# E-BLOW: E-Beam Lithography Overlapping aware Stencil Planning for MCC System

Bei Yu,  Kun Yuan[†],  Jhih-Rong Gao,  David Z. Pan

ECE Department, Univ. of Texas at Austin, Austin, TX, USA
[†]Cadence Design Systems, Inc., San Jose, CA, USA
{bei, jrgao, dpan}@cerc.utexas.edu

## ABSTRACT

Electron beam lithography (EBL) is a promising maskless solution for the technology beyond 14nm logic node. To overcome its throughput limitation, recently the traditional EBL system is extended into MCC system. In this paper, we present E-BLOW, a tool to solve the overlapping aware stencil planning (OSP) problems in MCC system. E-BLOW is integrated with several novel speedup techniques, i.e., successive relaxation, dynamic programming and KD-Tree based clustering, to achieve a good performance in terms of runtime and solution quality. Experimental results show that, compared with previous works, E-BLOW demonstrates better performance for both conventional EBL system and MCC system.

## Categories and Subject Descriptors

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

## General Terms

Algorithms, Design, Performance

## Keywords

Electron Beam Lithography (EBL), Overlapping aware Stencil Planning (OSP), Multi-Column Cell (MCC) System

## 1. INTRODUCTION

As the minimum feature size continues to scale to sub-22nm, the conventional 193nm optical photolithography technology is facing great challenge in manufacturing [1]. In the near future, double/multiple patterning lithography (DPL /MPL) has become one of viable lithography techniques for 22nm and 14nm logic node [2–4] . In the longer future, i.e., for the logic node beyond 14nm, extreme ultra violet (EUV) and electric beam lithography (EBL) are promising candidates for lithographic processes. However, EUV suffers from the delay due to the tremendous technical barriers such as lack of power sources, resists, and defect-free masks [5].
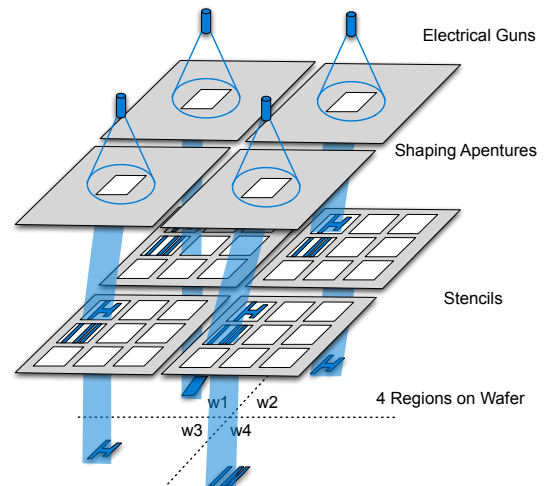
**Figure 1: Printing process of MCC system.**

EBL system, on the other hand, has been developed for several decades [6]. One of the conventional EBL systems is based on character projection (CP) mode. Some complex shapes, called *characters*, are prepared on the stencil. The key idea is that if a pattern is pre-designed on the stencil, it can be printed in one electronic shot, otherwise it needs to be fractured into a set of rectangles and printed one by one through variable shaped beam (VSB). Compared with purely VSB mode, in the CP mode the throughput can be improved significantly. Compared with the traditional lithographic methodologies, EBL has several advantages. (1) Electron beam can be easily focused into nanometer diameter with charged particle beam, which can avoid suffering from the diffraction limitation of light. (2) The price of a photomask set is getting unaffordable. As a maskless technology, EBL can reduce the manufacturing cost. (3) EBL allows a great flexibility for fast turnaround times and even late design modifications to correct or adapt a given chip layout. Because of all these advantages, EBL is being used in mask making, small volume LSI production, and R&D to develop the technological nodes ahead of mass production.

Even with decades of development, the key limitation of the EBL system has been and still is the low throughput. Recently, multi-column cell (MCC) system is proposed as an extension of conventional EBL system, where several independent character projections (CP) are used to further speed-up the writing process [7, 8]. Each CP is applied on one section of wafer, and all CPs can work parallelly to achieve better throughput. Due to the design complex-
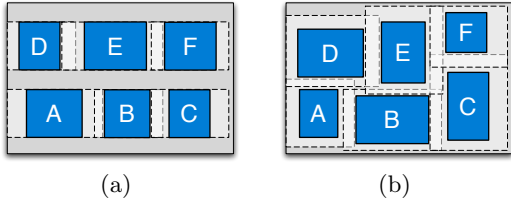
**Figure 2: (a) 1D-OSP. (b) 2D-OSP.**

ity and cost consideration, different CPs share one stencil design [9]. One example of MCC printing process is illustrated in Fig. 1, where four CPs are bundled to generate a MCC system. The whole wafer is divided into four regions, $w_1, w_2, w_3$ and $w_4$, and each region is printed through one CP. Note that the whole writing time of the MCC system is determined by the maximum writing time of the four regions. For modern design, because of the numerous distinct circuit patterns, only limited number of patterns can be employed on stencil. Therefore, the area constraint of stencil is the bottleneck. To improve the throughput, the stencil should be carefully designed/manufactured to contain the most repeated cells or patterns.

