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

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

ECE Dept. University of Texas at Austin, Austin, TX; †Cadence Inc., CA

Electric Beam Lithography (EBL)

- Promising candidate for next generation lithography process
- Variable Shaped Beam (VSB)
- Charactor Projection (CP): a pattern is pre-designed on the stencil, then it can be printed in one electronic shot;
- Key limitation: has been and still is the low throughput.

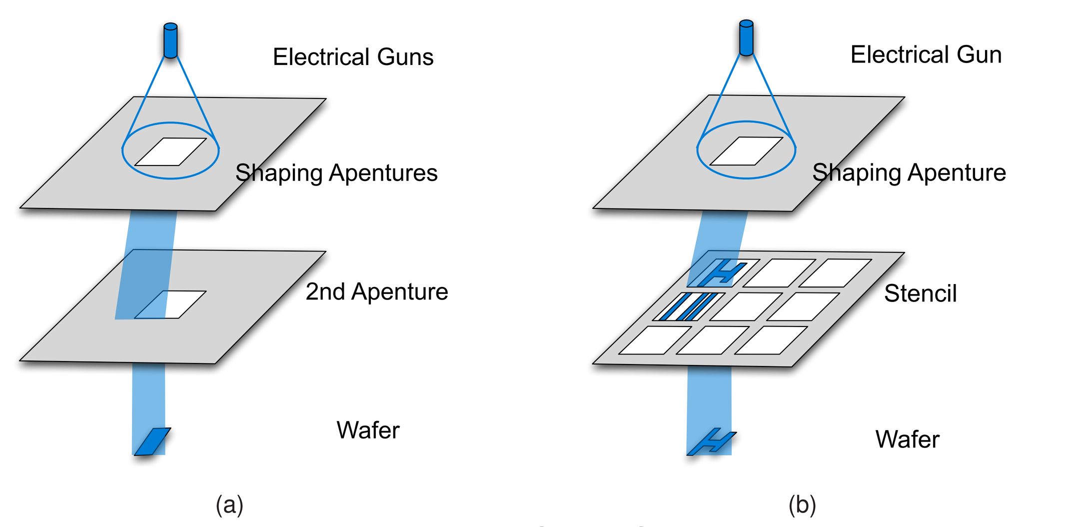
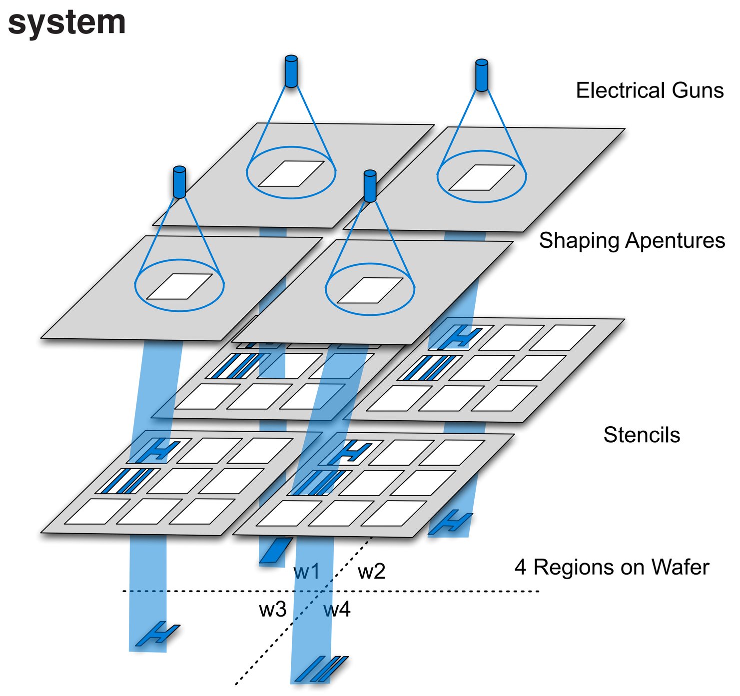


Figure : (a) VSB; (b) CP.

Multi-Column Cell (MCC) system



- Several independent character projections (CP) are used to further speed-up the writing process.
- Each CP is applied on one section of wafer, and all CPs can work parallelly to achieve better throughput.
- Different CPs share one stencil design.

