

Problem Formulation: Given a design and its set of character candidate C^C, select a subset C^{CP} out of C^C as characters, and place them on the stencil S. The objective is to minimize the total projection time (number of shots) of this design expressed by Equation (1), while the placement of C^{CP} is bounded by the outline of S. The width and height of stencil is W and H, respectively, and all the candidate has unique width w and height h. The maximum overlapping margin between adjacent characters is given by o_{ij}^H and o_{ij}^V.

In this paper, we will first investigate on the special case of one dimensional stencil design in Section 3, where the amount of blanking spaces differs only in either horizontal or vertical direction. Then, in Section 4, the algorithm, for generalized two dimensional problem, will be developed.

3. ONE DIMENSIONAL STENCIL DESIGN

Normally, each template implements one standard cell. That is to say, the enclosed circuit patterns of all the characters have the same height, and their layouts near top and bottom boundary edges are mostly regular power rails. As a result, illustrated by Fig. 5 (a), the required blanking spaces on the top t and bottom b are nearly identical for these candidates.

Therefore, in such case, characters are usually be placed on the stencil in a row-based manner, shown by Fig. 5 (b). All rows have a unique height h. The overlapped blanking margin h_o between adjacent rows are also the same, which is $\min(t_i, b_i)$. In consequence, as Fig. 5 (c) shows, the overlapping-aware stencil design becomes a one-dimensional problem. The number of character rows can be pre determined as $\lfloor (H - h_o)/(h - h_o) \rfloor$. The candidates would be packed into these rows with maximum width W.

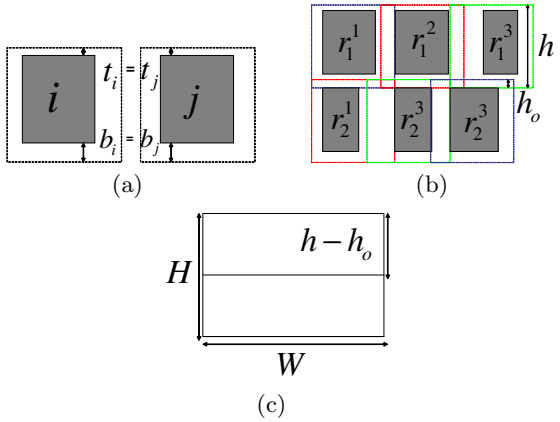


Figure 5: One-dimensional Stencil Design.

The overview of our four-phase iterative refinement algorithm for this special one-dimension problem is given in Fig. 6, and the details will be discussed in following subsections.

3.1 Greedy One Dimensional Bin Packing

To construct a reasonable good starting point, we adopt a descending best-fit bin packing algorithm to push the character candidates into stencil, until there is no enough capacity.

Note that the overall projection time (number of shots) of Objective (1) can also be represented as

$$\sum_{c_i \in C^C} r_i^c n_i^{VSB} - \sum_{c_i \in C^{CP}} r_i^c (n_i^{VSB} - n_i^{CP}) \qquad (2)$$

where the first part is independent with stencil design. To reduce the processing time, $\sum_{c_i \in C^{CP}} r_i^c (n_i^{VSB} - n_i^{CP})$ should be made as large as possible, during greedy bin-packing.

Therefore, as preprocessing, we first assign each candidate c_i a profit value p_i, $r_i^c(n_i^{VSB} - n_i^{CP})$. The bigger p_i is, the larger amount of projection efforts can be saved by printing c_i using CP than VSB method. For getting good greedy optimization result, the c_i with larger profit should be given higher priority to be placed on the stencil. Guided by this heuristic, in the second step, the character candidates, which have not been on the stencil yet, will be sorted decreasingly based on their profits and packed in a sequential manner.

Next, these sorted candidates will be pushed into stencil by a best-fit packing strategy. When c_i is to be packed, the row, which has the most amount of capacities left *after* accommodating c_i, will be picked. This is to consider the possible shared space between adjacent objects, when we are computing the remaining room in each row. As Fig. 7 (a) illustrates, suppose only two rows are available and candidate C is to be packed next. It appears that row R1 has more capacity left. However, as Fig. 7 (b) illustrates, when we try out C in both rows, it is R2 which has larger remaining room. As a result, candidate C is packed into R2, shown by Fig. 7 (c).