Much previous works focus on the design optimization for conventional EBL system [10–14]. Stencil planning, which is one of the most challenges, has earned much attentions. When blank overlapping is not considered, the stencil planning can be formulated as a character selection problem, where integer linear programming (ILP) was applied to select group of characters for throughput maximization [10]. Recently, Yuan et al. in [12] investigated on the overlapping aware stencil planning (OSP) problem.

However, no existing stencil planning work has done for the MCC system. Compared with conventional EBL system, MCC system introduces two main challenges. First, the objective is new: in MCC system the wafer is divided into several regions, and each region is written by one CP. Therefore the new OSP should minimize the maximal writing times of all regions. While in conventional EBL system, the objective is simply minimize the wafer writing time. Besides, the stencil for an MCC system can contain more than 4000 characters, previous methodologies for EBL system may suffer from runtime penalty.

The OSP problem can be divided into two sub-problems: *1D-OSP* and *2D-OSP* [12]. When standard cells with same height are selected into stencil, the problem is referred as 1D-OSP. As shown in Fig. 2(a), each character implements one standard cell, and the enclosed circuit patterns of all the characters have the same height. Note that here we only show the horizontal blanks, and the vertical blanks are not represented because they are identical. In 2D-OSP, the blanking spaces of characters are non-uniform along both horizontal and vertical directions. By this way, stencil can contain both complex via patterns and regular wires. Fig. 2(b) illustrates a stencil design example for 2D-OSP.

This paper presents E-BLOW, the first study for OSP problem in MCC system. E-BLOW integrates several novel techniques to achieve near-optimal solution in reasonable time. The main contributions of this paper are stated as follows: (1) First study for stencil planning problem in MCC system. (2) Shows that OSP problem for both EBL and MCC systems are NP-hard. (3) Proposes a simplified formulation for 1D-OSP, and proves its rounding lower bound theoretically. (4) A successive relaxation algorithm to find

a near optimal solution. (5) KD-Tree based clustering algorithm for speedup in 2D-OSP.

The remainder of this paper is organized as follows. Section 2 provides problem formulation. Section 3 presents algorithm details to resolve 1D-OSP problem in E-BLOW, while section 4 details the E-BLOW solutions to 2D-OSP problem. Section 5 reports experimental results, followed by the conclusion in Section 6.

## 2. PROBLEM FORMULATION

### 2.1 OSP Problem Formulation

In an MCC system with $P$ CPs, the whole wafer is divided into $P$ regions $\{w_1, w_2, \ldots, w_P\}$, and each region is written by one particular CP. We assume cell extraction [15] has been resolved first. In other words, a set of character candidates $C^C = \{c_1, \cdots, c_n\}$ has already been given to the MCC system. For each character candidate $c_i \in C^C$, its writing time through VSB mode is denoted as $n_i$, while its writing time through CP mode is 1.

The regions of wafer have different layout patterns, and the throughputs would be also different. Suppose character candidate $c_i$ repeats $t_{ic}$ times on region $w_c$. Let $a_i$ indicate selection of character candidate $c_i$ as follows.

$$a_i = \begin{cases} 1, & \text{candidate } c_i \text{ is selected on stencil} \\ 0, & \text{otherwise} \end{cases}$$

If $c_i$ is prepared on stencil, the total writing time of patterns $c_i$ on region $w_c$ is $t_{ic} \cdot 1$. Otherwise, $c_i$ should be printed through VSB. Since region $w_c$ comprises $t_{ic}$ candidate $c_i$, the writing time would be $t_{ic} \cdot n_i$. Therefore, for region $w_c$ the total writing time $T_c$ is as follows:

$$
\begin{aligned}
T_c &= \sum_{i=1}^{n} a_i \cdot (t_{ic} \cdot 1) + \sum_{i=1}^{n} (1 - a_i) \cdot (t_{ic} \cdot n_i) \\
&= \sum_{i=1}^{n} t_{ic} \cdot n_i - \sum_{i=1}^{n} t_{ic} \cdot (n_i - 1) \cdot a_i = T_c^{VSB} - \sum_{i=1}^{n} R_{ic} \cdot a_i
\end{aligned}
$$

where $T_c^{VSB} = \sum_{i=1}^{n} t_{ic} \cdot n_i$, and $R_{ic} = t_{ic} \cdot (n_i - 1)$. $T_c^{VSB}$ represents the writing time on $w_c$ when only VSB is applied, and $R_{ic}$ can be viewed as the writing time reduction of candidate $c_i$ on region $w_c$. In MCC system, both $T_c^{VSB}$ and $R_{ic}$ are constants. Therefore, the total writing time of the MCC system is formulated as follows:

$$
\begin{aligned}
T_{total} &= \max\{T_c\} \\
&= \max\{T_c^{VSB} - \sum_{i=1}^{n} R_{ic} \cdot a_i\}, \forall c \in P \qquad (1)
\end{aligned}
$$

**Problem 1.** *Overlapping aware Stencil Planning (OSP) for MCC system: Given a set of character candidate $C^C$, select a subset $C^{CP}$ out of $C^C$ as characters, and place them on the stencil. The objective is to minimize the total writing time $T_{total}$ expressed by (1), while the placement of $C^{CP}$ is bounded by the outline of stencil. The width and height of stencil is $W$ and $H$, respectively.*

For convenience, we use the term OSP to refer OSP for MCC system in the rest of this paper.

