Full-chip characterization of compression algorithms for direct-write maskless lithography systems

Vito Dai
Avideh Zakhor
George Cramer
University of California
Berkeley, California 94720
E-mail: avz@eecs.berkeley.edu

Abstract. Future lithography systems must produce more dense microchips with smaller feature sizes while maintaining throughput comparable to today's optical lithography systems. This places stringent data-handling requirements of up to 12 Tb/s on the design of any maskless lithography system. In past work, we have developed data-path architectures for such throughput and shown that lossless compression algorithms play a key role in such systems. We currently investigate the effectiveness of the Block C4 lossless layout compression algorithm in increasing throughput of direct-write maskless lithography systems. In particular, we characterize the compression efficiency of Block C4 on 1024×1024 blocks of select layers of two 65-nm chips: a state-of-the-art microprocessor chip and a low-density parity code chip used commonly in wireless communication applications. Overall, we have found that compression efficiency varies significantly from design to design, from layer to layer, and even within parts of the same layer. We propose a number of ways to cope with the variation of compression ratios within a layer and characterize the way each method affects the overall wafer layer throughput. © *2010 Society of Photo-Optical Instrumentation Engineers.* [DOI: 10.1117/1.3366553]

Subject terms: compression; lithography; maskless; direct; datapath; throughput; C4; microprocessor.

Paper 09029PR received Mar. 4, 2009; revised manuscript received Jul. 10, 2009; accepted for publication Feb. 5, 2010; published online Mar. 29, 2010.

1 Introduction

Future lithography systems must produce chips with smaller feature sizes while maintaining throughput comparable to today's optical lithography systems. This places stringent data-handling requirements on the design of any direct-write maskless system. Optical projection systems use a mask to project the entire chip pattern in one flash. An entire wafer can then be written in a few hundreds of such flashes. To be competitive with today's optical lithography systems, direct-write maskless lithography needs to achieve throughput of one wafer layer per minute. In addition, to achieve the required 1-nm edge placement with 22-nm pixels in 45-nm technology, a 5-bit/pixel data representation is needed. Combining these together, the data rate requirement for a maskless lithography system is ~12 Tb/s. To achieve such a data rate, we have recently proposed a data-path architecture shown in Fig. 1.[1–5] In this architecture rasterized, flattened layouts of an integrated circuit are compressed and stored in a mass storage system. The compressed layouts are then transferred to the processor board with enough memory to store one layer at a time. This board will then transfer the compressed layout to the writer chip, composed of a large number of decoders and actual writing elements. The outputs of the decoders correspond to uncompressed layout data and are fed into digital-to-analog converters driving the writing elements, such as a micromirror array or E-beam writers.[6–12]

In the proposed data-delivery path, compression is needed to minimize the transfer rate between the processor board and the writer chip, and also to minimize the required disk space to store the layout. Because there are a large number of decoders operating in parallel on the writer chip, an important requirement for any compression algorithm is to have an extremely low decoder complexity.

To this end, we have proposed a lossless layout compression algorithm for flattened, rasterized data called context copy combinatorial coding (C4).[1] The C4 algorithm automatically segments layout data into repetitive and nonrepetitive regions. Repetitive regions are compressed with 2-D copying similar to 2-D-Lempel-Ziv (2DLZ),[4] whereas nonrepetitive regions are compressed with image context-based prediction similar to lossless JPEG compression.[13] We have also devised Block C4, a variant of C4, with up to

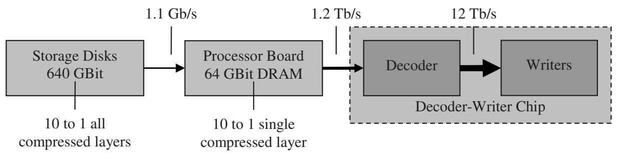

1.1 Gb/s
1.2 Tb/s
12 Tb/s
Storage Disks 640 GBit
Processor Board 64 GBit DRAM
Decoder
Writers
Decoder-Writer Chip
10 to 1 all compressed layers
10 to 1 single compressed layer

Fig. 1 System architecture of a data-delivery system for maskless lithography using compression to minimize data transfer rates.

1932-5150/2010/$25.00 © 2010 SPIE

Table 1 Specifications for an industry microprocessor designed for the 65-nm-device generation.