3.2 Single Row Reordering

After greedy bin packing, there is no room left to accommodate more candidates. However, as motivated by Fig. 3, we can adjust the relative locations of already-placed characters in each row to shrink its occupied width and increase remaining capacity. This allows pushing in more candidates, which further reducing the overall projection time. Therefore, in this phase, our goal is to minimize the total width of its characters in each row for maximizing remaining capacity.

Suppose row r contains a set of c_0^r .... c_n^r characters from left

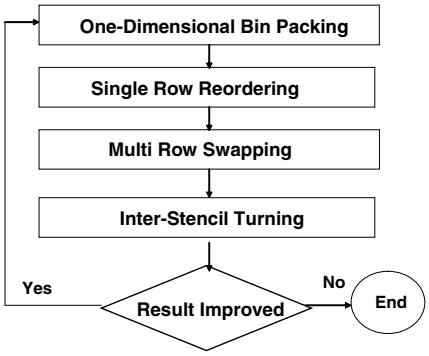


Figure 6: The overview of one dimensional stencil design with overlapped characters.

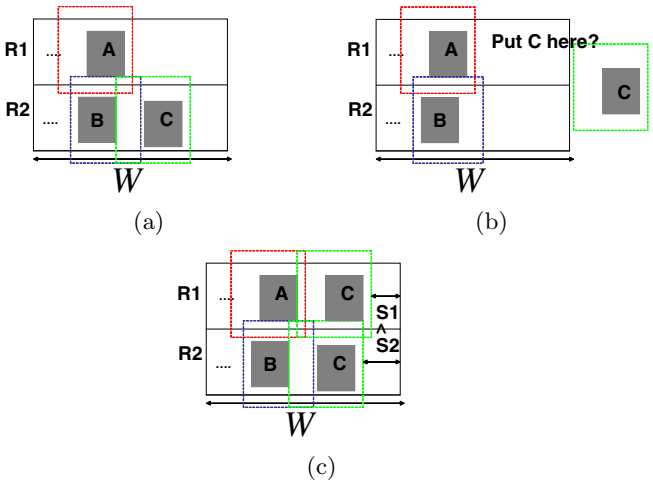


Figure 7: This figure illustrates the procedure of best-fit bin packing with overlapping awareness.

to right, its total occupied width can be computed as $\sum_{i=0}^{n} w - \sum_{i=0}^{n-1} o_{i,i+1}^H$. It is not difficult to see that $\sum_{i=0}^{n} w$ is a constant as long as the number of characters is not changed. Therefore, to minimize the total occupied width, the overall overlapped blanking margin $\sum_{i=0}^{n-1} o_{i,i+1}^H$ should be maximized.

To compute optimal character permutation for maximum amount of shared blanking width, we formulate a *minimum cost Hamiltonian path* problem. First of all, a graph G is constructed as follows: Each c_i^r is represented by a vertex v_i^r. For each pair of v_i^r and v_j^r, we add two directed edges e_{ij} and e_{ji}. The associated costs are $o_{big}^H - o_{ij}^H$ and $o_{big}^H - o_{ji}^H$, respectively. o_{ij}^H / o_{ij}^H is the shared space when c_i is put left/right adjacent to c_j, and o_{big}^H is a constant value, bigger than any of o_{ij}^H. To maximize $\sum_{i=0}^{n-1} o_{i,i+1}^H$, it suffice to find a path visiting each node of G exactly once such that the total edge weighs $(\sum_{e \in Path}(o_{big}^H - o_{ij}^H))$ along this path is minimized. As Fig. 8 (a) illustrates, a graph for three character placement (A,B,C) is given. Suppose the minimum cost Hamiltonian path is found as Fig. 8 (b), Fig. 8 (c) shows its corresponding character placement.