### 2.2 NP-Hardness

**Lemma 1.** *1D-OSP problem is NP-hard.*

Let us consider a special and simper case of 1D-OSP, where each candidate $c_i$ has zero blank space, and CP number is 1. Then the problem can be reduced from a multiple knapsack problem, which is a well known NP-hard problem [16].

**Lemma 2.** *2D-OSP problem is NP-hard.*

Let us consider a special case of 2D-OSP, where each candidate $c_i$ has zero blank space, and CP number is 1. The 2D-OSP problem includes two subproblems: candidate selection and candidate packing. After some candidates are selected on the stencil, the candidates packing problem can be reduced from a strip packing problem [17], which is NP-hard.

Combining Lemma 1 and Lemma 2, we can achieve the conclusion that OSP problem, even for conventional EBL system, is NP-hard.

## 3. E-BLOW FOR 1D-OSP

When each character implements one standard cell, the enclosed circuit patterns of all the characters have the same height. Corresponding OSP problem is called 1D-OSP, which can be viewed as a combination of character selection and single row ordering problems [12]. Different from two heuristic steps proposed in [12], we show that the two problems can be solved simultaneously through ILP formulation (2).

$$\min \quad T_{total} \qquad (2)$$

$$\text{s.t} \quad T_{total} \geq T_c^{VSB} - \sum_{i=1}^{n} (\sum_{k=1}^{M} R_{ic} \cdot a_{ik}), \ \forall c \in P \qquad (2a)$$

$$x_i + w_i \leq W, \qquad \forall i \in N \qquad (2b)$$

$$\sum_k^m a_{ik} \leq 1, \qquad \forall k \in M \qquad (2c)$$

$$x_i + w_{ij} - x_j \leq W(2 + p_{ij} - a_{ik} - a_{jk}) \qquad (2d)$$

$$x_j + w_{ji} - x_i \leq W(3 - p_{ij} - a_{ik} - a_{jk}) \qquad (2e)$$

$$a_{ik}, a_{jk}, p_{ij} : 0 - 1 \text{ variable} \qquad (2f)$$

In (2) $W$ is the width constraint of stencil, $M$ is the number of rows, and $w_i$ is width of character $c_i$. $x_i$ is the x-position of $c_i$. If and only if $c_i$ is assigned to $k$th row, $a_{ik} = 1$. In other words, $a_{ik}$ determines the y-position of $c_i$. $w_{ij} = w_i - o_{ij}^h$ and $w_{ji} = w_i - o_{ji}^h$, where $o_{ij}^h$ is the overlapping when candidates $c_i$ and $c_j$ are packed together. Constraints (2d) (2e) are used to check position relationship between $c_i$ and $c_j$. For $k$th row, it is easy to see that only when $a_{ik} = a_{jk} = 1$, i.e. both character $i$ and character $j$ are assigned to row $j$, then only one of the two constraints (2d) (2e) will be active. If either of them are not assigned to the row, neither of the constraints are active. The number of variables for (2) is $O(N^2)$, where $N$ is the number of character candidates.

Since ILP is a well known NP-hard problem, directly solving it may suffer from long runtime penalty. One straightforward speedup method is to relax ILP (2) into linear programming (LP) as following: replacing constraints (2f) by $0 \leq a_{ik}, a_{jk}, p_{ij} \leq 1$. It is obvious that the LP solution provides a lower bound to the ILP solution. However, we observe that the solution of relaxed LP would be like this: for each $i$, $\sum_j a_{ij} = 1$ and all the $p_{ij}$ are assigned 0.5. Although the objective function is minimized and all the constraints are satisfied, this LP relaxation provides no useful
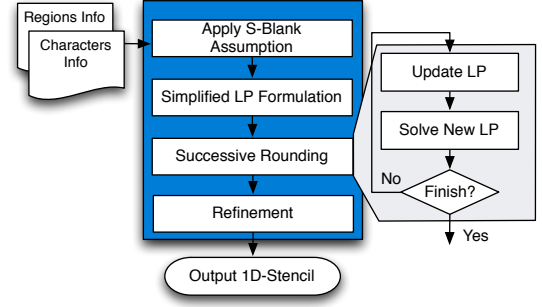


**Figure 3: E-BLOW overall flow for 1D-OSP.**

information to guide future rounding, i.e., all the character candidates are selected and no ordering relationship is determined.

To overcome the limitation of above rounding, E-BLOW proposes a novel iterative solving framework to search near-optimal solution in reasonable runtime. The main idea is to modify the ILP formulation, so that the corresponding LP relaxation can provide good lower bound theoretically. As shown in Fig. 3, the overall flow includes three parts: Simplified ILP formulation, Successive Rounding and Refinement. At section 3.2 the simplified formulation will be discussed, and its LP Rounding lower bound will be proved. Function $SuccRounding()$ is the successive rounding method, which will be introduced at section 3.3. At last, section 3.4 proposes a dynamic programming based refinement.

### 3.1 Symmetrical Blank (S-Blank) Assumption

Our simplified formulation is based on a symmetrical blank assumption: the blanks of each character is symmetry, left slack equals to right slack. Note that for different characters $i$ and $j$, their slacks $s_i$ and $s_j$ can be different.

