Lossless Compression Algorithm Based on Dictionary Coding for Multiple E-Beam Direct Write System

Pei-Chun Lin[1], Yu-Hsuan Pai[1], Yu-Hsiang Chiu[1], Shao-Yuan Fang[2] and Charlie Chung-Ping Chen[1]

[1]Graduate Institute of Electronics Engineering, National Taiwan University

No. 1, Sec. 4, Roosevelt Road, Taipei, 10617 Taiwan.

[2]Department of Electrical Engineering, National Taiwan University of Science and Technology

No.43, Sec. 4, Keelung Rd., Da'an Dist., Taipei 106, Taiwan.

ABSTRACT- **Electron-beam direct-write (EBDW) lithography is an attractive candidate of next-generation lithography in advanced semiconductor processes. The huge data stream bandwidth required for the data delivery path in EBDW systems could seriously deteriorate throughput, which is one of the major deficiencies constraining EBDW lithography from mass production. A lossless electron-beam layout data compression and decompression algorithm is proposed in this paper for 5-bit gray level bitmaps. Compared with the state-of-the-art LineDiff Entropy algorithm, the proposed method averagely improves the compression rate by 18% and achieves more than 7.5 times speedup for decompression.**

***Index Terms*—Design for Manufacturability, Electron-Beam Lithography, Data Compression**

1. Introduction and Related Work

As integrated circuit (IC) process nodes continue to shrink to 14 nm and below, the IC industry will face severe manufacturing challenges with conventional optical lithography technologies. EBDW lithography is an attractive candidate of next-generation lithography. There are two primary advantages of e-beam lithography. One is that it can pattern a layout with sub-nm resolution, and the other is that it can avoid extremely high photomask fabrication cost. However, the relatively low throughput restricts the development of EBDW. Due to this problem, there have been several multiple e-beam direct-write (MEBDW) systems developed by MAPPER [1] and KLA-Tencor [2] in recent years to achieve fast scanning by using massive and parallel e-beam writers. Once the throughput reaches more than 70 wafers per hour (WPH), the EBDW could be the main stream of the lithography.

As shown in Table. 1, we can estimate the data volume and required data transmission rate for the 14-nm technology node. From [3] we can get our device specifications shown below: the wafer size is 300 mm in the diameter and the expected throughput is 70 wafers per hour, which asks about 50 seconds to write a single layer of a wafer. The digital pattern generator (DPG) in an MEBDW system is a chip of size less than 26×33 mm^2, so the number of optical fibers connected to DPG is limited. The data transmission rate of a single fiber is about 10 Gb/s [4]. Under the specification, the required data transmission rate could be more than 144 Tb/s. To meet such a high rate, more than ten thousand fibers are required, which is much more than that a DPG can afford. Therefore, layout data must be compressed before being transferred to MEBDW systems. Besides, the compression rate requirement of the compression method, the decompression speed of the compression method is also a critical issue [3].

Table. 1 Specification of data transmission rate.

Device Specification		Maskless Process Specification	
Wafer size	300 mm	Pixel size	7 * 7 nm^2
Writing rate	70 WPH	Pixel depth	5 bits
Writing time	50 s	Wafer data size	7200 Tb
Optical fiber transmission rate	10 Gbps	Required rate	144 Tbps
Connected fibers	32	Required fibers	14400
450 times exceeded			

There have been some existing studies on layout data compression for different MEBDW systems. The state-of-the-art LineDiff Entropy algorithm outperforms others for 5-bits gray-level bitmaps [4]. There are two criteria used to determine the performance of layout data compression algorithms: the compression ratio and the decompression speed.

1.1 Data Preparation for 5-bit gray level bitmaps

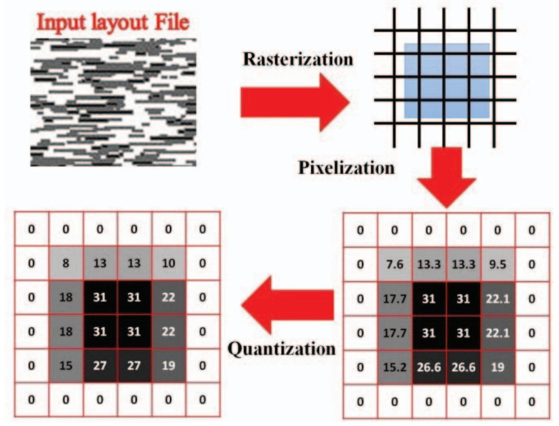


Fig. 1 Data processing for MEBDW system.

