Flexible Packed Stencil Design with Multiple Shaping Apertures for E-Beam Lithography

Chris Chu
Iowa State Univ.
U.S.

Wai-Kei Mak
National Tsing Hua Univ.
Taiwan
Introduction

• Next generation lithography solutions being actively pursued (EUV, e-beam, directed self-assembly, nanoimprint).

• Uses of e-beam
  1. For mask writing
  2. E-beam direct write
     • directly write on wafer
     • avoid ever-increasing mask cost
     • has very high resolution and no depth of focus problem

• Our objective: maximize e-beam writing throughput
E-beam Writing with Character Projection

- Character projection method
  - Patterns that occur many times in a die are made into a set of characters on a stencil
  - Then one shot can print a complex pattern rather than a single rectangle
Character Projection Reduces Shot Count

(a) As a character on the stencil, the whole pattern can be printed in 1 shot.
(b) Otherwise, requires 4 shots using variable-shaped beam (VSB) mode.
Traditional Stencil Design

- Stencil size is limited.
- Traditionally, stencil as a 2D-array for holding characters.
- Pick and place the $N$ most beneficial characters into $N$ pre-designated spots on the stencil.
Flexible Packed Stencil Design w/ Multiple Shaping Apertures

• Increase # chars on stencil by
  – using **multiple shaping apertures** (smaller characters can use smaller shaping apertures)
  – flexible bank space sharing

3 different sized shaping apertures
Flexible Packed Stencil Design w/ Multiple Shaping Apertures

- **Flexible blank space sharing** packs characters in the smallest space.

(a) Traditional packing of chars on a stencil.
(b) Previous works pack chars in smaller space.
(c) Our work pack chars in the smallest space.
Problem Formulation

• Given
  – $K$ : # of shaping apertures allowed
  – $C$ : set of char candidates for a cell-based circuit (each char corr. to a particular orientation of a standard cell)
  – Dimensions of stencil

• How to
  – determine optimal widths for the $K$ shaping apertures
  – choose an optimal subset of chars from $C$ and flexibly pack them on the stencil w/ blank space sharing in order to minimize total shot count for printing the circuit?
Good and Bad News

• Bad news
  – Flexible packed stencil design is NP-hard.

• Good news
  – **Tight linear packing** is near optimal and can be computed efficiently.

Non-tight packing

Tight packing (blank spaces of adjacent chars completely overlap)
Our Algorithm

1. Determine $K$ projection region widths & select a subset of chars to be put on stencil by dynamic programming.

2. Assign chars selected in Step 1 to rows on the stencil & construct a tight linear packing for each row.

3. Greedily pack some of the unselected chars at the end of each row, if possible.
Challenges for DP Formulation

- $O(2^{|C|})$ runtime and memory requirement just to determine whether to include each char in stencil.
  - $|C| > 1000$ (over 1000 cell types used in a circuit)

- Want to simultaneously determine the projection width used by each chosen char s.t. # different projection widths used $\leq K$.

- Width consumed by a set of chars is not equal to their total width (also depends on projection width used by each char & amount of blank space sharing).
Useful Properties

• Let $\nu_c$ be the width of char $c$.
• Let $S$ be the safety margin.
1. Projection width for char $c$ must be $\geq \nu_c + 2S$

2. To choose $K$ optimal projection region widths $e_1, \ldots, e_K$, suffice to consider $e_i = \nu_c + 2S$ for some $c$. (\# distinct widths in a cell library is limited.)
Useful Properties

• Let $E_c$ be the projection region width used by $c$.

3. **Effective width of $c$** in a tight linear packing is
   $$w_c + \frac{(E_c - w_c)}{2} = \frac{(w_c + E_c)}{2}$$

4. If $m$ chars are ordered s.t. $E_i - w_i \leq E_{i+1} - w_{i+1}$ for $i = 1, \ldots, m-1$, then a tight linear packing can be constructed in that order.

   *Proof: By induction*
Useful Properties

5. There exists an optimal solution s.t.

\[ w_c \leq w_d \Leftrightarrow E_c \leq E_d \]

Proof:

6. For chars with same width, one that produces a higher shot saving should be included in stencil with higher priority.

- Shot saving of including char \( c \) is \( r_c (n_{VSBC} - 1) \) where
  \( r_c = \# \text{ times } c \text{ appears in the circuit} \)
  \( n_{VSBC} = \# \text{ shots to print } c \text{ by VSB} \)
Our DP Formulation

• Take advantage of these properties.
• **Character Grouping** technique
  – Avoid considering each char separately.
  – Group all chars according to width.
  – Sort chars in each group in decreasing shot saving.
  – For each group $G$, there are only $|G| + 1$ possible choices (i.e., include first $i$ chars of $G$ where $i = 0$ to $|G|$)
  – Process the groups in decreasing order of their char widths (so no group ever use a projection width larger than that of a previous group).
Experimental Results

• Benchmarks
  – 1D-1 to 1D-4 (1000 chars each) from E-BLOW in DAC’2013
  – 1D-1h to 1D-4h (1200 chars each)

• Stencil size: 1000μm X 1000μm

• Memory requirement
  – without character grouping technique, >18GB in some cases
  – with character grouping technique, < 0.5GB in each case

• Maximum runtime ~40s (Linux server w/ 2.67GHz CPU)
## Experimental Results

<table>
<thead>
<tr>
<th></th>
<th>E-BLOW</th>
<th>Ours (K=1)</th>
<th>Ours (K=2)</th>
<th>Ours (K=3)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>#shots</td>
<td>#chars</td>
<td>#shots</td>
<td>#chars</td>
</tr>
<tr>
<td>1D-1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>29536</td>
<td>934</td>
<td>12972</td>
<td>980</td>
</tr>
<tr>
<td>1D-2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>44544</td>
<td>863</td>
<td>28594</td>
<td>895</td>
</tr>
<tr>
<td>1D-3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>78704</td>
<td>758</td>
<td>55761</td>
<td>797</td>
</tr>
<tr>
<td>1D-4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>107460</td>
<td>699</td>
<td>79275</td>
<td>734</td>
</tr>
<tr>
<td>Normalized</td>
<td>1.65</td>
<td>0.955</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Ours (K=1)</th>
<th>Ours (K=2)</th>
<th>Ours (K=3)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>#shots</td>
<td>#chars</td>
<td>#shots</td>
</tr>
<tr>
<td>1D-1h</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>58648</td>
<td>980</td>
<td>26467</td>
</tr>
<tr>
<td>1D-2h</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>86176</td>
<td>905</td>
<td>48891</td>
</tr>
<tr>
<td>1D-3h</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>135332</td>
<td>800</td>
<td>93109</td>
</tr>
<tr>
<td>1D-4h</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>169105</td>
<td>739</td>
<td>116219</td>
</tr>
<tr>
<td>Normalized</td>
<td>1</td>
<td>1</td>
<td>0.598</td>
</tr>
</tbody>
</table>
Conclusions

• Developed an efficient algorithm for flexible packed stencil design w/ multiple shaping apertures by taking note of several useful properties.

• # e-beam shots to print a circuit is greatly reduced by
  – Selecting optimal shaping aperture size(s)
  – Using multiple shaping apertures
  – Flexible blank space sharing

• Directly applicable to multi-beam direct write system.