Manufacturing specifications		Maskless lithography specifications	
Minimum feature	65 nm	Pixel size	32 nm
Edge placement	1 nm	Pixel depth	6 bits (0–32) gray
Chip size	8.3×14.1 mm	Pixel data (one chip layer)	689 Gb
Wafer size	300 nm	Wafer data (one wafer layer)	415 Tb
Wafer throughput (one layer)	1 wafer per 60 s	Average data throughput	6.9 Tb/s over one wafer layer

a hundred times faster encoding times, with little or no loss in compression efficiency as compared to C4.[5] Block C4 speeds up the segmentation step in C4 by using a fixed block size.

C4 has been shown to outperform all existing techniques, in terms of compression efficiency, especially under limited decoder buffer size, as required for hardware implementation.[1] The two most competitive among the existing techniques have been shown to be ZIP and BZIP2. ZIP is an implementation of the LZ77 compression[14] method used in a variety of compression programs, such as pkzip, zip, gzip, and WinZip. The ZIP algorithm treats the input as a generic stream of bytes and takes advantage of repetitions in the byte stream to compress the data. BZIP2 is an implementation of the Burrows–Wheeler transform (BWT).[15] The BZIP2 algorithm also treats the input as a generic stream of bytes but uses a technique called block sorting to permute a sequence of bytes to make it easier to compress.

In this paper, we show compression efficiency results of Block C4 and competing techniques, such as ZIP and BZIP2, for the Poly, Active, Contact, Metal1, Via1, and Metal2 layers of a complete industry 65-nm layout. Although our past work has focused on characterizing the compression efficiency of C4 and Block C4 on samples of a variety of industrial layouts, there has been no full chip performance characterization of these algorithms. Specifically, we reexamine the comparison between Block C4, ZIP, and BZIP2; only this time, statistics are presented for a full production industry microprocessing chip, rather than individual images sampled here and there across a chip.

The layout used for these calculations are for an industry-standard microprocessor designed for the 65-nm device generation. The specifications for manufacturing this design is shown in Table 1. The layout is 8.3×14.1 mm in size with polygons laid out on a 1 nm grid, and it has 9325 cells. The Graphic Data System (gds) file size for Poly, Metal1, Metal2, Contact, Active, and Via are 460 MB, 482 MB, 1.3 GB, 476 MB, 449 MB, and 1.1 GB, respectively. The appropriate pixel size for this generation is 32×32 nm, with 33 (0–32) levels of gray to achieve 1-nm edge placement control, which requires 6 bits/pixel. The computed rasterized pixel image data is 0.7 Tb per chip layer, 415 Tb per wafer layer. The manufacturing throughput requirement for lithography is 1 wafer layer per 60 s. Therefore, the required average maskless lithography data rate over one wafer layer is 6.9 Tb/s.

In previous work, individual layout clips are characterized as dense, sparse, repetitive, and nonrepetitive, with each term intuitively defined by visual inspection. This manual *ad hoc* characterization does not scale to a full chip run. Instead, we define here a metric for polygon complexity that intuitively matches to the concept of "dense," namely, the number of polygon vertices within a given area, or *vertex density*. If the number of vertices is large in a fixed area, then it must be caused by either the presence of many distinct polygons or polygons with very complex fragmented edges. In either case, for the simple three-pixel prediction mechanism used by C4 and Block C4, the number of vertices is directly correlated with the number of context-based prediction errors.

In terms of repetitions, it is difficult to find a single metric that decisively determines this for a 2-D image and that is reasonable to compute for such a large data set. One method would have been to use the same search-based segmentation used by LZ77/C4 itself, but this defeats the purpose of having an independent metric. Other techniques evaluated, such as image correlation and window-based discrete cosine transform, do not correlate well with the copy mechanism of C4 and LZ77, for which the cost of even correcting a small 1% intensity error is fairly high. Such techniques are more appropriate for lossy compression techniques where such errors may be ignored. In the end, we choose to use a metric taken from the layout hierarchy itself. The measure of repetition is defined as the number of cells in a given region minus the number of unique cells in that same region. As an example, suppose a region contains five instances of cell A, four instances of cell B, and one instance of C, D, and E. Then, the total number of cells in that region is $5+4+1+1+1=12$, whereas the total number of unique cells is 5 (A–E); thus, the "repetition" of this region is 7.

