E-Beam Lithography Character and Stencil Co-Optimization

Wai-Kei Mak, *Member, IEEE,* and Chris Chu, *Fellow, IEEE*

***Abstract*—Electron-beam maskless lithography is being actively explored by the semiconductor industry for chip production in the sub-22nm regime. Character projection allows in one e-beam shot the printing of complex pattern rather than merely a single rectangle or triangle as in variable-shaped beam projection. However, those circuit patterns that do not match any character on the stencil still have to be written by variable-shaped beam projection. We investigate a new problem of character and stencil co-optimization with blank space sharing between characters so as to minimize the total number of shots required for printing a circuit. We exploit the fact that the blank spaces on the sides of a character can be adjusted by moving the pattern to be printed within its projection region to facilitate blank space sharing so as to pack more characters into the stencil. Even though the co-optimization problem is shown to be NP-complete, we are able to design an elegant approximation algorithm, CASCO. Experiments confirm that the solutions by CASCO are nearly optimal. Compared to the published state of the art, CASCO reduces the shot count by 1.59× while it is also orders of magnitude faster.**

***Index Terms*—e-beam lithography, character projection, stencil design.**

I. Introduction

To extend Moore's law, different solutions like extensions of 193 immersion with multiple patterning, extreme ultraviolet (EUV) lithography, directed self-assembly (DSA), nanoimprint, and electron beam (e-beam) lithography [1]–[3] are being explored by the semiconductor industry. E-beam lithography directly shoots e-beams onto a wafer to define nanometer structures of a circuit. E-beam lithography has several advantages. Compared to optical lithography, it has very high resolution and has no depth of focus problem. In addition, e-beam direct write avoids the ever-rising mask costs. But the key issue with e-beam direct write is the time it takes to write a wafer.

The throughput of an e-beam writing system using the traditional variable-shaped beam (VSB) method [4] is very low. In the VSB method, the patterns to be printed are divided into constituent rectangles and triangles as in Fig. 1(a), and each of which is shot onto the wafer sequentially. So the method of character projection (CP) as shown in Fig. 2 has been introduced to improve the throughput of e-beam lithography [5]. In CP, a character represents some complex pattern within an area of several square microns that can be projected in just one shot (as in Fig. 1(b)). A collection of characters corresponding to some frequently utilized patterns are implemented in a stencil. For a given layout, any pattern corresponding to a character in the stencil can be printed in one shot. Remaining patterns that do not match any character would be printed by VSB. Since character projection can significantly reduce the shot count, which is directly proportional to the writing time, the throughput of e-beam writing system can be significantly improved. Besides, the throughput of shooting a wafer can be further increased by employing a multi-beam system [1], [2].

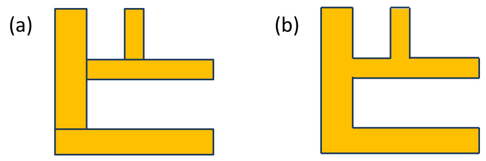

Fig. 1. (a) VSB requires four shots to print the pattern. (b) CP requires just a single shot.

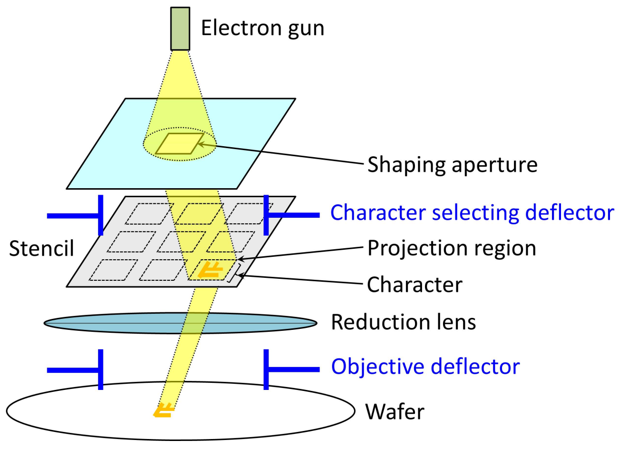

Fig. 2. An e-beam writing system for CP.

In conventional stencil design, characters are placed in a regular 2-D array on the stencil as shown in Fig. 3. An e-beam writing system uses electrostatic deflection (see Fig. 2) to switch between characters on the stencil instantaneously without any physical movement of the stencil, which would require re-calibration and take orders of minutes. The maximum size of a stencil is constrained to maintain the accuracy of e-beam deflection. This limits the number of characters that can be put on a stencil. Stencil design for the front-end-of-line layers of standard cell-based devices are considered in [6]–[9].

This work was supported in part by the National Science Council under Grant NSC 102-2220-E-007-013.

Wai-Kei Mak is with the Department of Computer Science, National Tsing Hua University, 101 Kuan Fu Rd. Sec. 2, Hsinchu, Taiwan 300 R.O.C. (email: wkmak@cs.nthu.edu.tw).

Chris Chu is with the Department of Electrical and Computer Engineering, Iowa State University, Ames, IA 50010 (email: cnchu@iastate.edu).

Character and/or stencil design for interconnect layers and via layers are addressed in [8]–[11].

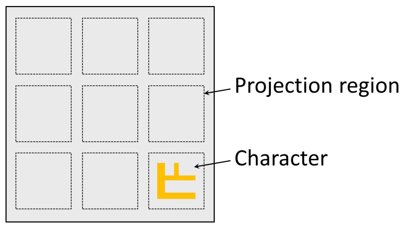

Fig. 3. Typical arrangement of characters on a stencil.

When an e-beam is projected onto the stencil, it covers a region which we call the *projection region* (see Fig. 2). The size of the projection region, which is determined by the shaping aperture, should be larger than the bounding box of the pattern in a character. Since character patterns vary greatly, a character is usually surrounded by considerable blank spaces on its four sides in its enclosing projection region. To pack more characters onto a stencil, Yoshida et al. [12] and Fujimura [13] pointed out that one can overlap the blank spaces of neighboring projection regions as illustrated in Fig. 4.

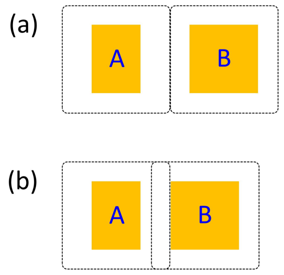

Fig. 4. (a) Two characters are placed side by side without overlapping of blank spaces. (b) The characters are placed with overlapping of blank spaces.

The problem formulation in Yuan et al. [8] and Yu et al. [9] assumed that for each character, the location of the pattern to be printed within its projection region was arbitrarily set. They made no attempt to adjust the location of the pattern. We want to point out that the character projection systems actually allow the pattern of a character to be located anywhere within its enclosing projection region as long as they are not too close to the boundary.[1] Hence, we propose a new problem of *character and stencil co-optimization with blank space sharing* for a cell-based chip in which we exploit this flexibility to co-optimize the character and stencil design. Our main contributions are summarized as follows.

1) We point out that there is a degree of freedom in packed stencil design which was overlooked in previous publications. By exploiting the flexibility of placing the pattern of a character almost anywhere within its enclosing projection region, more characters can be packed and the throughput of e-beam writing can be improved.
2) We formulate a new character and stencil co-optimization problem to make optimal use of a stencil.
3) We show that character and stencil co-optimization is NP-complete.
4) We design a very efficient approximation algorithm CASCO for character and stencil co-optimization that produces near optimal results.