According to the specification of the MEBDW system proposed by KLA-Tencor, the layout data are transformed into 5-bit gray level bitmaps. Fig. 1 shows the bitmap data processing procedures for the MEBDW system. The first step is called rasterization, where a circuit layout is rasterized according to the e-beam pixel size, and the pixel size is about the half of the minimum feature size. Then, each pixel is normalized by the ratio of the pattern area to the pixel area. The interior pixels of a layout pattern are normalized to 31. These procedures are called pixelization and quantization.

1.2 LineDiff Entropy

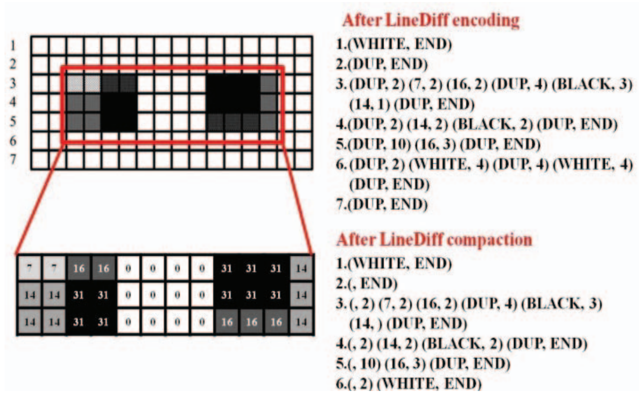
LineDiff Entropy [4] performs well in bitmap data compression and decompression, and its simple structure is also elegantly attractive. This algorithm is composed of three main steps: LineDiff Encoding, LineDiff Compaction, and Entropy Encoding, which are briefly introduced below.

The first step is LineDiff Encoding. Line N will be encoded by some pairs in the form of (OP, L), and either OP or L is defined by comparison to line N-1 in the same position. If the data being encoded is the same as data at the same position in the previous row, we can just encode it by setting OP to DUP (duplication). Otherwise, OP is the color of these pixels, encoded by black, white, or a 5-bit binary number. On the other hand, L stands for length, which is set to be the length of data that maintains this OP or set to be END to represent this OP will last to the end of this line.

The following step is called LineDiff Compaction. This step is the procedure to furthermore compress file size by omitting unnecessary data. There are three rules of LineDiff Compaction:

1) If any (OP, L) pair has L = 1, omit L.
2) If the first (OP, L) pair has OP = DUP, omit OP.
3) If consecutive pairs have the same pixel value, combine them.

The first rule has the highest priority and the third one has the lowest priority. Fig. 2 shows the example of LineDiff encoding and LineDiff compaction [4]. And the last step is the Entropy Encoding. The codes are defined according to the analysis of occurrence frequencies [4].


After LineDiff encoding

1.(WHITE, END)
2.(DUP, END)
3.(DUP, 2) (7, 2) (16, 2) (DUP, 4) (BLACK, 3) (14, 1) (DUP, END)
4.(DUP, 2) (14, 2) (BLACK, 2) (DUP, END)
5.(DUP, 10) (16, 3) (DUP, END)
6.(DUP, 2) (WHITE, 4) (DUP, 4) (WHITE, 4) (DUP, END)
7.(DUP, END)

After LineDiff compaction

1.(WHITE, END)
2.(, END)
3.(, 2) (7, 2) (16, 2) (DUP, 4) (BLACK, 3) (14,) (DUP, END)
4.(, 2) (14, 2) (BLACK, 2) (DUP, END)
5.(, 10) (16, 3) (DUP, END)
6.(, 2) (WHITE, END)

Fig. 2 Example of LineDiff encoding and LineDiff compaction [4].

Both the encoding and decoding processes in LineDiff Entropy run in linear time because only one scan of all data is required. In addition, it has better performance compared with other algorithms in terms of both the data compression ratio and the decompression time. Therefore, our algorithm mainly modifies and improves the LineDiff algorithm to further enhance its performance.