In order to provide a visualization of the these metrics, Fig. 2 shows a gray-scale picture of the vertex density metric as applied to the Metal1 layer. It is easy to discern from this image regions of very high vertex densities that are

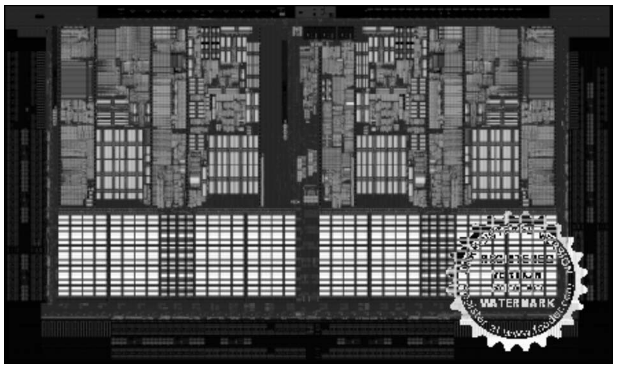


Fig. 2 A vertex density plot of Metal1 layer for a 65-nm microprocessor. Each pixel represents a 32×32 μm block. Higher vertex density blocks are assigned brighter pixels, and lower corner vertex density blocks are assigned darker pixels. Vertex densities range from 0 to 20,000 per block.

arranged in rectangular arrays throughout the design. These are the various memory arrays of the microprocessor. The darker grays are likely to be logic circuit areas, also arranged in rectangular arrays. Finally, the periphery regions are very dark, indicating low corner densities.

A plot of "repetitions" visually looks the same as Fig. 2. Although there are small differences in the data, which are detectable through data analysis, it is impossible to visually discern these differences. The visual similarity between a plot of "repetitions" and a plot of vertex density justifies the fundamental rationale behind C4. Highly dense layout regions are also highly repetitive and therefore compress well with copying techniques. Nonrepetitive regions tend to be sparse and hence compress well with context-based prediction techniques, because polygon corners generally correspond to prediction errors for Manhattan geometries.

For each of the 32×32 μm blocks, rasterization is performed using the methodology described in Ref. 16, where the pixel size is 32 nm and 33 gray levels are allowed (0–32), resulting in a fine 1-nm edge placement grid. One full chip layer contains 116,328 such blocks, equal to the number of pixels in Fig. 2. Each rasterized block is then passed through three compression algorithms (ZIP, BZIP2, and Block C4) and compression and decompression statistics are gathered for each. This process is then repeated for all the critical layers of the design: diffusion, also known as Active, Poly, Contact, Metal1, Via1, and Metal2.

For this experiment, decoder buffer size of ZIP, BZIP2, and Block C4 are 4, 900, and 1.7 kB, respectively, chosen based on the trade-off analysis presented in Ref. 5. The small buffer size used by Block C4 makes it particularly attractive for implementation in hardware for the data-path architecture presented earlier.

2 Full Chip Compression Statistics

Table 2 contains a summary of these full chip runs. Column 1 is the name of the full-chip statistic being reported. Column 2 is the chip layer, which is rasterized and compressed. Columns 3, 4, and 5 are the statistics for ZIP, BZIP2, and Block C4, respectively. Each row in Table 2 represents a layer statistic. The relevant statistics reported are the average compression ratio for the entire layer, the minimum compression ratio over individual 32×32 μm blocks, the total encoding run time for each layer, the total decoding run time for each layer, and the percentage of blocks with a compression ratio of <10.

Table 2 Full-chip compression summary table.

Statistic	Layer	ZIP	BZIP2	Block C4
Avg. compression ratio	Poly	12.6	15.3	14.1
	Metal1	4.2	4.5	5.2
	Metal2	6.1	7.2	7.2
	Contact	14.1	16.0	23.2
	Active	20.2	31.7	39.2
	Via1	12.3	14.1	14.0
Min. compression ratio	Poly	2.6	3.1	4.4
	Metal1	0.96	1.3	1.7
	Metal2	1.0	1.3	2.1
	Contact	2.7	4.3	4.8
	Active	8.1	11.1	12.8
	Via1	2.2	3.6	4.5
Total encoding time	Poly	42 min	2.3 h	420 h
	Metal1	45 min	2.3 h	420 h
	Metal2	45 min	1.9 h	408 h
	Contact	46 min	2.1 h	419 h
	Active	43 min	1.9 h	418 h
	Via1	46 min	2.1 h	419 h
Total decoding time	Poly	17 min	1.2 h	36 min
	Metal1	14 min	1.2 h	35 min
	Metal2	19 min	1.4 h	38 min
	Contact	15 min	1.4 h	38 min
	Active	15 min	1.3 h	37 min
	Via	15 min	1.4 h	38 min
Percentage of blocks with comparession ratio of <10 (lower is better) (%)	Poly	25.33	22.84	23.66
	Metal1	65.73	59.69	55.12
	Metal2	44.20	44.88	41.95
	Contact	0.73	0.07	0.00
	Active	7.85	0.00	0.00
	Via	4.94	0.22	0.14