At first glance this assumption may lose optimality, however, it provides several practical and theoretical benefits. (1) Single row ordering [12] was transferred into Hamilton Cycle problem, which is a well known NP-hard problem and even particular solver is quite expensive. Under the assumption, this ordering problem can be optimally solved in $O(n)$. (2) The ILP formulation can be simplified to provide a reasonable rounding bound theoretically. Compared with previous heuristic framework [12], the proved rounding bound provides a better guideline for a global view search. (3) To compensate the inaccuracy in the asymmetrical blank cases, E-BLOW provides a refinement to further improve the throughput.

Given $p$ character candidates, single row ordering problem adjusts the relative locations to minimize the total width. Under symmetrical blank assumption, this problem can be optimally solved by a two steps greedy approach. First, all characters are sorted decreasingly by blanking space $s_i$; second, they are inserted one by one. Each one can insert at either left end or right end.

**Theorem 1:** *Under S-Blank assumption, the greedy approach can get maximum overlapping space $\sum_i s_i - \max\{s_i\}$.*

In practical, we set $s_i = \lceil (sl_i + sr_i)/2 \rceil$, where $sl_i$ and $sr_i$ are $c_i$'s left slack and right slack, respectively.

### 3.2 Simplified Formulation

To further simplify (2), we modify the objective function through assigning each character $c_i$ with one profit value

$profit_i$. Then based on the Theorem 1, the formulation (2) can be simplified as follow:

$$\max \sum_i \sum_j a_{ij} \cdot profit_i \qquad (3)$$

$$\text{s.t.} \sum_i (w_i - s_i) \cdot a_{ij} \leq W - B_j, \forall j \qquad (3a)$$

$$B_j \geq s_i \cdot a_{ij}, \forall i \qquad (3b)$$

$$\sum_j a_{ij} \leq 1, \quad \forall c_i \in C^C \qquad (3c)$$

$$a_{ij} = 0 \;\; \text{or} \;\; 1 \qquad (3d)$$

(3a) and (3b) are based on Theorem 1 to calculate the row width, where (3b) is to linearize $max$ operation. Here $B_j$ can be viewed as the maximum blank space of all the characters on row $r_j$. (3c) means each character can be assigned into at most one row. It's easy to see that the number of variables is $O(nm)$. Generally speaking, single character number $n$ is much larger than row number $m$, so compared with basic ILP formulation (2), the variable number of (3) can be reduced dramatically.

Furthermore, theoretically the simplified formulation (3) can achieve reasonable LP rounding lower bound. To explain this, let us first look at a similar program (3′) as follows:

$$\max \sum_i \sum_j (w_i - s_i) \cdot a_{ij} \cdot ratio_i \qquad (3')$$

$$\text{s.t.} \sum_i (w_i - s_i) \cdot a_{ij} \leq W - max_s \qquad (3a')$$

$$(3c) - (3d)$$

where $ratio_i = profit_i / (w_i - s_i)$, and $max_s$ is the maximum horizontal slack length of every character, i.e. $max_s = \max\{s_i | i = 1, 2, \ldots, n\}$. Program (3′) is a well known multiple knapsack problem [16].

**Lemma 3.** *If each $ratio_i$ is the same, the multiple knapsack problem (3′) can find a $1/2-$approximation algorithm using LP Rounding method.*

For brevity we omit the proof, detailed explanations can be found in [18]. It shall be noted that if all $ratio_i$ are the same, program (3′) can be approximated to a max-flow problem. Based on Lemma 3, if we denote $\alpha$ as $\max\{ratio_i\} / \min\{ratio_i\}$, we can achieve the following theorem:

**Theorem 2:** *The LP Rounding solution of (3) can be a $0.5/\alpha-$ approximation to program (3′).*

Due to space limit, the detailed proof is omitted. The only difference between (3) and (3′) is that the right side values at (3a) and (3a′). Blank spacing is relatively small comparing with the row length, we can get that $W - max_s \approx W - B_j$. Then based on Theorem 2, we can conclude that program (3) has a reasonable rounding bound.

### 3.3 Successive Relaxation

Because of the reasonable LP rounding property shown in Theorem 2, we propose a successive relaxation algorithm to solve program (3) iteratively. The ILP formulation (3) becomes an LP if we relax the discrete constraint to a continuous constraint as: $0 \leq a_{ij} \leq 1$. The successive relaxation algorithm is shown in Algorithm 1. At first we set all $a_{ij}$ to variables since any $a_{ij}$ is not guided to rows. The LP is updated and solved iteratively. For each new LP solution, we

---

**Algorithm 1** SuccRounding( $th_{inv}$ )

**Input:** ILP Formulation (3)
1: set all $a_{ij}$ to variables;
2: **repeat**
3:　update $profit_i$ for all variables $a_{ij}$;
4:　solve relaxed LP of (3);
5:　**repeat**
6:　　find $a_{pq} = \max\{a_{ij}$, and $c_i$ can insert into row $r_j\}$;
7:　　**for all** $a_{ij} \geq a_{pq} \times th_{inv}$ **do**
8:　　　**if** $c_i$ can be assigned to row $r_j$ **then**
9:　　　　$a_{ij} = 1$ and set it to a non-variable;
10:　　　Update capacity of row $r_j$;
11:　　**end if**
12:　　**end for**
13:　**until** cannot find $a_{pq}$
14: **until**