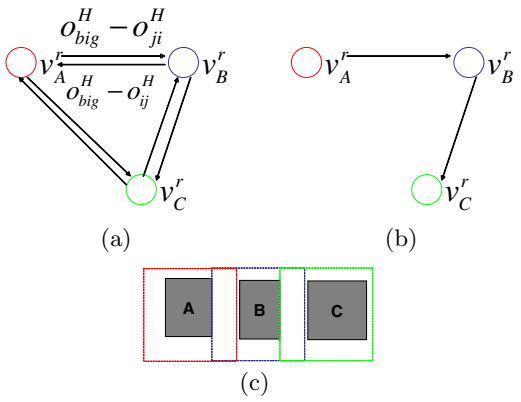


Figure 8: This figure shows how to optimize the occupied-width of each row as min-cost Hamiltonian path problem

Practically, since the problem of minimum cost Hamiltonian path is NP-hard, it may be expensive to solve the whole row in one time. In that case, our heuristic is to partition the row into

multiple overlapped smaller segments, and solve each segment by Hamiltonian path based method.

3.3 Multiple Row Swapping

After single row reordering, the character permutation within each row has been extensively optimized. However, it is still possible to increase their remaining capacities, by swapping characters from different rows. As Fig 9 illustrates, by swapping r_1^2 and r_2^2, both characters find "better" neighbors with more overlapped blanking space. For row R1 and R2, their remaining rooms are both increased.

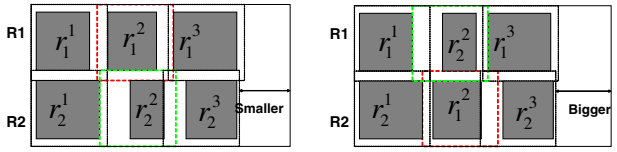


Figure 9: This figure explains the motivation of multi-row swapping.

The algorithm is briefly explained as follows. We test every pair of characters from different rows. Only when the remaining capacities of both rows are increased after swapping, it is considered as a *reasonable* swap. This ensures, the modified placement is definitely better than original one. The reason is that, after swapping, if one row gains more room but another has less, it is possible that the following optimization is hurt by the row with shrunk capacity.

After all the *reasonable* swap pairs are found, they are sorted increasingly by capacity gains, and performed one by one. When certain swap is done, the associated characters and their neighbors are locked. Any swap in the later trials is not allowed to move these locked characters as well as their neighbors. This honors previous optimization result.

3.4 Inter Stencil Tuning

The previous single and multiple row optimization are conducted based on the initial solution of bin-packing algorithm in Section 3.1. This may limit the optimization space. To get out of local optima, as the last step of ech iteration, we would like to exchange the placed characters with those which have not been selected.

Our approach is to randomly pick and exchange two character candidates, where one is from the stencil and another is not. The swapping will be accepted, only if the overall projection time, number of shots, Objective (1) is reduced and the remaining capacity of any row is not shrunk.

4. TWO DIMENSIONAL STENCIL DESIGN

In this section, we investigate on the general case of EBL stencil design with overlapped characters. The blanking spaces of templates are non-uniform along both horizontal and vertical directions. Due to NP-completeness of this problem, we adopt a simulated-annealing based heuristic approach to perform a robust iterative improvement.

4.1 Sequence Pair Representation

To represent the character placement solution, we make use of sequence pair (SP) proposed in [16].

Given a set of character candidates C^C, its SP consists in two permutations $\overline{X}\&\overline{Y}$ of these templates $(c_0, c_1...c_n)$, which specifies their geometry relationships as below.

$$(\overline{X} :< .., c_i..c_j.. >, \overline{Y} :< ...c_i...c_j... >) : c_i \text{ is left to } c_j \quad (3)$$

$$(\overline{X} :< .., c_j..c_i.. >, \overline{Y} :< ...c_i...c_j... >) : c_i \text{ is below } c_j \quad (4)$$

Based these constraints, we can map any SP into a solution of character placement as following procedure:

Procedure 1:

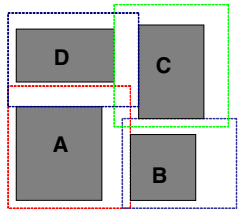
Step1: Compute a packing solution of C^C, following similar methods of [16, 17]. The details will be described in Section 4.1.1 and 4.1.2.

Step2: Assuming the left-bottom coordinates of packing results and stencil are the same, the candidates, which are located completely within the outline of stencil, are considered as selected characters. □

The step1 is the critical one in above transformation. Due to specific properties of our problem, its implementation actually differs from the conventional approaches of [16, 17], explained as follows.