The rest of the paper is organized as follows. In Section II, we formally define our character and stencil co-optimization problem. In Section III-A, we prove that the problem is NP-complete. In Section III-B, we introduce the notion of tight packing and derive some interesting properties which serve as the basis for solving the character and stencil co-optimization problem nearly optimally. In Section IV, we present our algorithm CASCO and show its performance bound. Experimental comparisons are reported in Section V. Finally, a conclusion is given in Section VI.

II. Preliminaries and Problem Formulation

It is easy to see that each time an e-beam is shot through the shaping aperture onto the stencil (see Fig. 2), it will only cover a certain region on the stencil. As pointed out by [12], in practice there could be some stray or scattered electrons from the e-beam shaped by the shaping aperture. In Fig. 5, we use the dotted boundary to represent the projection region boundary after accounting for stray electrons. The effective printing region is the area where the e-beam from the shaping aperture is supposed to cover ideally. The area outside the effective printing region but within the dotted boundary is the area where some stray electrons may reach. The required safety margin depends on the degree and amount of anticipated scattering and straying of electrons of the e-beam passing through the shaping aperture [12].

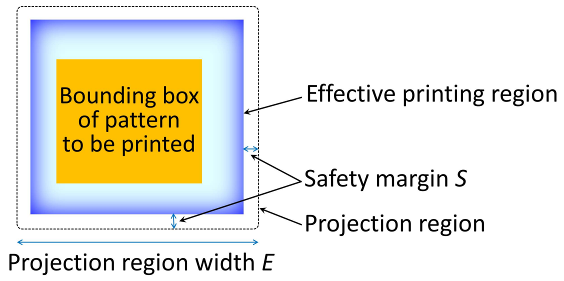
Bounding box of pattern to be printed
Effective printing region
Safety margin S
Projection region
Projection region width E

Fig. 5. Effective printing region and safety margin of an e-beam shot on the stencil.

In order to print characters on the wafer with no loss of accuracy, the following conditions have to be satisfied. First, the pattern of each character must lie within the effective printing region. Second, the pattern of any character A cannot overlap with the projection region of another character B. Otherwise, partial image of character A would be printed erroneously when printing character B. However, the blank space of two neighboring characters may overlap as in the example of Fig. 4(b).

We notice that as long as the two conditions above are satisfied, the e-beam writing system can print any character

[1]Previous works on character design like [10], [11] simply assume that a pattern has to be within its enclosing projection region, but in Section II, we point out that a safety margin must be left in practice.

with the pattern located anywhere within its effective printing region. We show by an example in Fig. 6 that this flexibility can be exploited to increase the number of characters packed in a stencil. Fig. 6(a) shows three characters A to C, and the available character area of a stencil. Note that the blank spaces in the three characters are quite different because the widths of the three patterns are different. The approach in [8] would try to pack them all into the available character area by heuristically maximizing the amount of shared blank space between neighboring projection regions. However, even though the stencil design in Fig. 6(b) would maximize blank space sharing, the total required width would still exceed that of the stencil. So only two characters A and C can be packed into the stencil. On the other hand, if we relocate the character patterns within their effective printing regions as in Fig. 6(c), all three characters can be packed onto the stencil as shown in Fig. 6(d). Note that it is not a problem for the blank space of a projection region to lie outside of the available character area of the stencil [12] since it is easy to filter out the part of the beam that falls outside the stencil.

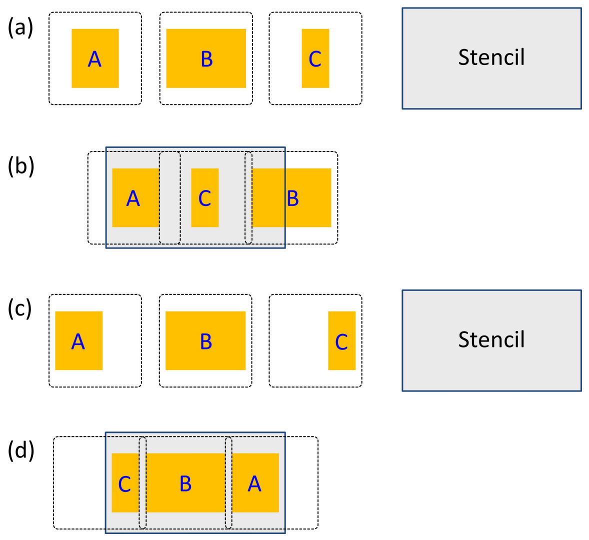

Fig. 6. (a) Three characters and a stencil. (b) The approach in [8] considers stencil design alone and cannot pack all three characters. (c) The character patterns are relocated within their effective printing regions. (d) All three characters can be packed after character and stencil co-optimization.

In this paper, we assume that a set of character patterns is given. We also assume that standard cells are implemented by the characters and all standard cells are of the same height. Hence, the heights of the character pattern for different characters are equal. However, the widths of the character patterns are largely different. Selected characters are arranged in a row-based manner on a stencil. To minimize wastage of inter-row blank space, a character pattern is placed within its projection region such that the top blank space and the bottom blank space are equal. In this way, the bottom blank space of a row of characters can completely overlap with the top blank space of the row of characters below as in Fig. 7. As a result, the number of rows that can fit into a stencil is fixed.

We introduce our notations and define the character and stencil co-optimization problem below.

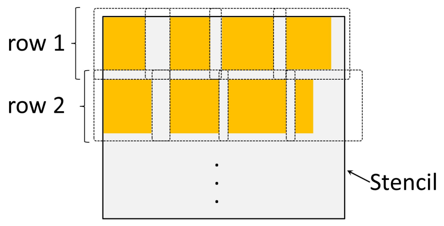

Fig. 7. Row-based stencil design.

- E denotes the width of the projection region for an e-beam shot.
- S denotes the safety margin from the effective printing region to the projection region boundary for an e-beam shot.
- W denotes the width of the available character area of the stencil.
- R denotes the number of character rows that can fit in the stencil.
- w_c denotes the width of the pattern of character c. For simplicity, we will refer to the width of the pattern of a character as the character width in the rest of the paper.
- r_c denotes the number of occurrence of character c in a circuit.
- n_{VSB_c} denotes the number of e-beam shots required to print character c by the VSB method.

Def 1 (Character and Stencil Co-Optimization): Given the values of E, S, W, and R for an e-beam writing system, and a set of character patterns extracted for a circuit with the values of w_c, r_c, and n_{VSB_c} for each character c, select a subset of character patterns and place them on the stencil in a row-based manner to maximize the shot saving with character projection for printing the entire circuit under the following constraints: (1) The pattern of each character must lie within the effective printing region. (2) The pattern of each character cannot lie within the projection region of any other character. (3) The patterns of all characters must lie within the available character area of the stencil.

We call this the character and stencil co-optimization problem since the placement location of each selected character on the stencil and the placement location of the pattern within each selected character have to be optimized simultaneously as in Fig. 6(d). It is easy to see that for each character c, its width w_c must satisfy $w_c \leq E - 2S$ or the character pattern cannot lie within the effective printing region of an e-beam shot. So we will assume that $w_c \leq E - 2S$ for any character c throughout the rest of the paper.

III. Single-Row Character Packing

In this section, we consider the single-row character packing (SRCP) problem, which is related to character and stencil co-optimization. The SRCP problem is shown to be NP-complete. On the other hand, we also derive some interesting theoretical results which will be used as the basis for solving the character and stencil co-optimization problem nearly optimally.