Examining the average compression ratio for all layers, the compression efficiency of ZIP is generally lower than that of BZIP2 and Block C4. BZIP2 and Block C4 are generally comparable to each other. Considering that BlockC4 uses two or three orders of magnitude less decoder buffer to achieve more or less the same compression efficiency as BZIP2, clearly it is the algorithm of choice for hardware implementation. From layer to layer, Metal1 is most challenging to compress, followed by Metal2, Via1, Poly, Contact, and then Active. One different characteristic of Poly layout in this particular design style is that all gates are oriented in a single direction and spaced apart by a characteristic common pitch. Regularized design styles such as these can take better advantage of the copy mechanism in C4 to achieve high-compression efficiency. Of particular concern is the average compression ratio of the Metal1 and Metal2 layers, which are 5.2 and 7.2, respectively, which fall below the target compression ratio of 10.

Another important metric to consider is the minimum compression ratio over all 32×32 μm blocks for a layer. This is the most difficult block of any given layer to compress. In this case, only the Active layer meets a target compression ratio of 10. The remaining five layers fall below that target, and in the worst-case block of Metal1, the compression ratio is 1.7.

3 Managing Local Variations in Compression Ratios

So what are the implications of missing the compression target and which is more relevant, the average compression ratio or the more pessimistic minimum compression ratio? The answer depends on how well the maskless lithography system as a whole can absorb *local variations* in data throughput. This can be accomplished by physically varying the throughput of the maskless lithography writers or by introducing various mechanisms in the data path to absorb these variations, which we will speculate on later. By *local variations*, we are referring to interblock variations of compression ratios. In choosing our block size for analysis, we already assume there is at least a single block buffer in the system so that we may ignore intrablock variations in compression ratio. This buffer is distinct from the memory used by the decompression hardware. An example of such a buffer is the “SRAM Writer Interface” found in Ref. 17.

3.1 *Adjusting Board to Chip Communication Throughput*

In the worst case, (*i*) the maskless lithography writers are fixed at a constant writing speed over all blocks of a layer and (*ii*) the datapath cannot help absorb these interblock variations of compression ratios. In this case, the writing speed is limited by the data throughput of the minimum compression ratio block. From the the maskless datapath presented earlier, the formula to compute actual wafer throughput is $r_{\mathrm{wafer}} = r_{\mathrm{comm,max}} \times C_{\min} / d_{\mathrm{wafer}}$ where r_{wafer} is the wafer layer throughput, $r_{\mathrm{comm,max}}$ is the maximum board to chip communication throughput, $C_{\min}$ is the minimum compression ratio for Block C4, and $d_{\mathrm{wafer}} = 415$ Tb is the total data for one wafer layer, from Table 1.

Because d_{wafer} is fixed and $C_{\min}$ has been empirically determined for each layer, the total wafer throughput depends entirely on $r_{\mathrm{comm,max}}$, which is the maximum data throughput of board-to-chip communication. The reason maximum is emphasized is that this throughput is only required for the minimum compression ratio block. For blocks of higher compression ratio, the communication throughput can be reduced. As an example, if maximum communication throughput $r_{\mathrm{comm,max}} = 1$ Tb/s, then the wafer layer throughput for Metall is $1 \text{ Tb/s} \times 1.7 / 415 \text{ Tb} \times 3600 \text{ s/h} = 14.7$ wafer layers per hour. This same formula can be applied to each layer for various assumed values of $r_{\mathrm{comm,max}}$. The results of this exercise are shown in the third and forth columns of Table 3.