Problem Formulation

Some Definitions

In an MCC system with P CPs, the whole wafer is divided into P regions $\{w_1, w_2, \dots, w_P\}$, and each region is written by one particular CP. 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. Suppose c_i repeats t_{ic} times on region w_c. Let a_i indicate whether c_i is selected. Therefore, for region w_c the total writing time T_c is as follows:

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

The total writing time of the MCC system is formulated as follows:

$$T_{total} = \max\{T_c\} = \max\{T_c^{VSB} - \sum_{i=1}^{n} R_{ic} \cdot a_i\}, \forall c \in P \tag{1}$$

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.

1D-OSP and 2D-OSP

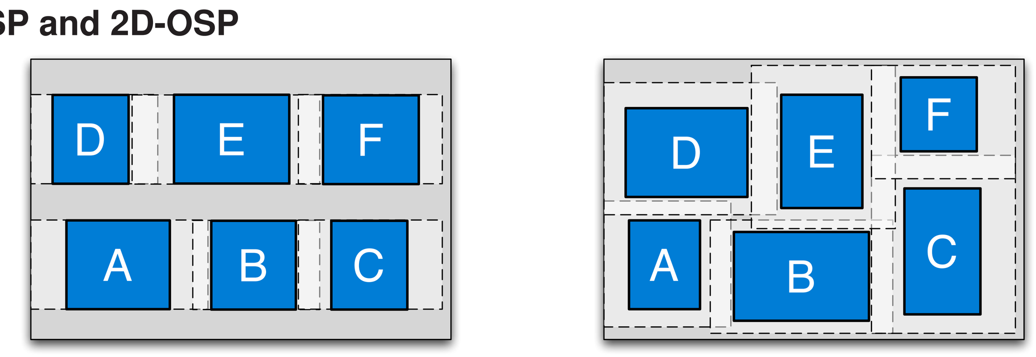
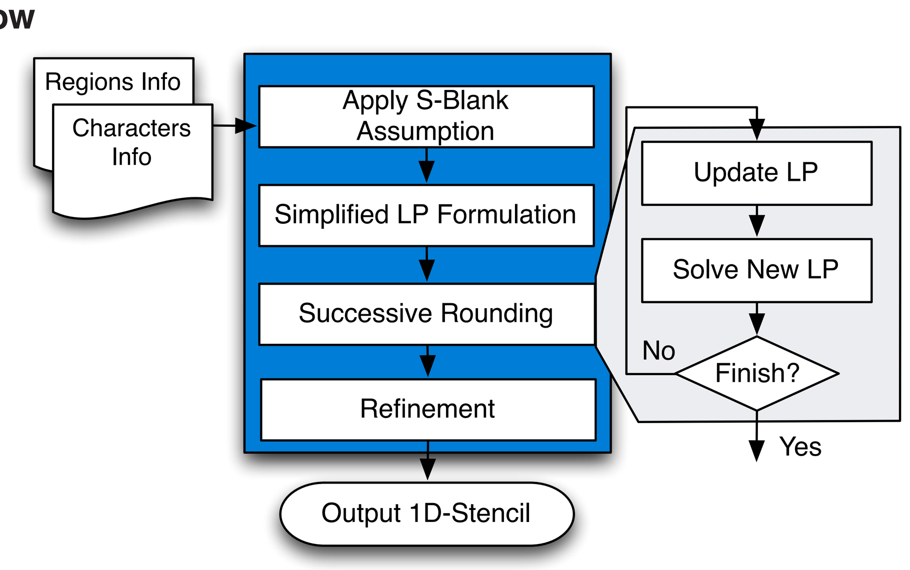


Figure : (a) 1D-OSP; (b) 2D-OSP.

E-BLOW for 1D-OSP

Overall Flow



- Novel iterative solving framework to search near-optimal solution
- Linear programming (LP) relaxation with lower bound theoretically
- Successive rounding
- Dynamic programming based refinement

E-BLOW for 1D-OSP (cont.)

ILP formulation

$$\min \quad T_{total} \tag{2}$$
$$\text{s.t} \quad T_{total} \geq T_c^{VSB} - \sum_{i=1}^{n} (\sum_{k=1}^{M} R_{ic} \cdot a_{ik}), \ \forall c \in P \tag{2a}$$
$$x_i + w_i \leq W, \qquad \forall i \in N \tag{2b}$$
$$\sum_{k}^{m} a_{ik} \leq 1, \qquad \forall k \in M \tag{2c}$$
$$x_i + w_{ij} - x_j \leq W(2 + p_{ij} - a_{ik} - a_{jk}) \tag{2d}$$
$$x_j + w_{ji} - x_i \leq W(3 - p_{ij} - a_{ik} - a_{jk}) \tag{2e}$$
$$a_{ik}, a_{jk}, p_{ij} : 0 - 1 \text{ variable} \tag{2f}$$

Symmetrical Blank (S-Blank) Assumption

- the blanks of each character is symmetry (left slack = right slack).
- Note that for different characters i and j, their slacks s_i and s_j can be different.