Def 2 (Single-Row Character Packing): Given a set of character patterns, place them into a single row to minimize the row width subject to the constraints below: (1) The pattern of each character must lie within the effective printing region. (2) The pattern of each character cannot lie within the projection region of any other character. (3) The patterns of all characters must lie within the row.

A. SRCP is NP-Complete

The decision version of the SRCP problem is to determine if a set of characters can all be packed into a single row of width W.

Lemma 1: Determining if a set of characters can all be packed into a single row of width W is NP-complete.
The proof of this lemma is included in the appendix.

The above lemma implies that the character and stencil co-optimization problem is also NP-complete. Consider the character and stencil co-optimization problem for a stencil with only one row. The decision version of the single-row character and stenicl co-optimization problem is to determine if we can select a subset of characters from $\mathcal{C}$ and pack them into a row of width W to achieve a shot saving of at least n. If we take n to be $\sum_{c\in\mathcal{C}} r_c(n_{VSB_c}-1)$, then determining if we can select a subset of characters from $\mathcal{C}$ and pack them into a row of width W to achieve a shot saving of at least $n=\sum_{c\in\mathcal{C}} r_c(n_{VSB_c}-1)$ is equivalent to determining if all characters in $\mathcal{C}$ can be packed into a single row of width W. But the latter problem is NP-complete by Lemma 1. Hence, the character and stencil co-optimization problem must be NP-complete.

Corollary 1: The character and stencil co-optimization problem is NP-complete.

B. Properties and Construction of Tight Packing

Since the SRCP problem is NP-complete, it is unlikely that it can be solved optimally in polynomial time. However, we show that if one can find a *tight packing*, it will not be far from optimal. Moreover, we show that a tight packing of k characters can be constructed in $O(k\log k)$ time.

We assume the characters in a packing are indexed from left to right by $1,2,3,\ldots$. We call a packing *tight* if for any two neighboring characters i and $i+1$, the right blank space of character i and the left blank space of character $i+1$ are exactly equal and completely overlap. For example, Fig. 8(a) shows a tight packing of six characters and Fig. 8(b) shows a tight packing of five characters. Note that in Fig. 8, each thick double arrow represents a distance of at least S. Hence, all character patterns are within their respective effective printing regions. Moreover, the pattern of each character is outside of the projection regions of all other characters.

It is interesting to note that the expression for the width of a tight packing depends on whether there is an even or an odd number of characters. Moreover, the width of a tight packing of an even number of characters depends on the width of the left blank space of the first character while the width of a tight packing of an odd number of characters does not. For example, the width of the tight packing in Fig. 8(a) is equal to $w_2+w_4+w_6+3E-s_0$ while the width of the tight packing in Fig. 8(b) is equal to $w_1+w_3+w_5+2E$.

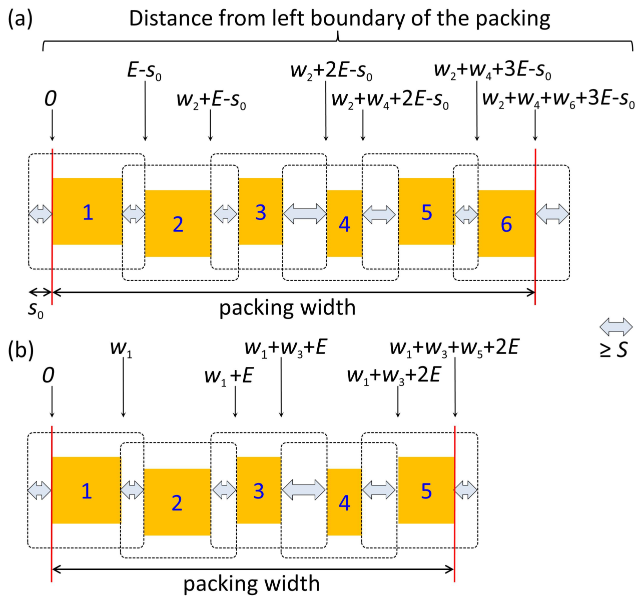

Fig. 8. (a) A tight packing with an even number of characters. (b) A tight packing with an odd number of characters. (Characters with even indexes are shifted down a bit to show the projection regions of different characters more clearly.)

In general, it is not difficult to see that for a tight packing of k characters where k is even, the packing width is given by $w_2+w_4+\ldots+w_k+kE/2-s_0$ where s_0 is the width of the left blank space of the first character. On the other hand, for a tight packing of k characters where k is odd, the packing width is given by $w_1+w_3+\ldots+w_k+(k-1)E/2$.

Alternatively, we may express the width of a tight packing in terms of the *effective widths* of its characters as shown below. The effective width of a character c is defined as $(w_c+E)/2$.

Lemma 2: For a packing P of k characters, let $W(P)$ denote the width of P, $w_i(P)$ denote the width of the i-th character $(i=1,2,\ldots,k)$, $s_0(P)$ denote the width of the left blank space of the first character, and $s_k(P)$ denote the width of the right blank space of the last character in P.
For a tight packing P_t,

$$W(P_t)=\sum_{i=1}^{k}(w_i(P_t)+E)/2-s_0(P_t)/2-s_k(P_t)/2 \quad (1)$$

For an arbitrary packing P_a,

$$W(P_a)\geq\sum_{i=1}^{k}(w_i(P_a)+E)/2-s_0(P_a)/2-s_k(P_a)/2 \quad (2)$$

Proof: For a tight packing P_t, the right blank space of the i-th character and the left blank space of the $(i+1)$-th character are exactly equal and completely overlap for $i=1,2,\ldots,k-1$. So, its width can be expressed as $W(P_t)=\sum_{i=1}^{k} w_i(P_t)+\sum_{i=1}^{k-1} s_i(P_t)$ where $s_i(P_t)$ denote the width of the right blank space of the i-th character in P_t.

For each character i in P_t, we have $s_{i-1}(P_t)+w_i(P_t)+s_i(P_t)=E$ since its left and right blank spaces are equal to $s_{i-1}(P_t)$ and $s_i(P_t)$, respectively. Hence, $\sum_{i=1}^{k}(s_{i-1}(P_t)+w_i(P_t)+s_i(P_t))=kE$. It implies that $2\times\sum_{i=1}^{k-1} s_i(P_t)=$

$kE-\sum_{i=1}^{k} w_i(P_t)-s_0(P_t)-s_k(P_t)$. So, the width of a tight packing P_t can be rewritten as

$$\begin{aligned} W(P_t) &= \sum_{i=1}^{k} w_i(P_t)/2+(kE-s_0(P_t)-s_k(P_t))/2 \\ &= \sum_{i=1}^{k}(w_i(P_t)+E)/2-s_0(P_t)/2-s_k(P_t)/2. \end{aligned}$$

For an arbitrary packing P_a, we let $g_i(P_a)$ be the width of the gap between the i-th and $(i+1)$-th character patterns in P_a for $i=1,2,\ldots,k-1$, and let $g_0(P_a)$ be $s_0(P_a)$ and $g_k(P_a)$ be $s_k(P_a)$. Then P_a's width can be expressed as $W(P_a)=\sum_{i=1}^{k} w_i(P_a)+\sum_{i=1}^{k-1} g_i(P_a)$. Note that $g_{i-1}(P_a)+w_i(P_a)+g_i(P_a)\geq E$ for each character i. It implies that $2\times\sum_{i=1}^{k-1} g_i(P_a)\geq kE-\sum_{i=1}^{k} w_i(P_a)-s_0(P_a)-s_k(P_a)$. So,

$$W(P_a)\geq\sum_{i=1}^{k}(w_i(P_a)+E)/2-s_0(P_a)/2-s_k(P_a)/2.$$

■

Equation (1) tells us that the width of any tight packing P_t is less than the total effective width of its characters by $s_0(P_t)/2+s_k(P_t)/2$. In fact, we can show that the width of any tight packing is not far from optimal.