---

search the maximal $a_{pq}$ (line 6). Then for all $a_{ij}$ that is close the the maximal value $a_{pq}$, we try to pack $c_i$ into row $r_j$, and set it as non-variable. Note that since several $a_{ij}$ are assigned permanent value, the number of variables in updated LP formulation would continue to decrease. This procedure repeats until no appropriate $a_{ij}$ can be found. One key step of the Algorithm 1 is the $profit_i$ update (line 3). For each character $c_i$, we set its $profit_i$ as follows:

$$profit_i = \sum_c \frac{t_c}{t_{max}} \cdot (n_i - 1) \cdot t_{ic} \qquad (4)$$

where $t_c$ is current writing time of region $w_c$, and $t_{max} = \max\{t_c, \forall c \in P\}$. Through applying the $profit_i$, the region $w_c$ with longer writing time would be considered more during the LP formulation. During successive relaxation, if $c_i$ hasn't been assigned to any row, $profit_i$ would continue to updated, so that the total writing time of the whole MCC system can be minimized.

### 3.4 Refinement

Simplified formulation and successive relaxation are under the symmetrical blank assumption. Although it can be effectively solved, for asymmetrical cases it would lose some optimality. To compensate the losing, we present a dynamic programming based refinement procedure. As discussed above, for $k$ characters, single row ordering can have $2^{k-1}$ possible solutions. Under symmetrical blank space assumption, all these orderings get the same length. But for the asymmetrical cases, it does not hold anymore. Our dynamic programming based algorithm $Refine(k)$ finds the best solution from these $2^{k-1}$ options. The detailed is shown in Algorithm 2. At first, if $k > 1$, then $Refine(k)$ will recursively call $Refine(k-1)$ to generate all old partial solutions. All these partial solutions will be updated by adding candidate $c_k$ (lines 6-8). Note that maintaining all solutions is impractical and unnecessary, because many of them are inferior to others. In $SolutionPruning()$, all solutions are checked, if one solution $S_A$ is inferior to another solution $S_B$, $S_A$ would be pruned to save computation cost. For each solution a triplet $(w, l, r)$ is constructed to store the information of width, left slack and right slack. We define the *inferior* relationship as follow. For two solutions $S_A = (w_a, l_a, r_a)$ and $S_B = (w_b, l_b, r_b)$, $S_B$ is inferior to $S_A$ if and only if $w_a \geq w_b$, $l_a \leq l_b$ and $r_a \leq r_b$.

After $Refine(k)$ for each row, if more available spaces are generated, a greedy insertion approach similar to [12] would

**Algorithm 2** Refine(k)

1: **if** k = 1 **then**
2:  Generate partial solution $(w_1, sl_1, sr_1)$;
3: **else**
4:  Refine(k-1);
5:  **for** each partial solution $(w, l, r)$ **do**
6:   $(w_1, l_1, r_1) = (w + w_k - \min(sr_k, l), sl_k, r)$;
7:   $(w_2, l_2, r_2) = (w + w_k - \min(sl_k, r), l, sr_k)$;
8:   Replace $(w, l, r)$ by $(w_1, l_1, r_1)$ and $(w_2, l_2, r_2)$;
9:   **if** solution set size $\geq$ threshold **then**
10:    SolutionPruning();
11:   **end if**
12:  **end for**
13: **end if**

---

**Algorithm 3** KD-Tree based Clustering

**Input:** set of candidates $C^C$.

1: **repeat**
2:  Sort all candidates by $profit_i$;
3:  Set each candidates $c_i$ to unclustered;
4:  **for all** unclustered candidate $c_i$ **do**
5:   Find pair $(c_i, c_j)$ with similar blank spaces and profits;
6:   Cluster $(c_i, c_j)$, label them as clustered;
7:  **end for**
8:  Update candidate information;
9: **until** reach clustering threshold



Figure 5: KD-Tree based region searching.

---

be proposed to further improve the throughput.
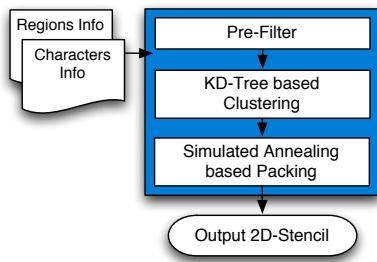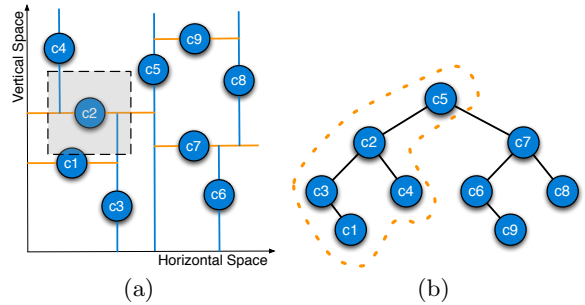
## 4. E-BLOW FOR 2D-OSP



Figure 4: E-BLOW overall flow for 2D-OSP.

