The Impact of Parallelism on Data Volumes for a Multibeam Mask Writer

N. Chaudhary, Y. Luo, S. A. Savari[1]
Department of Electrical & Computer Engineering, Texas A&M University
savari@ece.tamu.edu

Single beam mask writer architectures have satisfied mask patterning requirements for decades, but there is considerable interest in multibeam mask writers to handle the throughput and resolution challenges associated with the needs of sub-10 nm technology nodes[2,3]. Future mask writers must transmit terabits of information per second and handle petabytes of data[3]. For electron-beam direct-write (EBDW) lithography systems parallelism and lossless data compression are techniques which have been considered together to approach the data transfer problem[4,5]. Just as the throughput requirements for EBDW lithography systems necessitate the lossless data compression decoders to quickly reproduce the layout images from the compressed files it is likewise interesting to study simple compression algorithms for multibeam mask writers. We will examine how parallelism affects the total beam compressed data for one multibeam architecture.

We are inspired by the IMS Nanofabrication multibeam mask writer series eMET[6,7]. The writing strategy for eMET is proprietary, so we consider an idealization based on some known parameters. The eMET series proof-of-concept tools use an aperture plate system consisting of a square array of 512×512 programmable beams each of which is either 20 nm or 10 nm to write 5 nm pixels. Each shot is exposed with a dose in the range $\{0,\ 1,\ \ldots,\ 15\}$. As illustrated in Figure 1, the shots overlap, so the pixels for the 20 nm beam array are written at most 16 times and can take on $241(= 15 \times 16 + 1)$ gray levels while the pixels for the 10 nm beam array are written at most four times and can take on $61(= 15 \times 4 + 1)$ gray levels. We focus here on 10 nm beams and likewise assume overlapping shots and a scanning strategy that writes each pixel at most four times. The arrays we study have dimensions $2^N \times (2^N - 1)$ beams, where N is an odd integer. The distance between successive pixels written by a single beam grows as the array size increases, so the dependencies between successive dose values decreases. Therefore, for a fixed image one would expect the total size of the compressed data for a beam array to increase with the number of beams.

The experimental results in Table 2 are for a layout image based on the motif pattern of the Inverse Lithography Technology (ILT) features of contact holes with a minimum element of 80 nm. We used the electron beam proximity correction algorithm BEAMER of GenISys, Inc. We considered the widely used algorithm *deflate* as well as some simpler run-length codes[8] that have also been applied to image compression. In all cases compression helps with data volume, but the amount of data increases as the size of the beam array increases.

Acknowledgments: This work was made possible by the support of GenISys, Inc. The authors also thank N. Hayashi of Dai Nippon Printing for providing an ILT test layout image and T. Groves for helpful conversations.

1. This work was supported in part by NSF grant ECCS-1201994.

2. M. Chandramouli et al., *Proc. SPIE* **8522**, 85221K, (2012).
3. J. Choi et al., *Proc. SPIE* **9658**, 96580C, (2015).
4. H. J. Levinson, *Principles of Lithography* (SPIE, Bellingham, 2010), Third Edition, pp. 466-467.
5. A. Carroll et al., *Proc. SPIE* **9049**, 904917, (2014).
6. E. Platzgummer, *Proc. SPIE* **7637**, 763703, (2010).
7. E. Platzgummer, C. Klein, H. Loeschner, SPIE J. Micro/Nanol., MEMS, MOEMS, **12**, 031108, (2013).
8. S. Xue, Y. Xu and B. Oelmann, IEE Proc.-Vis. Image Signal Process., **150**(4), 256-260, (2003).

Figure 1. Overlap of Shots from Four Beams. Dose of Beam A: 15, Dose of Beam B: 14, Dose of Beam C: 13, Dose of Beam D: 12.

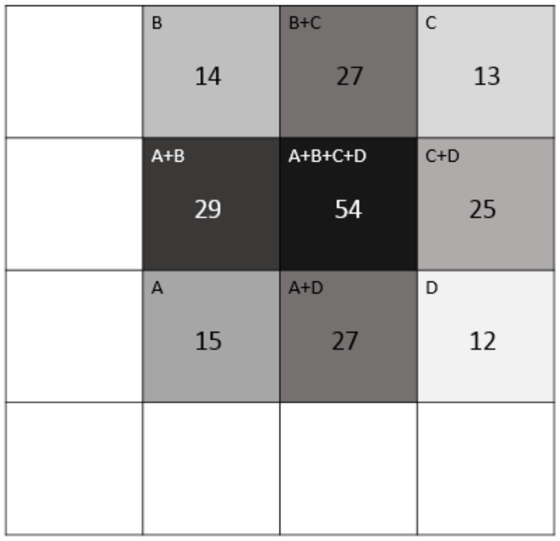

Table 1. Overview of Scanning Strategy for a Family of Beam Arrays.

Array	Number of beams	Distance in pixels between the centers of two neighboring beams in the same row or column	Horizontal movement of array from one writing to the next within a "stripe"	Vertical movement of array from one writing to the next within a "stripe"
$2^N \times (2^N-1)$, N odd	$2^N(2^N-1)$	$d=\sqrt{2^{N+1}}$	$\frac{d}{2}$ or $\frac{d}{2}-1$ pixels depending on the iteration	+1 or -1 pixels depending on position of array within zigzag
2 × 1	2	2	0 every 2nd iteration +1 pixel for all other iterations	Progresses in the sequence $n,\ n+1,\ n+2,\ n+1,\ n,\ n+1,\ n+2,\ldots$ for some n
8 × 7	56	4	+1 every 4th iteration +2 pixels for all other iterations	Progresses in the sequence $n,\ n+1,\ n+2,\ n+3,\ n+4,\ n+3,\ n+2,\ n+1,\ n,\ n+1,\ n+2,\ \ldots$ for some n
32 × 31	992	8	+3 every 8th iteration +4 pixels for all other iterations	Progresses in the sequence $n,\ n+1,\ \ldots,\ n+8,\ n+7,\ \ldots,\ n+1,\ n,\ n+1,\ n+2,\ \ldots$ for some n
128 × 127	16256	16	+7 every 16th iteration +8 pixels for all other iterations	Progresses in the sequence $n,\ n+1,\ \ldots,\ n+16,\ n+15,\ \ldots,\ n+1,\ n,\ n+1,\ n+2,\ \ldots$ for some n

Table 2. Data Volumes in MB for an ILT image with 30017*33300 pixels. EGk is the exponential-Golomb code of order k.

Array	Uncoded	*deflate*	Gallager	EG0	EG1	EG2	EG3	hybrid (ref. 8)
2 × 1	476.7	11.3	46.5	23.8	28.5	33.8	39.9	23.8
8 × 7	477.3	28.9	46.8	32.8	35.6	39.1	43.2	33.9
32 × 31	484.1	41.9	47.0	42.0	41.8	43.1	44.8	43.9
128 × 127	532.2	56.7	48.1	47.8	46.0	45.3	45.8	50.2