Theorem

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

Simplified ILP Formulation

$$\max \sum_i \sum_j a_{ij} \cdot profit_i \tag{3}$$
$$\text{s.t.} \sum_i (w_i - s_i) \cdot a_{ij} \leq W - B_j, \forall j \tag{3a}$$
$$B_j \geq s_i \cdot a_{ij}, \forall i \tag{3b}$$
$$\sum_j a_{ij} \leq 1, \ \forall c_i \in C^C \tag{3c}$$
$$a_{ij} = 0 \text{ or } 1 \tag{3d}$$

The simplified ILP formulation is similar to the following multiple knapsack problem:

$$\max \sum_i \sum_j (w_i - s_i) \cdot a_{ij} \cdot ratio_i \tag{3'}$$
$$\text{s.t.} \sum_i (w_i - s_i) \cdot a_{ij} \leq W - max_s \tag{3a'}$$
$$(3c) - (3d)$$

where $ratio_i = profit_i/(w_i - s_i)$, and max_s is the maximum horizontal slack length of every character, i.e. $max_s = \max\{s_i | i = 1, 2, \dots, n\}$.

Lemma

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

Theorem

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

Successive Relaxation Because of the reasonable LP rounding property, we propose a successive relaxation algorithm to solve program (3) iteratively.

Algorithm: SuccRounding(th_{inv})

Require: 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**

One key step of the Algorithm is the $profit_i$ update (line 3). For each character c_i, we set its $profit_i$ as follows:

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

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.

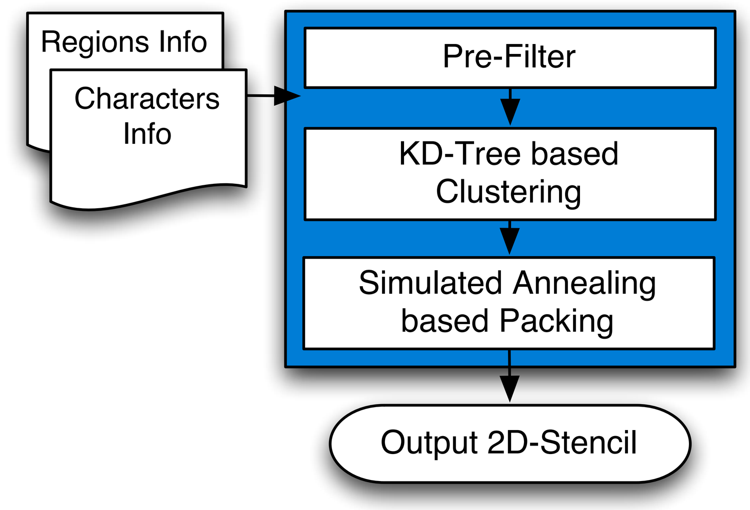
1D-OSP Refinement Simplified formulation and successive relaxation are under the symmetrical blank assumption. Although it can be effectively solved, for asymmetrical cases it would lose some optimality. To compensate the losing, we present a dynamic programming based refinement procedure.

Algorithm: Refine(k)

if k = 1 **then**
Generate partial solution (w_1, sl_1, sr_1);
else
Refine(k-1);
for each partial solution (w, l, r) **do**
$(w_1, l_1, r_1) = (w + w_k - \min(sr_k, l), sl_k, r)$;
$(w_2, l_2, r_2) = (w + w_k - \min(sl_k, r), l, sr_k)$;
Replace (w, l, r) by (w_1, l_1, r_1) and (w_2, l_2, r_2);
if solution set size $\geq$ threshold **then**
SolutionPruning();
end if
end for
end if