Lemma 3: Given k characters, the width of any tight packing is less than $E-2S$ away from the minimum width packing.

Proof: Consider a tight packing P_t and an optimal packing P^*.

By Equation (1) and Equation (2), we get

$$W(P_t)-W(P^*)\leq(s_0(P^*)+s_k(P^*)-s_0(P_t)-s_k(P_t))/2$$

as $\sum_{i=1}^{k} w_i(P_t)=\sum_{i=1}^{k} w_i(P^*)$.

Since $s_0(P^*)\leq E-S-w_1(P^*)$ and $s_k(P^*)\leq E-S-w_k(P^*)$, while $s_0(P_t)\geq S$ and $s_k(P_t)\geq S$, it implies that

$$\begin{aligned} &W(P_t)-W(P^*) \\ &\leq\ (E-S-w_1(P^*)+E-S-w_k(P^*)-S-S)/2 \\ &<\ E-2S \end{aligned}$$

Hence, the width of P_t is less than $E-2S$ away from the width of P^*. ■

In the following, we derive the necessary and sufficient conditions for the existence of a tight packing under a given character order. Subsequently, we propose some simple ways to construct a tight packing given k characters.

Lemma 4: Suppose a character order is given and w_i denotes the width of the i-th character from the left. The necessary and sufficient conditions for the existence of a tight packing under the given character order are:

$$\begin{aligned} s_0 &\geq S \\ E &\geq S+s_0+w_1 \\ s_0+w_1 &\geq S+w_2 \\ E+w_2 &\geq S+s_0+w_1+w_3 \\ s_0+w_1+w_3 &\geq S+w_2+w_4 \\ E+w_2+w_4 &\geq S+s_0+w_1+w_3+w_5 \\ s_0+w_1+w_3+w_5 &\geq S+w_2+w_4+w_6 \\ &\vdots \end{aligned}$$

where s_0 denotes the width of the left blank space of the first character.

Proof: Refer to Fig. 9 which shows a tight packing. The length of each double-headed arrow must be greater than or equal to the safety margin S. Hence, we have

$$\begin{aligned} s_0 &\geq S \\ (E-s_0)-w_1 &\geq S \\ (w_1+E)-(w_2+E-s_0) &\geq S \\ (w_2+2E-s_0)-(w_1+w_3+E) &\geq S \\ (w_1+w_3+2E)-(w_2+w_4+2E-s_0) &\geq S \\ (w_2+w_4+3E-s_0)-(w_1+w_3+w_5+2E) &\geq S \\ (w_1+w_3+w_5+3E)-(w_2+w_4+w_6+3E-s_0) &\geq S \\ &\vdots \end{aligned}$$

After simplification and re-arrangement, we can get the conditions listed in the lemma. ■

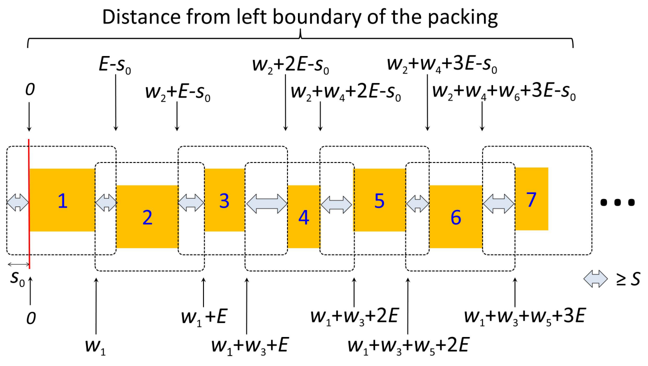

Fig. 9. Conditions for a tight packing.

Lemma 5: A tight packing can be constructed with any character as the first character and the remaining characters ordered in decreasing width.

Proof: Note that the conditions in Lemma 4 will be satisfied if $w_2\geq w_3\geq\ldots\geq w_k$ as long as it is also true that $s_0\geq S$, $E\geq S+s_0+w_1$, and $s_0+w_1\geq S+w_2$. If we set s_0 to $E-S-w_1$, then $s_0\geq S$, $E\geq S+s_0+w_1$, and $s_0+w_1\geq S+w_2$ are all true since $w_1\leq E-2S$ and $w_2\leq E-2S$ by assumption. Hence, a tight packing can be constructed if the second to the last characters are ordered in decreasing width. ■

Fig. 10 shows several tight packings of four characters with widths 8, 7, 5 and 2. In this example, we assume $E=10$ and $S=1$. In each tight packing, a different character is used as the first character and the remaining characters are ordered in decreasing width.

By Lemma 5, any character can be used as the first character to construct a tight packing. A natural question is which character should be chosen as the first character in order to minimize the packing width. We answer the question in the lemma below. In short, the character with the smallest width

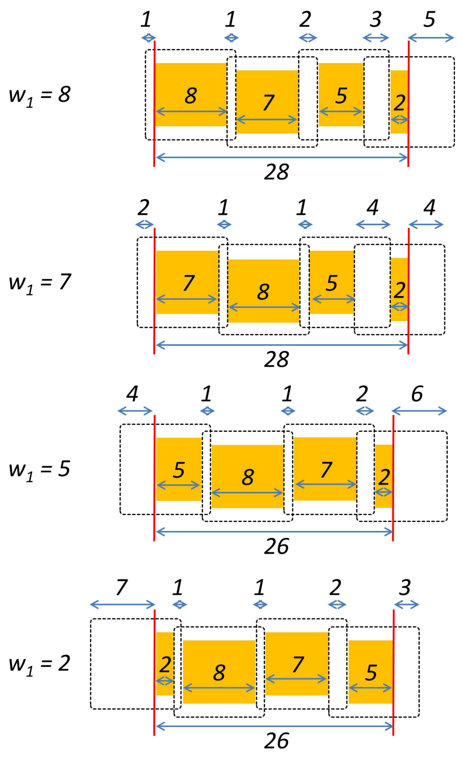

Fig. 10. Tight packings of four characters with widths 8, 7, 5 and 2. We assume $E = 10$ and $S = 1$ here. Different characters are used as the first character.

should be chosen as the first character. In the next section, we will apply this result in our design of the character and stencil co-optimization algorithm.

Lemma 6: Suppose the width of character c_j ($j = 1, 2, \ldots, k$) is u_j and $u_1 \geq u_2 \geq \ldots \geq u_k$. Let W_j denote the minimum tight packing width with c_j as the first character and the remaining characters ordered in decreasing width. Then, $W_1 \geq W_2 \geq \ldots \geq W_k$.

Proof: We consider the cases for even and odd k separately since the expression for the width of a tight packing depends on whether there is an even or odd number of characters.