2. Lossless Dictionary-based Compression Algorithm

In this chapter, we will introduce our new algorithm which has higher compression factor and better decompression efficiency.

2.1 Modified LZ77 with 2-tail

We take the compression algorithm LZ77 as our prototype and address the features extracted from the 5-bit gray level bitmap to enhance the compression ratio.

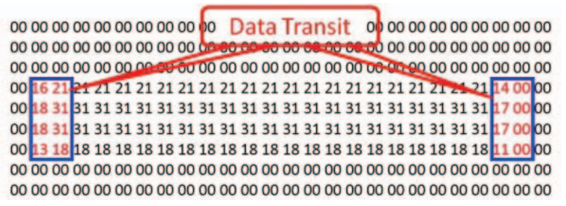


Fig. 3 Transition characteristics of bitmaps.

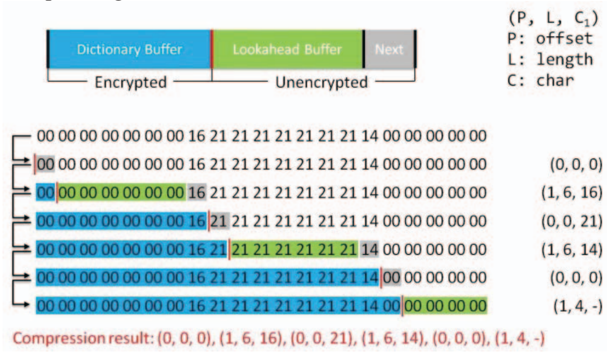
Unlike the bitmaps of general pictures, 5-bit gray level bitmaps for layout data have regular patterns. For example, while tracing a bit line, pixels with the value 0 continue until a feature is encountered. The first pixel on the left boundary of the feature usually has a value smaller than 31. Then, the first pixel is followed by a series of pixels with value 31 or a value larger than the first pixel. Finally, the series of non-empty pixels should end with a pixel on the right boundary of the feature with another value smaller than 31. The bit transition characteristics are highlighted in Fig. 3.

The original LZ77 [6], after synchronizing the match characters, records only the first one character of the unencrypted stream. As shown in Fig. 4, LZ77 compresses the data into several terms (P, L, C1), where P is the offset between current character and the match in the dictionary buffer, L is the length of the identical pairs, and C1 is the first character of the unencrypted stream. We replace the feature by recording the first two characters of the unencrypted stream and name it as the 2-tail method.

Another characteristic of the bitmap is that the copy operation always copies the last character, unless the unencrypted stream meets the boundaries of feature. So we can simply set the dictionary buffer size to 1. It leads to an efficient way to compress the entire bitmap data by just looking back the last character of encrypted stream and determining whether the copy operation should be applied or not.

Fig. 4 and Fig. 5 shows an example of two algorithms, the original LZ77 and our modified LZ77, compressing the same stream extracted from the bitmap data. Obviously, the modified LZ77 uses fewer terms than the original one does to compression the same stream. In other words, this modification results in the higher compression ratio. The size of the data compressed by our algorithm is about half of the data size compressed by the original LZ77. Such a huge improvement could contribute to the competitiveness of MEBDW lithography.

Another advantage of our algorithm over the LineDiff Entropy is that every term (P, L, C1, C2), the coding format in our algorithm which shows in Fig. 5, is only relative to the last term in the decompression process. This characteristic makes our algorithm has higher decompression rate, because LineDiff Entropy has to look back the previous scanline while decompressing data.



```
00 00 00 00 00 00 00 16 21 21 21 21 21 21 21 14 00 00 00 00 00
00 00 00 00 00 00 00 16 21 21 21 21 21 21 21 14 00 00 00 00 00    (0, 0, 0)
00 00 00 00 00 00 00 16 21 21 21 21 21 21 21 14 00 00 00 00 00    (1, 6, 16)
00 00 00 00 00 00 00 16 21 21 21 21 21 21 21 14 00 00 00 00 00    (0, 0, 21)
00 00 00 00 00 00 00 16 21 21 21 21 21 21 21 14 00 00 00 00 00    (1, 6, 14)
00 00 00 00 00 00 00 16 21 21 21 21 21 21 21 14 00 00 00 00 00    (0, 0, 0)
00 00 00 00 00 00 00 16 21 21 21 21 21 21 21 14 00 00 00 00 00    (1, 4, -)
Compression result: (0, 0, 0), (1, 6, 16), (0, 0, 21), (1, 6, 14), (0, 0, 0), (1, 4, -)
```

Fig. 4 Original LZ77 compression example.