Now we consider a more general case: the blanking spaces of characters are non-uniform along both horizontal and vertical directions. This problem is referred as 2D-OSP problem. In [12] the 2D-OSP problem was transformed into a floorplanning problem. However, several key differences between traditional floorplanning and OSP were ignored. (1) In OSP there is no wirelength to be considered, while at floorplanning wirelength is a major optimization objective. (2) Compared with complex IP cores, lots of characters may have similar sizes. (3) Traditional floorplanner could not handle the problem size of modern MCC design. To deal with all these properties, a approximation packing framework is proposed (see Fig. 4). Given the input character candidates, the pre-filter process is first applied to remove characters with bad profit (defined in (4)). Then the second step is a KD-Tree based clustering algorithm to effectively speed-up the design process. Followed by the final floorplanner to pack all candidates.

### 4.1 KD-Tree based Clustering

Clustering is a well studied problem, and there are many of works and applications in VLSI [19]. However, previous methodologies cannot be directly applied here. (1) Traditional clustering is based on netlist, which provides the all clustering options. Generally speaking, netlist is sparse, but in OSP the connection relationships are so complex that any two characters can be clustered, and totally there are $O(n^2)$ clustering options. (2) Given two candidates $c_i$ and $c_j$, there are several clustering options. For example, horizontal clustering and vertical clustering may have different overlapping space.

The details of our clustering procedure are shown in Algorithm 3. The clustering is repeated until the clustered candidate number reaches the clustering threshold. Initially all the candidates are sorted by $profit_i$, it means those candidates with more shot number reduction are tend to be clustered. Then clustering (lines 3-8) is carried out. For each candidate $c_i$, finding available $c_j$ may need $O(n)$, and complexity of the horizontal clustering and vertical clustering are both $O(n^2)$. Then the complexity of the whole procedure is $O(n^2)$, where $n$ is the number of candidates.

A KD-Tree [20] is used to speed-up the process of finding available pair $(c_i, c_j)$. It provides fast $O(logn)$ region searching operations which keeping the time for insertion and deletion small: insertion, $O(logn)$; deletion of the root, $O(n(k-1)/k)$; deletion of a random node, $O(logn)$. Using KD-Tree, the complexity of the Algorithm 3 can be reduced to $O(nlogn)$. A simple example is shown in Fig. 5. For the sake of convenience, here for each candidate we only consider horizontal and vertical space. Given candidate $c_2$, to find another candidate with the similar space, it may need $O(n)$ to scan all other candidates. However, using the KD-Tree structure shown in Fig. 5(a), this finding procedure can be viewed as a region searching, which can be resolved in $O(logn)$. Particularly, as shown in Fig. 5(b), only candidates $c_1 - c_5$ are scanned.

### 4.2 Approximation Framework for 2D-OSP

In E-BLOW we adopt a simulated annealing based framework similar to that in [12]. To demonstrate the effectivity of our pre-filter and clustering methodologies, E-BLOW uses the same parameters with that in [12]. Sequence Pair [21] is used as a topology representation.

## 5. EXPERIMENTAL RESULTS

E-BLOW is implemented in the C++ programming language and executed on a Linux machine with two 3.0GHz

CPU and 32GB Memory. GUROBI [22] is used to solve linear programming. Eight benchmarks from [12] are tested. Besides, eight benchmarks (1M-x) are designed for 1D-OSP and the other eight (2M-x) are generated for the 2D-OSP problem. Character projection (CP) number are all set as 10. For small cases (1M-1, ..., 1M-4, 2M-1, ..., 2M-4) the character candidate number is 1000, and the stencil size is set as $1000\mu m \times 1000\mu m$. For larger cases (1M-5, ..., 1M-8, 2M-5, ..., 2M-8) the character candidate number is 4000, and the stencil size is set as $2000\mu m \times 2000\mu m$. The size and blank width of each character is similar to that in [12].

## 5.1 Comparison for 1D-OSP

For 1D-OSP, Table 1 compares E-BLOW with greedy method and the heuristic framework in [12]. Note that the greedy method was also described in [12]. Column "char #" is number of character candidates, and column "CP#" is number of character projections. For each algorithm, we record "shot #", "char #" and "CPU(s)", where "shot #" is final number of shots and "char #" is number of characters on final stencil, "CPU(s)" reports the runtime. From table 1 we can see E-BLOW achieve better performance and runtime. Compared with E-BLOW, the greedy algorithm introduces 47% more shots number, and [12] would introduce 19% more shots number. Note that compared with heuristic method in [12], mathematical formulation can provide global view, even for traditional EBL system (1D-1, ..., 1D-4), E-BLOW achieves better shot number. Besides, E-BLOW can reduce 34.3% of runtime.

## 5.2 Comparison for 2D-OSP

For 2D-OSP, Table 2 gives the similar comparison. For each algorithm, we also record "shot #", "char #" and "CPU(s)", where the meanings are the same with that in Table 1. From the table we can see that for each test case, although the greedy algorithm is faster, its design results are not good that it would introduce 30% more shot number. Besides, compared with the work in [12], E-BLOW can achieve better performance that the shot number can be reduced by 14%. Meanwhile, because of the clustering method, E-BLOW can reach $2.8\times$ speed-up.