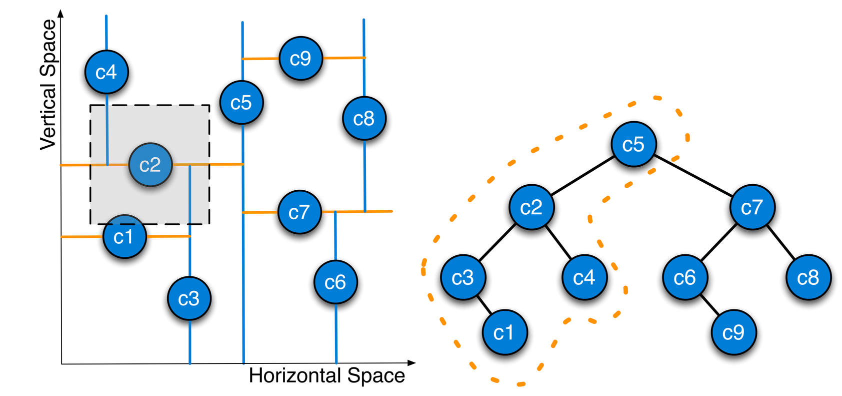
E-BLOW for 2D-OSP

Overall Flow



- Simulated annealing based framework.
- Sequence Pair as topology representation.
- Pre-filter process to remove bad characters.
- Clustering is applied to achieve speedup.

KD-Tree based Clustering

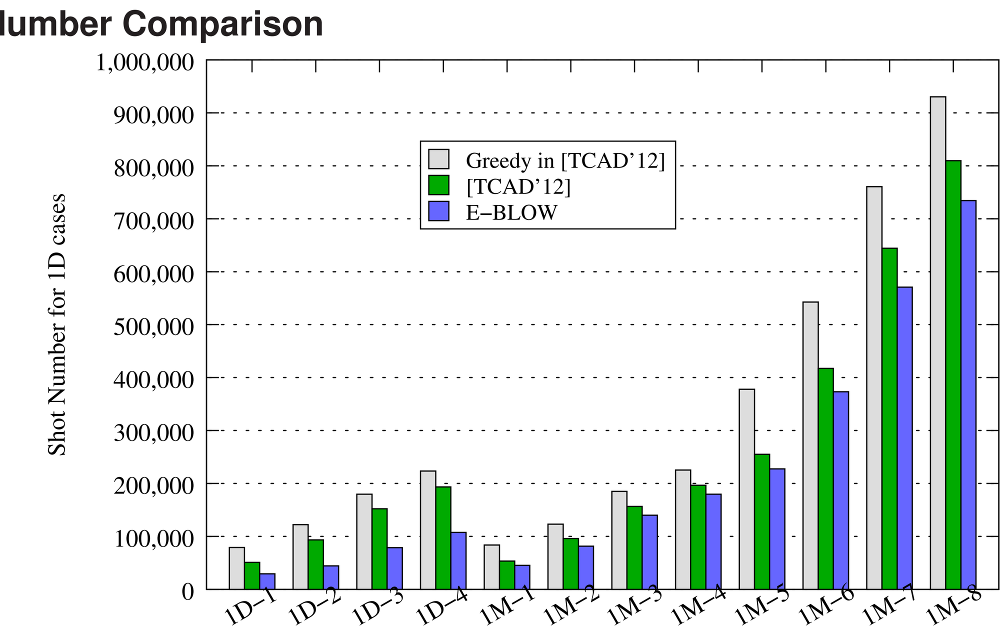


- Speed-up the process of finding available pair (c_i, c_j);
- From $O(n)$ to $O(logn)$;
- For c_2, to find another candidate with the similar space, only scan $c_1 - c_5$.