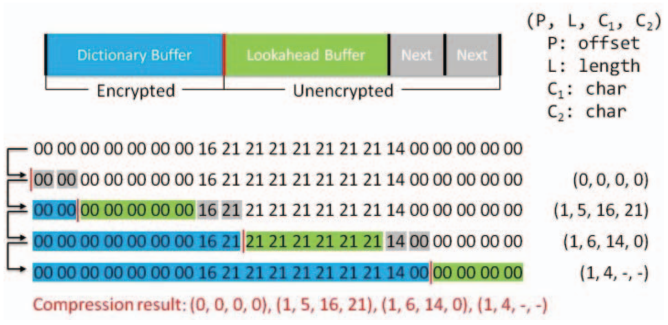


Fig. 5 Modified LZ77 compression example.

2.2 *Omit Offset*

After the 2-tail method, we can find that all the offsets (the first value of each term) are "1" except the beginning term of each scanline. We can make an initial state by recording the first few characters in the beginning of the scanline. So the offset can be omitted.

According to the LineDiff Entropy specification, we split an entire bitmap into stripes with 1024-pixel width. Due to the analysis of the split data, there are three different situations as shown in the Fig. 6.

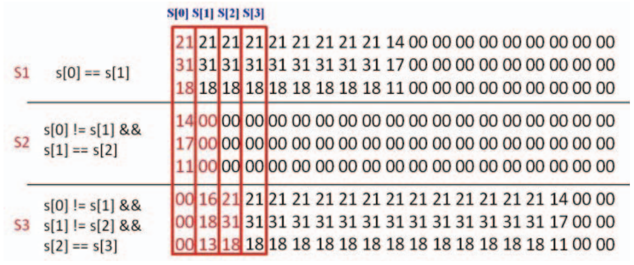


Fig. 6 Three situations of trimming the bitmap.

The first situation is show in the first three highlighted rows. We cut the bitmap in the middle of the consecutive value of the stream, so we only have to record the first character in the beginning of the row, and we can start the copy from previous by offset 1, till the end of the row. Second, we cut the bitmap at the boundary of the edge, we have to record the first two characters in the beginning of the row. In the third situation, we have to record the first three characters, and the rest of the stream can apply the copy behavior with the offset 1.

2.3 *Copy-line*

The second feature we added to our algorithm is the idea inspired from the LineDiff Entropy algorithm. In LineDiff Entropy, the duplication can be applied to any length of equivalent pairs [4]. In our work, we only allow duplicating the entire scanline, called copy-line. As shown in Fig. 7, all the blue arrows indicate the scanlines which can apply the copy-line instruction.

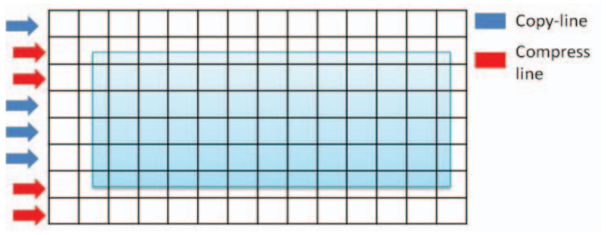


Fig. 7 Blue region can be compressed by copy-line.

2.4 *Huffman-like Coding*

After the bitmap data has been compressed by the above algorithms, the compression ratio can be further improved by applying a Huffman-like encoding. The characteristic of the Huffman coding is that the higher the frequency is, the shorter the code will be. Therefore, it can help us further improve the compression factor.

To define the Huffman-like coding in this work, there are 4 situations at the beginning of each scanline. S0 represents copying from the previous scanline. As Fig. 6 shows, S1, S2 and S3 define the situations at the beginning of each scanline, and the rest of the scanline should be encoded by Modified LZ77. The copy length L in Modified LZ77 can be split into two groups: length less than or equal to 31 and length larger than 31. Following L are the characters C_1 and C_2 which should be remained.