From both tables we can see that compared with [12], E-BLOW can achieve a better tradeoff between runtime and performance.

## 6. CONCLUSION

In this paper, we have proposed E-BLOW, a tool to solve OSP problem in MCC system. For 1D-OSP, a successive relaxation algorithm and a dynamic programming based refinement are proposed. For 2D-OSP, a KD-Tree based clustering method is integrated into simulated annealing framework. Experimental results show that compared with previous works, E-BLOW can achieve better performance in terms of shot number and runtime, for both MCC system and traditional EBL system. As EBL, including MCC system, are widely used for mask making and also gaining momentum for direct wafer writing, we believe a lot more research can be done for not only stencil planning, but also EBL aware design.

## Acknowledgment

## 7. REFERENCES

[1] B. Yu, J.-R. Gao, D. Ding, Y. Ban, J.-s. Yang, K. Yuan, M. Cho, and D. Z. Pan, "Dealing with ic manufacturability in extreme scaling," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2012, pp. 240–242.

[2] A. B. Kahng, C.-H. Park, X. Xu, and H. Yao, "Layout decomposition for double patterning lithography," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2008, pp. 465–472.

[3] B. Yu, K. Yuan, B. Zhang, D. Ding, and D. Z. Pan, "Layout decomposition for triple patterning lithography," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2011, pp. 1–8.

[4] K. Lucas, C. Cork, B. Yu, G. Luk-Pat, B. Painter, and D. Z. Pan, "Implications of triple patterning for 14 nm node design and patterning," in *Proc. of SPIE*, vol. 8327, 2012.

[5] Y. Arisawa, H. Aoyama, T. Uno, and T. Tanaka, "EUV flare correction for the half-pitch 22nm node," in *Proc. of SPIE*, vol. 7636, 2010.

[6] H. C. Pfeiffer, "New prospects for electron beams as tools for semiconductor lithography," in *Proc. of SPIE*, 2009.

[7] H. Yasuda, T. Haraguchi, and A. Yamada, "A proposal for an MCC (multi-column cell with lotus root lens) system to be used as a mask-making e-beam tool," in *Proc. of SPIE*, 2004.

[8] T. Maruyama, Y. Machida, S. Sugatani, H. Takita, H. Hoshino, T. Hino, M. Ito, A. Yamada, T. Iizuka, S. Komatsue, M. Ikeda, and K. Asada, "CP element based design for 14nm node EBDW high volume manufacturing," in *Proc. of SPIE*, 2012.

[9] M. Shoji, T. Inoue, and M. Yamabe, "Extraction and utilization of the repeating patterns for CP writing in mask making," in *Proc. of SPIE*, 2010.

[10] M. Sugihara, T. Takata, K. Nakamura, R. Inanami, H. Hayashi, K. Kishimoto, T. Hasebe, Y. Kawano, Y. Matsunaga, K. Murakami, and K. Okumura, "Cell library development methodology for throughput enhancement of character projection equipment," *IEICE Transactions on Electronics*, vol. E89-C, pp. 377–383, 2006.

[11] K. Yuan and D. Z. Pan, "E-Beam lithography throughput improvement with stencil planning and optimization," in *ACM International Symposium on Physical Design (ISPD)*, 2011.

[12] K. Yuan, B. Yu, and D. Z. Pan, "E-Beam lithography stencil planning and optimization with overlapped characters," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 31, no. 2, pp. 167–179, Feb. 2012.

[13] P. Du, W. Zhao, S.-H. Weng, C.-K. Cheng, and R. Graham, "Character design and stamp algorithms for character projection electron-beam lithography," in *IEEE/ACM Asia and South Pacific Design Automation Conference (ASPDAC)*, 2012.

[14] B. Yu, J.-R. Gao, and D. Z. Pan, "L-Shape based layout fracturing for e-beam lithography," in *IEEE/ACM Asia and South Pacific Design Automation Conference (ASPDAC)*, 2013.

[15] S. Manakli, H. Komami, M. Takizawa, T.Mitsuhashi, and L. Pain, "Cell projection use in mask-less lithography for 45nm & 32nm logic nodes," in *Proc. of SPIE*, 2009.

[16] S. Martello and P. Toth, *Knapsack problems: algorithms and computer implementations*. New York, NY, USA: John Wiley & Sons, Inc., 1990.

[17] C. Kenyon and E. Remila, "Approximate strip packing," in *Foundations of Computer Science, 1996. Proceedings., 37th Annual Symposium on*, oct 1996, pp. 31 –36.

[18] M. Dawande, J. Kalagnanam, P. Keskinocak, F. Salman, and R. Ravi, "Approximation algorithms for the multiple knapsack problem with assignment restrictions," *Journal of Combinatorial Optimization*, vol. 4, pp. 171–186, 2000.

[19] C. J. Alpert and A. B. Kahng, "Recent directions in netlist partitioning: a survey," *Integr. VLSI J.*, vol. 19, pp. 1–81, August 1995.

[20] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Commun. ACM*, vol. 18, pp. 509–517, September 1975.