4.1.1 Correct Packing Algorithm

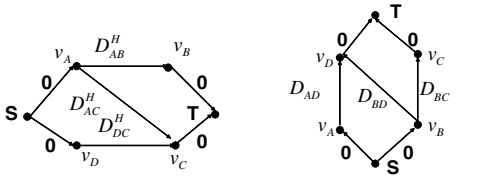
The key step of packing solution evaluation from SP is to determine the physical coordinates of each block. This problem has been well investigated, when overlapping is not considered between adjacent blocks. The original algorithm is proposed in [16], and improved by [17] with new solution pruning technique. The work of [16] is extensible for our overlapping-enabled character placement problem. However, the key speed-up idea in [17] does not apply, although it is much faster.



$SP: \overline{X} = (D\ A\ C\ B)$

$\overline{Y} = (A\ B\ D\ C)$

(a)



$$D^H_{AB} = \frac{1}{2}w_A + \frac{1}{2}w_B - o^H_{A,B} \qquad D^V_{BC} = \frac{1}{2}h_B + \frac{1}{2}h_C - o^V_{B,C}$$

$$D^H_{AC} = \frac{1}{2}w_A + \frac{1}{2}w_C - o^H_{A,C} \qquad D^V_{BD} = \frac{1}{2}h_B + \frac{1}{2}h_D - o^V_{B,D}$$

$$D^H_{DC} = \frac{1}{2}w_D + \frac{1}{2}w_C - o^H_{D,C} \qquad D^V_{AD} = \frac{1}{2}h_A + \frac{1}{2}h_D - o^V_{A,D}$$

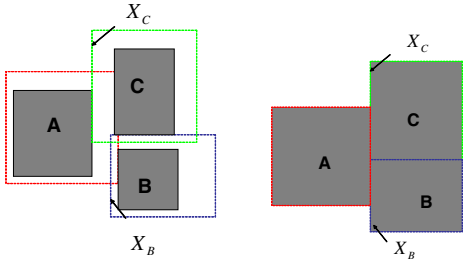
(b) H graph (c) V graph

Figure 10: This figure explains packing evaluation of [16] based on sequence pair

The method of [16] is based on longest path algorithm, and starts from constraint graph construction. Given a SP, a H/V graph is built first to capture the horizontal/vertical relationship between different blocks. Assume there are totally C^C candidates, the H/V graph has $|C^C|+2$ vertexes, one v_i for

each candidate c_i plus a source s and sink t. If c_j is (left adjacent to)/(below) c_k, a directed edge e_{jk} is added from v_j to v_k. The weight of e_{jk} is the minimum possible horizontal/vertical distance between the centers of c_j and c_k. Beside these, there is a zero-weight edge from source to every v_i, and a zero-weight edge from every v_i to sink. For the example of Figure 10 (a), Figure 10 (b) and (c) show the resulting H and V constraint graphs, respectively.

After that, the x/y coordinates of these candidates can be obtained by finding its weighted longest path algorithm from source. As easy to see, this methodology is also applicable for our problem, where overlapped space is allowed between adjacent vertexes. The only difference is that, when the weights of edge are assigned, the amount of shared blanking space must be considered, as highlighted by the red cycles in Figure 10 (b) and (c).



(a) non-overlapping (b) overlapping

$SP: \overline{X} = (D\ A\ C\ B) \quad \overline{Y} = (A\ B\ D\ C)$

Figure 11: This figure illustrates the key idea of of [17]

On the other side, the work of [17] does not explicitly build the constraints graphs but depends on the longest common subsequence computations. They evaluate the placement of character candidates much faster than [16], depending on the following property.

Property 1 *Given two blocks B and C, if we put them (right adjacent to)/below a common component A, then the x/y coordinates of these two blocks should be same.*

The correctness of this property can be easily seen for the conventional packing, as shown by Figure 11 (a), while overlapping is not considered. However, it does not hold true, when the sharing of characters becomes possible. As Figure 11 (b) illustrates, due to different overlapping margins, the coordinates of B and C are not the same.

4.1.2 Fast Packing Evaluation

After evaluating packing solution, in the step2 of Procedure 1, the candidates outside the outline of stencil will not be taken as characters. This implies, the detailed locations of these candidates are not important, and do not have to be computed in the step1. Great speedup can be achieved by making use of this property.