Table. 2 Relative frequency of occurrence.

Beam Size	Beginning of each scanline				Copy-Line length	
	S0	S1	S2	S3	L≦31	L>31
$(7nm)^2$	85%	14.8%	0.2%	0%	71.4%	28.6%

Table. 3 Complete code for the beginning of each scanline.

OPERATION	FORMAT	Bits
S0	1	1
S1	01 + [5-bit]	7
S2	001 + [5-bit] + [5-bit]	13
S3	000 + [5-bit] + [5-bit] + [5-bit]	18

Table. 4 Complete code for the Modified LZ77.

OPERATION	FORMAT	Bits
$L \leq 31$	$0 + L[5\text{-bit}] + C_1[5\text{-bit}] + C_2[5\text{-bit}]$	16
$L > 31$	$1 + L[10\text{-bit}] + C_1[5\text{-bit}] + C_2[5\text{-bit}]$	21

Table. 3 shows the frequency of each instruction derived from our testcases. Table. 3 and Table. 4 shows the format of our Huffman-like coding for the beginning of each scanline and the Modified LZ77.

2.5 *The flowchart of compression and decompression*

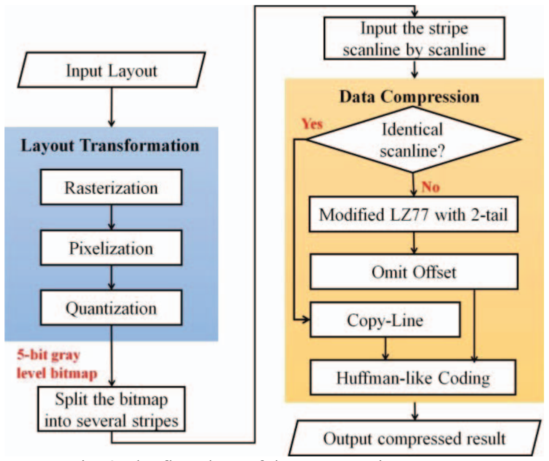


Fig. 8 The flowchart of the compression process.

Fig. 8 shows the flowchart of the compression process. We transform the layout into 5-bit gray level bitmap and split the bitmap into stripes with 1024-pixel width. Then, we compare the scanline with the previous scanline to check whether it can be copied. If not, we compress it according to the modified LZ77 and omit the offset. Finally, we apply the Huffman-like coding and output the compressed layout data.

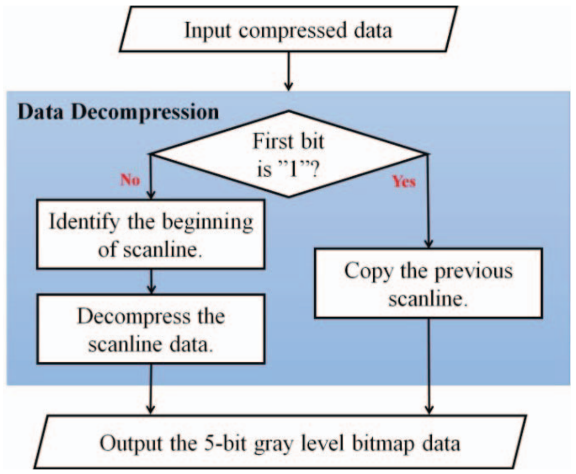

Input compressed data
Data Decompression
First bit is "1"?
No
Yes
Identify the beginning of scanline.
Decompress the scanline data.
Copy the previous scanline.
Output the 5-bit gray level bitmap data

Fig. 9 The flowchart of the compression process.

Fig. 9 shows the flowchart of the decompression process. We copy the previous scanline if the first bit of the input compressed data is "1". Otherwise, we identify the beginning of the scanline and decompress the data by following our proposed method.

3. Experimental Result

We targeted on two types of properties in our experiments. One is the compression rate, the other is the decompression time. We compare our works with the LineDiff Entropy by compressing the same 5-bit gray level bitmap data. All the experiments above are conducted on a workstation with the CPU Intel(R) Xeon(R) CPU E5-2643 v2 3.50 GHz system running Linux and equipped with 70 GB RAM.