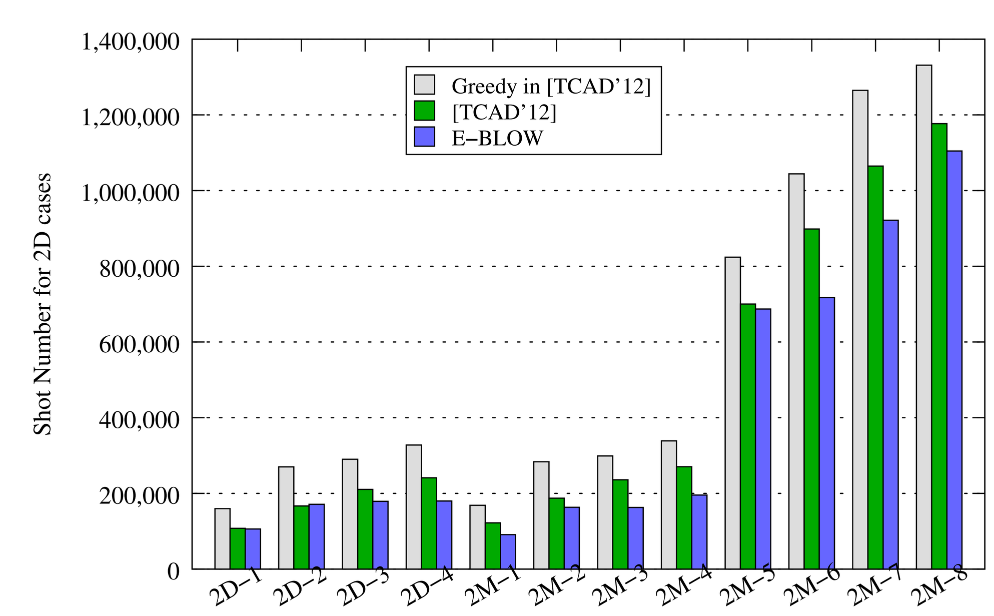
Experimental Results

- Implemented in C++
- Intel Core 3.0GHz Linux machine with 32G RAM
- GUROBI as linear programming (LP) solver

Shot Number Comparison

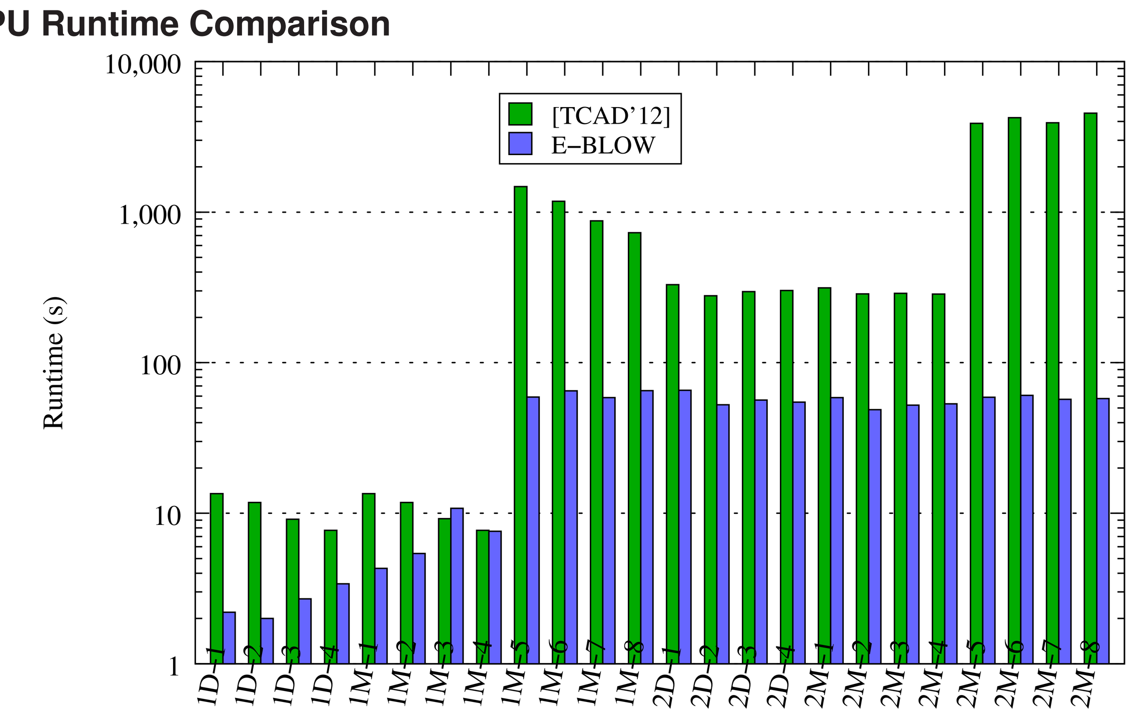


For 1D cases, the greedy algorithm introduces 47% more shots number, and [TCAD'12] introduces 19% more shots number.



For 2D cases, greedy introduces 30% more shot number, while [TCAD'12] introduces 14% more shot number.

CPU Runtime Comparison



Compared with [TCAD'12], E-BLOW can reduce 34.3% of runtime for 1D cases, while 2.8× speedup for 2D cases.

Acknowledgements

- This work is supported in part by NSF and NSFC.