In detail, in the implementation of SP-based minimum area packing, we stop placement evaluation as soon as the contour of already-packed character candidates is completely outside the outline of stencil by at least a margin of o_{max}, given that o_{max} is the maximum value of o_{ij}^H and o_{ij}^V.

This strategy will not effect the solution of character placement. For any of unpacked candidates by the stopping time, it can not be totally fit into the stencil no matter how to push it around the boundaries of already-packed character clusters.

4.2 Simulated Annealing

During simulated annealing, we continuously make small modification on sequence pair, and evaluate the resulting stencil design. The new SP/solution will be for sure adopted if reducing the total time of Objective (1). While it is actually a worse character placement, this non-improving result is accepted with probability decreasing over time.

In this subsection, we present two effective SP perturbation methods for better local search towards shorter projection time: throughput-driven swapping and slack-based insertion.

4.2.1 Throughput-driven Swapping

The first type of perturbation we perform is throughput-driven swapping. The basic idea is to try reducing overall projection time by swapping the positions of two candidates in the $\overline{X}$&$\overline{Y}$ SP. This is equivalent to exchange their relative locations in the packing solution.

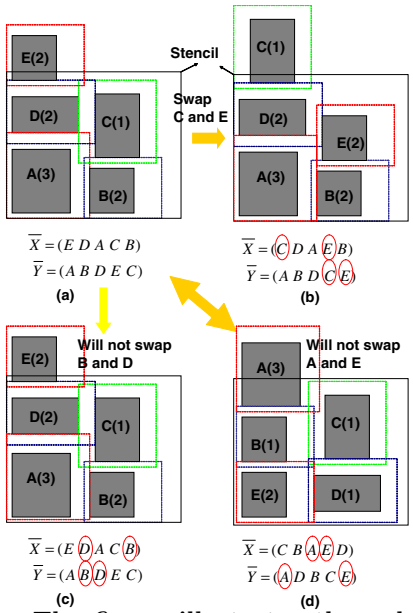


Figure 12: The figure illustrates throughput-driven swapping.

Fig. 12 illustrates a motivational example, which has five blocks A-E to be packed. The required number of shots, to project any of these candidates once, are assumed as 1 and 10 for CP (n_i^{CP})and VSB (n_i^{VSB}) methods respectively. The digit in the parentheses denotes how many times r_i of each comp<onent, will be used and printed in the design.

Fig. 12 (a) gives a SP representation and its corresponding stencil design, based on the Procedure 1 in Section 4.1. Following the definition of Objective 1, the total processing time (number of shots) are $3+2+1+2+10\times 2 = 28$, since A-D are selected as characters while E is not. If swapping the locations of C and E in SP as Fig. 12 (b), we would end up with a better stencil design with less amount processing time. It only takes a number of 19 shots, which is computed as $3+2+10+2+2 = 19$, in this case.

In the detailed implementation, we enforce two heuristic swapping constraints, to enable efficient and effective shot number reduction.

First of all, given a SP, out of the pair of elements to be changed, we require that one candidate c_s, should have been selected as characters by its corresponding stencil packing, while the other one, c_o is not. For the example of Fig. 12 (a),

we only allow the exchange of the positions between E and any of A-D. The swapping among any two of A-D is not enabled. The reason is that if the two candidates to be swapped are both in or out of stencil already, most likely the new SP generates a stencil solution with same set of selected characters and just different geometrical ordering. As an example, if we swap candidate B and D< which are both already in the stencil, like from Fig. 12 (a) to Fig. 12 (c), the resulting packing result also selects A-D as characters, still requiring 28 shots totally.

Secondly, after randomly picking in-stencil candidate c_s and out-of-stencil one c_o for swapping, we will compute the difference of their profits p_o-p_s, to decide whether this swapping would be tried on. The profit p_o/p_s is defined as same as $r_i(n_i^{VSB} - n_i^{CP})$ in Section 3.1, which reflects the reduction of the shoot number by printing this candidate by CP rather than VSB. If we swap the locations of c_s and c_o, it is highly like that c_s will be pushed out of stencil but the c_o would be selected as character in turn. Assuming all the other candidates stay either in or outside the stencil, as the state before the swapping, the total shot reduction by this exchange can be approximated as p_o-p_s. Therefore, if the difference p_o-p_s is smaller than zero, it is in high possibility that the swapping under consideration will not lead to better packing result. For the example of Fig. 12 (a), suppose c_s and c_o are A and E, respectively, and it turns out p_o-p_s is -9. In this case, the corresponding stencil design indeed becomes worse, taking 35 shots as Fig. 12 (d) shows.