The routing layers of six MCNC circuits are used as our benchmarks, which are shrunk to fit the 14-nm technology node following 2013 ITRS [8], and the pixel size is set to be 7×7 nm^2. We split the bitmaps into stripes with 1024-pixel width by following the experiment flow in LineDiff Entropy. For the evaluation and comparison purposes, we have implemented LineDiff Entropy and our method in C++ language. Note that by observing that an e-beam writer can scan in both directions according to the application of vector scan [5], we rotate 1-D layouts into the same direction to derive higher compression rates for both LineDiff Entropy and our algorithm.

3.1 The compression results

Table. 5 The comparison of compression factor.

Circuit Layout	Size(μm^2) (after shrink)	Compression Factor		
		LineDiff	Ours	Data reduction
S5378	541.25 x 296.25	202.81	265.03	30.7%
S9234	502.5 x 278.75	239.8	285.79	19.2%
S13207	823.75 x 455	240.43	256.96	5.6%
S15850	880 x 485	216.28	264.19	22.2%
S38417	1428.75 x 772.5	234.37	283.6	21%
S38584	1617.5 x 838.75	221.1	242.12	9.5%

Table. 5 shows the compression factor of LineDiff Entropy and our method. There are three layers for each circuit layout. And the shrank area size of them are listed in the table. The results show that the compressed data volume of the proposed method averagely improves the compression rate by 18% compared with Linediff Entropy.

3.2 The decompression results

Table. 6 The comparison of decompression time.

Circuit Layout	Time (s)		
	LineDiff(A)	Ours(B)	A/B
S5378	2.48	0.13	19.08
S9234	1.74	0.11	15.82
S13207	5.81	0.41	14.17
S15850	5.2	0.49	10.61
S38417	10.44	1.39	7.51
S38584	20.54	1.77	11.6

Table. 6 shows the comparison of decompression time between LineDiff Entropy and our method. The ratio results show that our decompression method is at least 7.5 times faster than that of LineDiff Entropy.

4. Summary

In this paper, our method has achieved both higher compression factor and decompression speed due to the comprehensive analysis of 5-bit gray-level bitmap. Our experimental results demonstrate that the algorithm is very effective in compressing e-beam data. In addition, our algorithm also achieves the higher decompression speed, which helps the MEBDW systems to meet higher WPH in advanced lithography.

5. Acknowledgment

The authors would like to thank S.J. Lin and his colleagues at Multiple E-beam Program, Nano Patterning Technology Division, Taiwan Semiconductor Manufacturing Company. This work was supported by the project "5-nm Node and beyond Resolution Limitation Study in Electron Beam Lithography".

6. References

[1] G. de Boer *et al.*, "MAPPER: progress towards a high volume manufacturing system," *Proc. SPIE* 8680, 86800O (2013).

[2] Thomas Gubiotti *et al.*, "Reflective electron beam lithography: lithography results using CMOS controlled digital pattern generator chip," *Proc. SPIE* 8680, 86800H (2013).

[3] Cheng-Chi Wu, Jensen Yang, Wen-Chuan Wang, Shy-Jay Lin, "An Instruction-based High-Throughput Lossless Decompression Algorithm for E-Beam Direct-Write System," *Proc. SPIE,* vol. 9423, Alternative Lithographic Technologies VII, 94231P, (2015).

[4] Chin-Khai Tang *et al*, "LineDiff Entropy: Lossless Layout Data Compression Scheme for Maskless Lithography Systems," *IEEE Signal Processing Letters*, Vol. 20, No. 7, (2013).

[5] Paul Petric *et al*. "New advances with REBL for maskless high-throughput EBDW lithography," *Proc. SPIE 7970,* 797018 (2011).

[6] Jacob Ziv *et al*. "A Universal Algorithm for Sequential Data Compression". *IEEE Trans Inf Theory*, Vol. IT-23, No. 3, (1977).

[7] Shao-Yun Fang, Iou-Jen Liu, and Yao-Wen Chang, "Stitch-Aware Routing for Multiple E-Beam Lithography," *DAC'13*, (2013).

[8] 2013 ITRS roadmap: http://www.itrs.net/