Case 1: k is odd. Recall that for an odd number of characters, the width of a tight packing is given by $w_1 + w_3 + \ldots + w_k + (k-1)E/2$ where w_i is the width of the i-th character from the left in the packing. By definition, W_j is the minimum tight packing width when the character sequence is (c_j, c_1, c_2, $c_3, \ldots, c_{j-1}$, c_{j+1}, c_{j+2}, $c_{j+3}, \ldots, c_k$). So,

$$W_j = \begin{cases} u_j + u_2 + u_4 + \ldots + u_{j-1} \\ +u_{j+2} + u_{j+4} + \ldots + u_k \\ +(k-1)E/2 & \text{if } j \text{ is odd} \\ \\ u_j + u_2 + u_4 + \ldots + u_{j-2} \\ +u_{j+1} + u_{j+3} + \ldots + u_k \\ +(k-1)E/2 & \text{if } j \text{ is even} \end{cases}$$

Similarly, W_{j+1} is the minimum tight packing width when the character sequence is (c_{j+1}, c_1, c_2, $c_3, \ldots, c_j$, c_{j+2}, c_{j+3}, $c_{j+4}, \ldots, c_k$). So,

$$W_{j+1} = \begin{cases} u_{j+1} + u_2 + u_4 + \ldots + u_{j-1} \\ +u_{j+2} + u_{j+4} + \ldots + u_k \\ +(k-1)E/2 & \text{if } j \text{ is odd} \\ \\ u_{j+1} + u_2 + u_4 + \ldots + u_{j-2} \\ +u_j + u_{j+3} + u_{j+5} + \ldots + u_k \\ +(k-1)E/2 & \text{if } j \text{ is even} \end{cases}$$

It implies that

$$\begin{aligned} W_j - W_{j+1} &= \begin{cases} u_j - u_{j+1} & \text{if } j \text{ is odd} \\ 0 & \text{if } j \text{ is even} \end{cases} \\ &\geq 0 \end{aligned}$$

Hence, we have $W_1 \geq W_2 \geq \ldots \geq W_k$.

Case 2: k is even. Recall that for an even number of characters, the width of a tight packing is given by $w_2 + w_4 + \ldots + w_k + kE/2 - s_0$ where w_i is the width of the i-th character from the left in the packing and s_0 is the left blank space of the first character. s_0 should be adjusted to minimize the width, therefore s_0 should be set to $E - S - w_1$ since the right blank space of the first character cannot be smaller than the safety margin S. Then, the packing width can be expressed as $w_2 + w_4 + \ldots + w_k + kE/2 - (E - S - w_1)$.

By definition, W_j is the minimum tight packing width when the character sequence is (c_j, c_1, c_2, $c_3, \ldots, c_{j-1}$, c_{j+1}, c_{j+2}, $c_{j+3}, \ldots, c_k$). So,

$$W_j = \begin{cases} u_1 + u_3 + \ldots + u_{j-2} \\ +u_{j+1} + u_{j+3} + \ldots + u_k \\ +kE/2 - (E - S - u_j) & \text{if } j \text{ is odd} \\ \\ u_1 + u_3 + \ldots + u_{j-1} \\ +u_{j+2} + u_{j+4} + \ldots + u_k \\ +kE/2 - (E - S - u_j) & \text{if } j \text{ is even} \end{cases}$$

Similarly, W_{j+1} is the minimum tight packing width when the character sequence is (c_{j+1}, c_1, c_2, $c_3, \ldots, c_j$, c_{j+2}, c_{j+3}, $c_{j+4}, \ldots, c_k$). So,

$$W_{j+1} = \begin{cases} u_1 + u_3 + \ldots + u_{j-2} + u_j \\ +u_{j+3} + u_{j+5} + \ldots + u_k \\ +kE/2 - (E - S - u_{j+1}) & \text{if } j \text{ is odd} \\ \\ u_1 + u_3 + \ldots + u_{j-1} \\ +u_{j+2} + u_{j+4} + \ldots + u_k \\ +kE/2 - (E - S - u_{j+1}) & \text{if } j \text{ is even} \end{cases}$$

It implies that

$$\begin{aligned} W_j - W_{j+1} &= \begin{cases} 0 & \text{if } j \text{ is odd} \\ u_j - u_{j+1} & \text{if } j \text{ is even} \end{cases} \\ &\geq 0 \end{aligned}$$

Hence, we have $W_1 \geq W_2 \geq \ldots \geq W_k$ again. ∎

For the example in Fig. 10, $u_1 = 8$, $u_2 = 7$, $u_3 = 5$ and $u_4 = 2$. According to Fig. 10, $W_1 (= 28) \geq W_2 (= 28) \geq W_3 (= 26) \geq W_4 (= 26)$. This example confirms Lemma 6.

IV. CASCO: Algorithm for Character and Stencil Co-optimization

Here we present our main algorithm, CASCO, for character and stencil co-optimization. We want to maximize the shot saving with character projection by selecting appropriate characters, placing them properly in their effective printing regions, and packing them tightly onto a stencil.

The shot saving of printing a character c by CP instead of VSB is $r_c(n_{VSB_C}-1)$ where r_c is the number of occurrence of character c in the circuit and n_{VSB_c} is the number of e-beam shots required to print c by VSB. We define the *efficiency* of a character c as its shot saving per unit effective width, i.e., $2r_c(n_{VSB_C}-1)/(w_c+E)$. Recall that the effective width of character c is $(w_c+E)/2$.

We will utilize a simple condition for checking whether a set of characters C can be packed within a row without exceeding the stencil width W in our algorithm. Suppose

$$\sum_{c\in C}(w_c+E)/2-(E-w_{min(C)})/2\le W \qquad (3)$$

where $min(C)$ is the minimum width character in C, then all characters in C can be packed within a row of the stencil. The above is a sufficient condition since Lemma 5 and Equation (1) imply that there exists a tight packing of C with width no more than $\sum_{c\in C}(w_c+E)/2-(E-S-w_{min(C)})/2-S/2$ if we use $min(C)$ as the first character and set the width of its left blank space to $E-S-w_{min(C)}$.

The CASCO algorithm is shown in Algorithm 1. We process each character in decreasing order of efficiency in the for loop from lines 3 to 12. The characters are inserted into the stencil row by row until all rows are used up. It is worthy to note that in lines 5 and 7, we simply insert a character into a row without ordering or constructing any packing for the characters in the row. But the character insertion criterion in line 4 can ensure that we will never put too many characters into a row. Only in line 13, we construct a tight packing for the characters inserted into each row according to Lemma 5 with the row's minimum width character as its first character as suggested by Lemma 6. The final for loop from lines 14 to 16 attempts to greedily pack one extra character to the end of each row to increase the shot saving.

Our next lemma shows a performance guarantee of CASCO.

Lemma 7: The shot saving by CASCO is more than $(W-w_{max})/(W+E-S-w_{min})$ of the optimal, where w_{max} and w_{min} are the maximum width and minimum width over all characters, respectively.

Proof: To prove our lemma, we first show that for any optimal solution, the total effective width of all characters cannot exceed $R(W+E-S-w_{min})$. Second, we argue that the total effective width of all characters picked by CASCO upon exiting the for loop at line 12 already exceeds $R(W-w_{max})$.

1. We want to show that for any optimal solution, the total effective width of all characters is no more than $R(W+E-S-w_{min})$. Suppose O^* is an optimal solution. Let P_j^* be the packing of characters in row j of O^*.