4.2.2 Slack-based insertion

Given a SP and its corresponding character solution, our purpose of slack-based insertion is to add-in a new candidate, which currently is not serving as character, into the stencil. To ensure robust throughput improvement, we would like to find a good strategy to insert such extra candidate, so that all the previously already-placed characters are still kept on the stencil in most trials. This equals to increase the number of usable templates. In this subsection, we make use of the concept of slack, applied in [18], to search such a good insertion location.

Given a character c_s on the stencil, its x/y slack is defined as the allowed movement range of x/y coordinates of c_s, under the constraint that none of all the other already-placed characters would be pushed outside the stencil after such move. Fig. 13 (a)-(b) illustrate a simple example, with four characters A-D. Their leftmost and rightmost packing solutions are shown by Fig. 13 (a) and (b), respectively. Based on this two extreme cases, the x slack of C, for example, can be computed as $X_c^{right} - X_c^{left}$.

Once slacks are known, we randomly pick a *base* character c_b, which has large slacks in both x and y directions, and insert a new candidate c_{new} before it. The reason is that the location of such *base* can be moved in relatively big amount to make space for additional character. In terms of SP operation, this can be done by simply changing the position of c_{new} right before c_b in $\overline{X}$ and $\overline{Y}$ permutations. As illustrated by Fig. 13 (c), suppose the c_b and c_{new} are candidate C and E, respectively. The resulting new SP is obtained by insert E right in front of the position of C, as shown by Fig. 13 (d) .

5. EXPERIMENTAL RESULTS

We implement our algorithm in C++ and test on Intel Core 3.0GHz Linux machine with 32G RAM. LKH [19] is chosen as the solver for min-cost Hamilton path. Moreover, Parquet [18] is adopted as our simulated annealing framework.

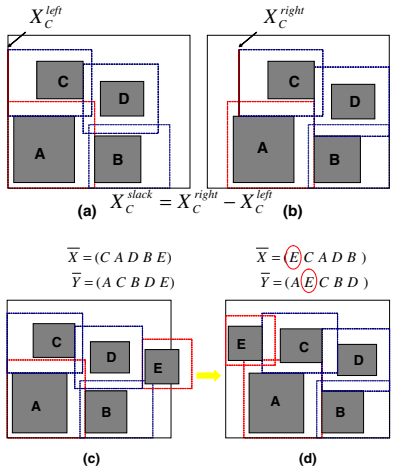


Figure 13: The simple example of slack-based insertion.

Table 1: Statistics on testcases.

ckts	csize	total area	total blanks	optimal area
1D-1	3.8x3.8	1.444	0.416	1.028
1D-2	4.0x4.0	1.6	0.479	1.121
1D-3	4.2x4.2	1.764	0.514	1.25
1D-4	4.4x4.4	1.936	0.569	1.367
2D-1	3.8x3.8	1.444	0.414	1.03
2D-2	4.0x4.0	1.6	0.529	1.071
2D-3	4.2x4.2	1.764	0.662	1.102
2D-4	4.4x4.4	1.936	0.774	1.162

To test the efficiency of proposed methods, we randomly generate eight benchmarks. The size of stencil is set as 100um x 100um, and a total number of 1000 character candidates with unique size are generated. The sharable blanking area within each candidate is randomly decided, uniformly distributed between 0%-50% character width. For the special case of one dimensional problem, the blanking space along vertical direction is set as a constant value. Moreover, for each candidate c_i, we randomly assign a triple of value (r_i, n_i^{VSB}, n_i^{CP}) as its referred time in chip, and respective number of shots by VSB and CP. n_i^{VSB} is made 5-10x larger than n_i^{CP}.

The detailed statistical data for individual testcase is shown in Table 1. The first column denotes the name of benchmarks, which "1D-x" and "2D-x" are applied for one and two dimensional problem, respectively. "csize" is the size of each character candidate, formatted by "um x um". The units of all the other columns are "$1e^4 um^2$". "total area" shows the total area of all the character candidates, and "total blanks" is the summation of their sharable blanking space. "optimal area" is computed as "total area" minus "total blanks", typically larger than the area of given stencil. This matches the fact that even under best possible case of stencil design, where all the blanking area are indeed shared by adjacent characters, the entire set of the candidates can not be fully pushed into the stencil.