[21] H. Murataand, K. Fujiyoshi, S. Nakatake, and Y. Kajitani, "VLSI module placement based on rectangle-packing by the sequence-pair," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 12, pp. 1518–1524, 1996.

[22] "GUROBI," http://www.gurobi.com/html/academic.html.

**Table 1: Result Comparison for 1D-OSP**

| | char # | CP # | Greedy in [12] | | | [12] | | | E-BLOW | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | shot # | char # | CPU(s) | shot # | char # | CPU(s) | shot # | char # | CPU(s) |
| 1D-1 | 1000 | 1 | 79193 | 876 | 0.2 | 50809 | 926 | 13.5 | 29536 | 934 | 2.2 |
| 1D-2 | 1000 | 1 | 122259 | 806 | 0.2 | 93465 | 854 | 11.8 | 44544 | 863 | 2 |
| 1D-3 | 1000 | 1 | 179822 | 708 | 0.2 | 152376 | 749 | 9.13 | 78704 | 758 | 2.7 |
| 1D-4 | 1000 | 1 | 223420 | 645 | 0.2 | 193494 | 687 | 7.7 | 107460 | 699 | 3.4 |
| 1M-1 | 1000 | 10 | 83786 | 876 | 0.2 | 53333 | 926 | 13.5 | 45243 | 938 | 4.3 |
| 1M-2 | 1000 | 10 | 123048 | 806 | 0.2 | 95963 | 854 | 11.8 | 81636 | 868 | 5.4 |
| 1M-3 | 1000 | 10 | 184950 | 708 | 0.2 | 156700 | 749 | 9.2 | 140079 | 769 | 10.8 |
| 1M-4 | 1000 | 10 | 225468 | 645 | 0.2 | 196686 | 687 | 7.7 | 179890 | 707 | 7.6 |
| 1M-5 | 4000 | 10 | 377864 | 3417 | 1.02 | 255208 | 3629 | 1477.3 | 227456 | 3650 | 59.2 |
| 1M-6 | 4000 | 10 | 542627 | 315 | 1.02 | 417456 | 3346 | 1182 | 373324 | 3388 | 65.1 |
| 1M-7 | 4000 | 10 | 760650 | 2809 | 1.02 | 644288 | 2986 | 876 | 570730 | 3044 | 58.68 |
| 1M-8 | 4000 | 10 | 930368 | 2565 | 1.01 | 809721 | 2734 | 730.7 | 734411 | 2799 | 65.3 |
| Avg. | - | - | 319454.6 | 1264.7 | 0.47 | 259958.3 | 1594.0 | 362.5 | 217751.1 | 1618.1 | 23.9 |
| Ratio | - | - | **1.47** | 0.78 | **0.02** | **1.19** | 0.99 | **15.2** | **1.0** | 1.0 | **1.0** |

**Table 2: Result Comparison for 2D-OSP**

| | char # | CP # | Greedy in [12] | | | [12] | | | E-BLOW | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | shot # | char # | CPU(s) | shot # | char # | CPU(s) | shot # | char # | CPU(s) |
| 2D-1 | 1000 | 1 | 159654 | 734 | 2.1 | 107876 | 826 | 329.6 | 105723 | 789 | 65.5 |
| 2D-2 | 1000 | 1 | 269940 | 576 | 2.4 | 166524 | 741 | 278.1 | 170934 | 657 | 52.5 |
| 2D-3 | 1000 | 1 | 290068 | 551 | 2.6 | 210496 | 686 | 296.7 | 178777 | 663 | 56.4 |
| 2D-4 | 1000 | 1 | 327890 | 499 | 2.7 | 240971 | 632 | 301.7 | 179981 | 605 | 54.7 |
| 2M-1 | 1000 | 1 | 168279 | 734 | 2.1 | 122017 | 811 | 313.7 | 91193 | 777 | 58.6 |
| 2M-2 | 1000 | 1 | 283702 | 576 | 2.4 | 187235 | 728 | 286.1 | 163327 | 661 | 48.7 |
| 2M-3 | 1000 | 1 | 298813 | 551 | 2.6 | 235788 | 653 | 289 | 162648 | 659 | 52.3 |
| 2M-4 | 1000 | 1 | 338610 | 499 | 2.7 | 270384 | 605 | 285.6 | 195469 | 590 | 53.3 |
| 2M-5 | 4000 | 10 | 824060 | 2704 | 19 | 700414 | 2913 | 3891 | 687287 | 2853 | 59 |
| 2M-6 | 4000 | 10 | 1044161 | 2388 | 20.2 | 898530 | 2624 | 4245 | 717236 | 2721 | 60.7 |
| 2M-7 | 4000 | 10 | 1264748 | 2101 | 21.9 | 1064789 | 2410 | 3925.5 | 921867 | 2409 | 57.1 |
| 2M-8 | 4000 | 10 | 824060 | 2704 | 18.9 | 1176700 | 2259 | 4550.0 | 1104724 | 2119 | 57.7 |
| Avg. | - | - | 507832.1 | 1218.1 | 8.3 | 448477 | 1324 | 1582.7 | 389930.5 | 1291.9 | 56.375 |
| Ratio | - | - | **1.30** | 0.94 | **0.15** | **1.15** | 1.02 | **28.1** | **1.0** | 1.0 | **1.0** |