By (2), $W(P_j^*)$ is at least $\sum_{c\in P_j^*}(w_c+E)/2-a_j^*/2-b_j^*/2$ where a_j^* is the width of the left blank space of the first character and b_j^* is the width of the right blank space of the last character in P_j^*. Now, both a_j^* and b_j^* must be less than $E-S-w_{min}$. So, $W(P_j^*)$ is at least $\sum_{c\in P_j^*}(w_c+E)/2-(E-S-w_{min})$.

On the other hand, $W(P_j^*)$ must be no more than the stencil width W. Hence, $\sum_{c\in P_j^*}(w_c+E)/2-(E-S-w_{min})\le W(P_j^*)\le W$ which implies $\sum_{c\in P_j^*}(w_c+E)/2\le W+(E-S-w_{min})$. As a result, $\sum_{c\in O^*}(w_c+E)/2\le R[W+(E-S-w_{min})]$. Therefore, the total effective width of all characters in O^* is bounded by $R(W+E-S-w_{min})$.

Algorithm 1 CASCO

```
1:  Sort all characters in decreasing order of efficiency;
2:  $j = 1$;
3:  for each character $x$ in sorted order do
4:      if $x$ can be inserted into row $j$ according to Equation (3) then
5:          Insert $x$ into row $j$;
6:      else if $j < R$ then
7:          Insert $x$ into row $j+1$;
8:          $j = j+1$;
9:      else
10:         break;
11:     end if
12: end for
13: For each row, construct a tight packing by putting its minimum width character $f$ first and then other characters in decreasing width, and setting the left blank space of $f$ to $E-S-w_f$;
14: for each row $j$ do
15:     Tightly pack at the end of row $j$ an unused character with largest possible shot saving without exceeding the stencil width, if possible;
16: end for
```

2. We want to show that the total effective width of all characters picked by CASCO upon exiting the for loop at line 12 is more than $R(W-w_{max})$. Let C_j be the set of characters in row j upon exiting the for loop at line 12 of CASCO.

We claim that for each row j, $\sum_{c\in C_j}(w_c+E)/2>W-w_{max}$. Otherwise, assume to the contrary that $\sum_{c\in C_{j'}}(w_c+E)/2\le W-w_{max}$ for some row j'. Let c' be the first character following those characters inserted into row j' in the sorted list of line 1. If $\sum_{c\in C_{j'}}(w_c+E)/2\le W-w_{max}$, then $\sum_{c\in C_{j'}}(w_c+E)/2+(w_c'+E)/2-(E-w_{min(C_{j'}\cup\{c'\})})/2$ $\le (W-w_{max})+(w_c'+E)/2-(E-w_{min(C_{j'}\cup\{c'\})})/2$ $=W-(2w_{max}-w_{c'}-w_{min(C_{j'}\cup\{c'\})})/2\le W$. Therefore, the character insertion condition in Equation (3) would have been true and c' would have been inserted into $C_{j'}$ by line 5. Hence, $\sum_{c\in C_{j'}}(w_c+E)/2$ must be larger than $W-w_{max}$.

So, $\sum_{c\in C_j}(w_c+E)/2>W-w_{max}$ for all row j upon exiting the for loop at line 12 of CASCO. Hence, the total effective width of all characters picked by CASCO upon exiting the for loop at line 12 exceeds $R(W-w_{max})$.

By parts 1 and 2 above, and the fact that all characters picked by CASCO before exiting the for loop at line 12 have higher shot saving per unit effective width than any other characters, we can conclude that the shot saving by CASCO is more than $(W - w_{max})/(W + E - S - w_{min})$ of the optimal. ■

V. Experimental Results

We have implemented our algorithm CASCO in C and tested it on a Linux server with a 2.67 GHz Intel processor and 47 GB of memory. We compare CASCO with the methods in [8] and [9] on stencil design for standard cell-based circuits. We have obtained executable code and benchmarks from the authors of [8] and [9].

In the first experiment, benchmarks 1D-1 to 1D-4 from [9] were used. Each benchmark contains 1000 character candidates and the available area of the stencil is $1000\mu m \times 1000\mu m$. Before we can use their benchmarks, we need to define the effective printing region (or equivalently, the safety margin) of each benchmark. For each benchmark, we set the effective printing region to be the smallest value possible (i.e., the safety margin to be the biggest value possible) such that all original character patterns are still within the effective printing region.[2]

In Table I, we report the values of E and S for each benchmark as well as the results of upper bound[3], [8], [9], and CASCO. Columns 4, 6, 9, and 12 report the number of shots that can be saved if CP is also used. Columns 5, 7, 10, and 13 report the number of characters that can be packed in the stencil.[4] The runtimes of [8], [9], and CASCO are listed in columns 8, 11, and 14, respectively. It can be seen that CASCO achieves near optimal shot saving on all four benchmarks. Moreover, as CASCO exploits (1) the flexibility of placing a character pattern anywhere within its enclosing effective printing region to increase the amount of blank space sharing between adjacent characters and (2) the flexibility that the blank space of a character can lie outside of the available character area of the stencil, the number of characters packed in the stencil is about 4% to 5% more than [8] and [9]. It also increases the shot saving by about 11% and 3% compared to [8] and [9], respectively. Finally, CASCO is orders of magnitude faster than the other two methods.

Besides the four benchmarks from [9], we created some harder benchmarks (1D-1h to 1D-4h) for more testing. We randomly generated 200 extra character candidates with characteristics similar to other character candidates into each of the original benchmarks while keeping the stencil size unchanged. The results are shown in Table II. The shot savings by CASCO are again less than 1% away from the upper bound values. Comparing the results in Table II to the results in Table I, we can observe that the solution quality of [8] degraded with harder benchmarks while the runtime of [9] increased quickly with harder benchmarks. However, both the solution quality and runtime are very stable for CASCO irrespective of the difficulty of the benchmarks.

In Table III, we show the effectiveness of CASCO in reducing the writing time of e-beam lithography, which is proportional to the number of shots needed. For the benchmarks from [8], the writing time of CASCO is $25.97\times$ shorter than a system using VSB only, $2.98\times$ shorter than that of [8], and $1.59\times$ shorter than that of [9]. For the harder benchmarks, the writing time of CASCO is $9.38\times$ shorter than a system using VSB only, $2.25\times$ shorter than that of [8], and $1.29\times$ shorter than that of [9]. For the harder benchmarks, inherently a smaller fraction of all characters can be put in the stencil and printed by CP, so the ratios are reduced compared to the original benchmarks.

	#shots needed			
Circuit	VSB only	[8]	E-BLOW [9]	CASCO
1D-1	770543	50809	27636	14491
1D-2	770543	93465	44263	27812
1D-3	770543	152376	78704	57698
1D-4	770543	193494	107460	79930
Normalized	25.97	2.98	1.59	1.00
1D-1h	922770	165471	85153	61167
1D-2h	922770	210722	117370	88986
1D-3h	922770	279757	169347	137811
1D-4h	922770	327572	207009	171594
Normalized	9.38	2.25	1.29	1.00

TABLE III
Number of shots needed by four different methods on benchmarks from [9] and on harder benchmarks.

VI. Conclusions

We have introduced a new character and stencil co-optimization problem to maximize the shot saving for the method of character projection in e-beam lithography. We have proved the NP-completeness of the problem and provided an elegant approximation algorithm CASCO to solve the problem nearly optimally and very efficiently. CASCO performs much better than the published state of the art [8] and [9] mainly because it exploits the flexibility in the location of a character pattern within its effective printing region. Finally, we note that CASCO can be applied to the stencil design of a multi-beam system [1], [2]. In a multi-beam system, multiple beam columns furnished with their own stencils write different regions of a wafer in parallel. The stencils for all beam columns should be exactly the same and should not be different from that of a single beam system since identical dies are manufactured on a wafer.