For comparative reason, we implement two different stencil design approaches. The first one **NO-OVERLAP** is based on the work of [12], where no overlapped characters is allowed. A little difference is that, in its implementation, only one stencil with unique character size is considered. Moreover, for our problem, their algorithm is somewhat degenerated into a method of selecting the most profitable candidates, which profit is judged by $r_i(n_i^{CP} - n_i^{VSB})$. In the second comparative approach **GREEDY**, possible sharing is taken into account, but a greedy methodology is applied to chose character candidates. In 1D problem, only the first phase of heuristic Descending Best-Fit (DBF) packing in Section 3.1 is performed.

For two-dimensional problem, 2D DBF packing is conducted.

5.1 One-dimensional Stencil Design

Table 2 lists the comparison of stencil design in one-dimensional case. "#shot" shows the total processing time (number of shots) of the circuit by using corresponding stencil design methodologies, which is computed by the equation of Objective 1. "#char" is the number of characters that fits into stencils, and "$\#CPU$" tells the runtime of these stencil optimization methods, in terms of seconds.

As we can see, compared to **NO-OVERLAP**, we are able to averagely put 42% more characters on the stencil, and reduce the total projection time (number of shots) by 51%. With respect to **GREEDY** algorithm, our approach still achieves averagely 14% more projection time reduction, by allowing 7% more characters placed. The CPU time of our approach is relatively large but its absolute value is only around 20s. These results show the effectiveness and efficiency of our proposed four-phase iterative refinement algorithm.

For this special one-dimension problem, **GREEDY** looks also quite useful. The reason is that the vertical blanking spaces of these candidates are uniform in this case, and have been fully shared during the stencil design.

Table 2: Result Comparison for 1D problem

	NO-OVERLAP			GREEDY			our approach		
ckts	#shot	#char	$CPU(s)$	#shot	#char	CPU	#shot	#char	CPU
1D-1	28654	676	1.2	13528	901	2.2	10083	951	22.3
1D-2	41727	625	1.1	17929	836	2.1	14921	880	21.8
1D-3	38460	529	0.9	25155	727	1.9	22503	768	20.6
1D-4	41260	484	0.8	29462	665	1.8	26756	702	20.1
total	150101	2314	4	86074	3129	8	74263	3301	84.8
ratio	2.0	1	0.05	1.16	1.35	0.10	1	1.42	1

5.2 Two-dimensional Stencil Design

Table 3 lists the comparison of stencil design in general two-dimensional case. The meaning of labels are the same as Table 2. Compared to **NO-OVERLAP** and **GREEDY** methods, in average, our proposed SP-based algorithm places 28% and 24% more characters on stencil, which reduces the projection time (number of shots) by 31% and 25%, respectively. The **GREEDY** algorithm does not work that well in this 2D problem, because the blanking area varies in both horizontal and vertical directions and the native first-bin-best-fit packing very easily get stuck in local optima.

Due to two-dimensional optimization, the runtime of our approach is much longer than 1D problem, comparatively. It takes a few hundred seconds, but is still satisfactory. The design of stencil is only a one-time process before projecting large volume of chips by EBL. Several minutes preprocessing time is relatively very tiny in the whole manufacturing procedure.

Table 3: Result Comparison for 2D problem

	NO-OVERLAP			GREEDY			our approach		
ckts	#shot	#char	CPU	#shot	#char	CPU	#shot	#char	CPU
2D-1	23319	676	1.3	26832	625	2.3	16877	803	466
2D-2	29368	576	1	25977	642	2.6	20141	750	447
2D-3	32399	526	0.9	30411	558	2.5	$<$ 23850	688	424
2D-4	35410	474	0.8	31930	531	2.7	25278	660	416
total	120496	2252	4	115150	2356	10.1	86146	2901	1755
ratio	1.40	1	0.002	1.33	1.05	0.006	1	1.30	1

6. CONCLUSION

In this paper, we have developed two algorithms for overlapping aware stencil design in electronic beam lithography. The experimental results show 51% reduction on the total projection time, compared to the conventional design when the characters are not overlapped.