Appendix: NP-Completeness Proof For SRCP

In this appendix, we prove that the decision version of the Single-Row Character Packing problem is NP-complete.

[2]In other words, S is set to the minimum original blank space on both sides over all characters.

[3]Since we have shown in the proof of Lemma 7 that the total effective width of all characters in an optimal solution cannot exceed $R(W+E-S-w_{min})$. An upper bound on shot saving can be obtained by a linear relaxation of a 0-1 knapsack problem with capacity $R(W + E - S - w_{min})$ such that the profit and weight of each item correspond to the shot saving and effective width of each character in our problem.

[4]As the authors of [8] and [9] have given us the updated versions of their codes, their results here are slightly better than those reported in [9].

Circuit			Upper Bound		[8]			E-BLOW [9]			CASCO		
Name	$E(\mu m)$	$S(\mu m)$	#shot saved	#char.	#shot saved	#char.	CPU(s)	#shot saved	#char.	CPU(s)	#shot saved	#char.	CPU(s)
1D-1	3.8	0.2	758577.9	986.2	719734	926	12.51	742907	935	2.66	756052	970	<0.0005
1D-2	4.0	0.3	745922.5	912.5	677078	854	10.49	726280	863	2.13	742731	898	<0.0005
1D-3	4.2	0.3	717776.8	805.1	618167	749	8.10	691839	758	3.45	712845	791	<0.0005
1D-4	4.4	0.3	696221.5	746.5	577049	687	6.78	663083	699	3.63	690613	734	<0.0005
Normalized			100.00%	100.00%	88.66%	93.14%		96.73%	94.29%		99.44%	98.34%	

TABLE I
COMPARISON OF SHOT SAVING AND CHARACTERS PACKED BY CASCO AGAINST AN UPPER BOUND, [8] AND E-BLOW [9] ON BENCHMARKS FROM [9].

Circuit			Upper Bound		[8]			E-BLOW [9]			CASCO		
Name	$E(\mu m)$	$S(\mu m)$	#shot saved	#char.	#shot saved	#char.	CPU(s)	#shot saved	#char.	CPU(s)	#shot saved	#char.	CPU(s)
1D-1h	3.8	0.2	866746.5	988.7	757299	926	12.39	837617	938	5.58	861603	974	<0.0005
1D-2h	4.0	0.3	840517.7	914.8	712048	851	13.35	805400	865	4.35	833784	899	<0.0005
1D-3h	4.2	0.3	792379.0	808.6	643013	752	10.08	753423	763	3.81	784959	796	<0.0005
1D-4h	4.4	0.3	759893.0	748.6	595198	685	7.27	715761	699	13.23	751176	735	<0.0005
Normalized			100.00%	100.00%	82.89%	92.80%		95.43%	94.29%		99.13%	98.35%	

TABLE II
COMPARISON OF SHOT SAVING AND CHARACTERS PACKED BY CASCO AGAINST AN UPPER BOUND, [8] AND E-BLOW [9] ON HARDER BENCHMARKS.

Lemma 1: Determining if a set of characters can all be packed into a single row of width W is NP-complete.

Proof: First, the given problem is obviously in NP. Next, we show that the partition problem [14], which is NP-complete, polynomially transforms to the SRCP problem. The partition problem is stated as follows. Given positive integers v_1, v_2, …, v_k, is there a subset $T \subseteq \{1, 2, \ldots, k\}$ such that $\sum_{j\in T} v_j = V/2$ where $V = \sum_{j=1}^{k} v_j$?

Given a partition instance, we define a SRCP instance with $2k+2$ characters where k of them have width v_1, v_2, …, v_k, and the remaining have width 0. The enclosure width E is set to V and the safety margin S is set to 0. We now claim that there is a subset $T \subseteq \{1, 2, \ldots, k\}$ such that $\sum_{j\in T} v_j = V/2$ if and only if there is a packing of all $2k+2$ characters with length $(V + 2kE)/2$.

($\Leftarrow$) Suppose there exists a packing of the characters with length $(V + 2kE)/2$. Assume in that packing, the character widths from left to right are $w_1, w_2, \ldots, w_{2k+2}$. Note that the distance d_1 from the left of character 1 to the right of character $2k+1$ is at least $w_1+w_3+\ldots+w_{2k+1}+kE$, while the distance d_2 from the left of character 2 to the right of character $2k+2$ is at least $w_2 + w_4 + \ldots + w_{2k+2} + kE$ as shown in Fig. 11. Hence, $d_1 + d_2 \geq V + 2kE$. Now, the length of the packing is the distance from the left of character 1 to the right of character $2k+2$ which must be greater than both d_1 and d_2. By our assumption, the length of the packing is $(V+2kE)/2$, which implies $(V+2kE)/2 \geq d_1$ and $(V + 2kE)/2 \geq d_2$, and so $V + 2kE \geq d_1 + d_2$. For all the above to be true, d_1, $w_1+w_3+\ldots+w_{2k+1}+kE$, d_2, and $w_2+w_4+\ldots+w_{2k+2}+kE$ must all be equal to $(V+2kE)/2$. Therefore, $w_1+w_3+\ldots+w_{2k+1}+kE = (V+2kE)/2$, which means $w_1 + w_3 + \ldots + w_{2k+1} = V/2$. In other words, there exists a subset T such that $\sum_{j\in T} v_j = V/2$ for the given partition instance.

($\Rightarrow$) Suppose there exists a subset T such that $\sum_{j\in T} v_j = V/2$ for the given partition instance. We can construct a packing of the characters with length $(V+2kE)/2$ as follows.

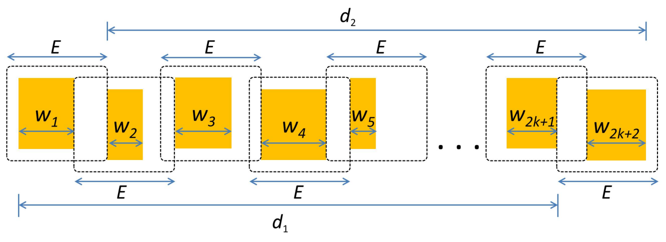

Fig. 11. A packing of $2k+2$ characters.

Let $w_1, w_3, \ldots, w_{2k+1}$ be the values of v_j for all $j \in T$ together with $k+1-|T|$ 0's such that $0 = w_1 \leq w_3 \leq \ldots \leq w_{2k+1}$. Let $w_2, w_4, \ldots, w_{2k+2}$ be the values of v_j for all $j \in \overline{T}$ together with $k+1-|\overline{T}|$ 0's such that $w_2 \geq w_4 \geq \ldots \geq w_{2k+2} = 0$. Let s_0, the width of the left blank space of the first character, be E.