7. ACKNOWLEDGEMENT

The authors would also like to thank Dr. Gi-Joon Nam at IBM Austin Research for helpful discussions on this problem.

8. REFERENCES

[1] Andrew B. Kahng, Chul-Hong Park, Xu Xu, and Hailong Yao. Layout decomposition for double patterning lithography. In *Proc. Int. Conf. on Computer Aided Design*, November 2008.

[2] Kun Yuan, Jae-Seok Yang, and David Z. Pan. Double patterning layout decomposition for simultaneous conflict and stitch minimization. In *Proc. Int. Symp. on Physical Design*, March 2009.

[3] Jae-Seok Yang, Katria Lu, Minsik Cho, Kun Yuan, and David Z. Pan. A New Graph-theoretic, Multi-objective Layout Decomposition Framework for Double Patterning Lithography. In *Proc. Asia and South Pacific Design Automation Conf.*, Janunary 2010.

[4] Aki Fujimur. Beyond Light: The Growing Importance of E-Beam. In *Proc. Int. Conf. on Computer Aided Design*, November 2009.

[5] Aki Fujimur. Design for E-Beam: Getting the Best Wafers Without the Exploding Mask Costs. In *Proc. Int. Symp. on Quality Electronic Design*, March 2010.

[6] Akira Fujimura, Takashi Mitsuhashi, Kenji Yoshida, Shohei Matsushita, Larry Lam Chau, Tam Dinh Thanh Nguyen, and Donald MacMillen. Stencil Design and Method for Improving Character Density for Cell Projection Charged Particle Beam Lithography. In *US Patent*, Jan. 2010.

[7] Hans Co Pfeiffer. New Prospects for Electron Beams as Tools for Semiconductor Lithography. In *Proc. of SPIE*, May 2009.

[8] Takeshi Fujino, Yoshihiko Kajiya, and Masaya Yoshikawa. Character-build standard-cell layout technique for high-throughput character-projection EB lithography . In *Proc. of SPIE*, July 2005.

[9] Makoto Sugihara, Taiga Takata, Kenta Nakamura, Yusuke Matsunaga, and Kazuaki Murakami. A CP mask development methodology for MCC systems. In *Proc. of SPIE*, May 2006.

[10] Makoto Sugihara, Kenta Nakamura, Yusuke Matsunaga, and Kazuaki Murakami. CP mask optimization for enhancing the throughput of MCC systems. In *Proc. of SPIE*, Octerber 2005.

[11] Yusuke Matsunaga Makoto Sugihara and Kazuaki Murakami. Technology mapping technique for enhancing throughput of multi-column-cell systems. In *Proc. of SPIE*, March 2007.

[12] Makoto Sugihara. Optimal character-size exploration for increasing throughput of MCC lithographic systems. In *Proc. of SPIE*, Feb. 2009.

[13] Lin Xie, Azadeh Davoodi, and Kewal K. Saluja. Post-silicon diagnosis of segments of failing speedpaths due to manufacturing variation. In *Proc. Design Automation Conf.*, June 2010.

[14] Sean Shi and David Pan. Wire Sizing and Shaping with Scattering Effect for Nanoscale Interconnection. In *Proc. Asia and South Pacific Design Automation Conf.*, Jan 2006.

[15] Yongchan Ban, Savithri Sundareswaran, and David Z. Pan. Total Sensitivity Based Standard Cell Layout Optimization. In *International Symposium on Physical Design (ISPD)*, 2010.

[16] Hiroshi Murata, Kunihiro Fujiyoshi, Shigetoshi Nakatake, and Yoji Kajitani. VLSI Module Placement Based on Rectangle-Packing by theSequence-Pair. In *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, Decemember 1996.

[17] Xiaoping Tang, Ruiqi Tian, and Martin Wong. Fast Evaluation of Sequence Pair in Block placement by Longest Common Subsequence Computation. In *Proc. Design, Automation and Test in Eurpoe*, March 2000.

[18] Saurabh H. Adya and Igor L. Markov. Fixed-outline Floorplanning : Enabling Hierarchical Design. In *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, Decemember 2003.

[19] http://www.akira.ruc.dk/∼keld/research/LKH.