According to Lemma 4, we can construct a tight packing in the order $w_1, w_2, \ldots, w_{2k+2}$ if the following conditions are satisfied. (Note that the safety margin S is 0 by our assumption here.)

$$\begin{aligned}
s_0 &\geq 0 \\
E &\geq s_0 + w_1 \\
s_0 + w_1 &\geq w_2 \\
E + w_2 &\geq s_0 + w_1 + w_3 \\
s_0 + w_1 + w_3 &\geq w_2 + w_4 \\
E + w_2 + w_4 &\geq s_0 + w_1 + w_3 + w_5 \\
s_0 + w_1 + w_3 + w_5 &\geq w_2 + w_4 + w_6 \\
&\vdots \\
E + w_2 + w_4 + \ldots + w_{2k} &\geq s_0 + w_1 + w_3 + \ldots + w_{2k+1}
\end{aligned}$$

$$s_0 + w_1 + w_3 + \ldots + w_{2k+1} \;\; \geq \;\; w_2 + w_4 + \ldots + w_{2k+2}$$

Since $s_0 = E = V = \sum_{j=1}^{k} v_j = \sum_{j=1}^{2k+2} w_j$, all the inequalities above with s_0 on the left hand side must be satisfied. Moreover, since $s_0 = E$, the rest of the inequalities can be re-written as

$$w_2 + w_4 + \ldots + w_{2j} \geq w_1 + w_3 + \ldots + w_{2j+1} \qquad (4)$$

for $j = 0, 1, \ldots, k$. Note that $w_2 + w_4 + \ldots + w_{2k+2} = w_1 + w_3 + \ldots + w_{2k+1} = V/2$. And $w_{2k+2} = 0$ by our assumption, which implies $w_2 + w_4 + \ldots + w_{2k} = w_1 + w_3 + \ldots + w_{2k+1}$. In other words, Equation (4) holds for $j = k$. In addition, since $w_2 \geq w_4 \geq \ldots \geq w_{2k}$ and $w_1 \leq w_3 \leq \ldots \leq w_{2k+1}$, by induction we can show that Equation (4) holds for $j = k-1, k-2, \ldots, 1$. Finally, Equation (4) also holds for $j = 0$ since $w_1 = 0$ by assumption. Therefore, all the conditions above are satisfied and by Lemma 4 we can construct a tight packing in the order $w_1, w_2, \ldots, w_{2k+2}$.

Moreover, the width of the tight packing is $w_2 + w_4 + \ldots + w_{k+2} + (2k+2)E/2 - s_0 = V/2 + (2k+2)E/2 - E = V/2 + kE = (V + 2kE)/2$. ■

References

[1] T. Maruyama, Y. Machida, and S. Sugatani. CP based EBDW throughput enhancement for 22nm high volume manufacturing. In *Proc. of SPIE 7637*, pages 76371S–1–76371S–8, Feb. 2010.

[2] T. Maruyama et al. CP element based design for 14nm node EBDW high volume manufacturing. In *Proc. of SPIE 8323*, pages 832314–1–832314–11, Apr. 2012.

[3] B. J. Lin. Future of multiple-E-beam direct-write systems. In *Proc. of SPIE 8323*, pages 832302–1–832302–11, March 2012.

[4] H. Pfeiffer. Variable spot shaping for electron-beam lithography. *Journal of Vaccum Sci. and Tech.*, 15(3):887–890, May 1978.

[5] H. Pfeiffer. Recent advances in electron-beam lithography for the high-volume production of VLSI devices. *IEEE Trans. on Electron Devices*, 26(4):663–674, Apr. 1979.

[6] R. Inanami, S. Magoshi, S. Kousai, M. Kamada, T. Takayanagi, K. Sugihara, Katsuya Okumura, and T. Kuroda. Throughput enhancement strategy of maskless electron beam direct writing for logic device. In *Proc. of International Electron Devices Meeting*, pages 833–836, Dec. 2000.

[7] 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 electron beam direct-write lithography systems. In *Proc. of International Symposium on System-on-Chip*, pages 137–140, Nov. 2005.

[8] K. Yuan, B. Yu, and D. Z. Pan. E-beam lithography stencil planning and optimization with overlapped characters. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 31(2):167–179, Feb. 2012.

[9] B. Yu, K. Yuan, J.-R. Gao, and D.Z. Pan. E-BLOW: e-beam lithography overlapping aware stencil planning for MCC system. In *Proc. of Design Automation Conference*, 2013.

[10] 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 *Proc. of Asia and South Pacific Design Automation Conference*, pages 725–730, Jan. 2012.

[11] R. Ikeno, T. Maruyama, T. Iizuka, S. Komatsu, M. Ikeda, and K. Asada. High-throughput electron beam direct writing of via layers by character projection using character sets based on one-dimensional via arrays with area-efficient stencil design. In *Proc. of Asia and South Pacific Design Automation Conference*, pages 255–260, Jan. 2013.

[12] K. Yoshida, T. Mitsuhashi, S. Matsushita, L. L. Chau, T. D. T. Nguyen, D. MacMillen, and A. Fujimur. Stencil design and method for improving character density for cell projection charged particle beam lithography. US Patent, Dec. 31, 2009.

[13] A. Fujimura. Design for E-beam: Design insights for direct-write maskless lithography. In *Proc. of SPIE 7823*, pages 137–140, Sep. 2010.

[14] Richard Karp. Reducibility among combinatorial problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103, 1972.

PLACE
PHOTO
HERE

Wai-Kei Mak Wai-Kei Mak received the B.S. degree from the University of Hong Kong, Hong Kong, in 1993, and the M.S. and Ph.D. degrees from the University of Texas at Austin, U.S., in 1995 and 1998, respectively, all in computer science.

Dr. Mak is now a Professor with the Department of Computer Science, National Tsing Hua University, Taiwan. He was Assistant Professor with the Department of Computer Science and Engineering, University of South Florida, U.S., from 1999 to 2003. His current research interests include VLSI physical design automation, and CAD for field-programmable technologies.

Dr. Mak's lab won the first place at the FPT 2008 Logic Block Clustering Contest, the third place at the IEEE CEDA PATMOS 2011 Timing Analysis Contest, and the second place at the TAU 2013 Variation-Aware Timing Analysis Contest. He has served on the Program and/or the Organizing Committee of Asia South Pacific Design Automation Conference, the International Conference on Field Programmable Logic and Applications, and the International Conference on Field-Programmable Technology (FPT). He was the Technical Program Chair of FPT in 2006 and was the General Chair of the same conference in 2008. Since 2009, he has been a Steering Committee Member of the International Conference on Field-Programmable Technology.

PLACE
PHOTO
HERE

Chris Chu Chris Chu received the B.S. degree in computer science from the University of Hong Kong, Hong Kong, in 1993. He received the M.S. degree and the Ph.D. degree in computer science from the University of Texas at Austin in 1994 and 1999, respectively.

Dr. Chu is a Professor in the Electrical and Computer Engineering Department at Iowa State University. His area of expertises include CAD of VLSI physical design, and design and analysis of algorithms.

Dr. Chu is currently an associate editor for IEEE TCAD and for ACM TODAES. He has served on the technical program committees of several major conferences including DAC, ICCAD, ISPD, ISCAS, DATE, ASP-DAC, and SLIP.

Dr. Chu received the IEEE TCAD best paper award at 1999 for his work in performance-driven interconnect optimization. He received another IEEE TCAD best paper award at 2010 for his work in routing tree construction. He received the ISPD best paper award at 2004 for his work in efficient placement algorithm. He received another ISPD best paper award at 2012 for his work in floorplan block shaping algorithm. He received the Bert Kay Best Dissertation Award for 1998-1999 from the Department of Computer Sciences in the University of Texas at Austin. He is a Fellow of IEEE.