The first four columns of Table 3 are layer, minimum compression ratio, maximum board-to-chip communication throughput, and wafer-layer throughput, respectively. In the first six rows, we assume a maximum communication throughput of 1 Tb/s and compute the wafer throughput for various layers. In the next six rows, we target a wafer throughput of 60 wafer layers per hour and compute the maximum communication throughput needed to support this writing rate for each layer. As a point of reference, a state-of-the-art HyperTransport 3.0 (HT3) link offers 0.32 Tb/s maximum throughput.[18] Examining the third column of Table 3 for Metall with a target wafer throughput of 60 wafers per hour, a maskless data path requires at least $\lceil 4.07/0.32 \rceil = 13$ such links to achieve the required communications throughput. Implementing 13 links is costly, in terms of both circuit power dissipation and chip area.[5,19] However, a chip designer may be able to conserve power by taking advantage of the fact that the maximum communication throughput is only needed for a few blocks. The average communication throughput, as we shall see, is significantly lower.

The equation relating wafer throughput r_{wafer} to average board-to-chip communication throughput $r_{\mathrm{comm,avg}}$ and average compression ratio C_{avg} is straightforward: $r_{\mathrm{wafer}} = r_{\mathrm{comm,avg}} \times C_{\mathrm{avg}} / d_{\mathrm{wafer}}$. To be precise, the average is computed over all blocks of a wafer layer. Using this formula, we can relate wafer throughput to average communication throughput for various layers. The results are presented in the last three columns of Table 3. The columns are average compression ratio, average board-to-chip communication throughput, and wafer-layer throughput, respectively. The first six rows assume an average communication throughput of 1 Tb/s, and the next six rows target a wafer throughput of 60 wafer layers per hour.

Because the average compression ratio is significantly higher than the minimum compression ratio for all layers, the average communication throughput is also significantly lower than the maximum communication throughput computed previously. Continuing our previous example using an HT3 link as reference, for Metall with a target wafer throughput of 60 wafers per hour, a maskless data path requires only $1.33/0.32 = 4.2$ links on average. Thus, even though 13 links are required to accommodate the maximum throughput, on average only $4.2/13 = 32\%$ of the capacity is being used. The maskless data path can take advantage of this by powering down unused communication links to conserve power. However, that still leaves an area cost of implementing 13 links in the first place. What can be done to effectively smooth the data throughput so that communication links can be utilized more effectively?

Table 3 Maximum communication throughput versus wafer layer throughput for various layers in the worst-case scenario, when data throughput is limited by the minimum compression ratio for block C4.

Layer	$C_{\min}$	$r_{\mathrm{comm,max}}$ (Tb/s)	r_{wafer} (wafer layer/h)	C_{avg}	$r_{\mathrm{comm,avg}}$ (Tb/s)	r_{wafer} (wafer layer/h)
Poly	4.4	1	38.2	14.1	1	122
Metal1	1.7	1	14.7	5.2	1	45.1
Metal2	2.1	1	18.2	7.2	1	62.5
Contact	4.8	1	41.6	23.2	1	201
Active	12.8	1	111	39.2	1	340
Via1	4.5	1	39.0	14.0	1	121
Poly	4.4	1.57	60	14.1	0.49	60
Metal1	1.7	4.07	60	5.2	1.33	60
Metal2	2.1	3.29	60	7.2	0.96	60
Contact	4.8	1.44	60	23.2	0.30	60
Active	12.8	0.54	60	39.2	0.18	60
Via1	4.5	0.54	60	14.0	0.49	60

3.2 *Statistical Multiplexing Using Parallel Decoders*

An important feature to take advantage of is the opportunity to utilize averaging inherent in the parallel design of the maskless lithography data path. As described in Ref. 20, the decoder in Fig. 1 is implemented as a parallel array of decoder paths (i.e., multiple blocks are being decoded simultaneously). In its simplest form, the communication throughput is evenly divided among the parallel decoder paths. However, additional logic, such as packet scheduling, can be implemented to allocate communication throughput to each decoder path based on need. As such, a decoder path working on a block with low compression ratio is allocated more communication packets than a decoder path working on a block with high compression ratio. The result is that interblock variations in compression ratio are effectively statistically multiplexed by the number of decoder paths in the system.

Suppose we have N decoder paths working in parallel on N adjacent blocks in a row. In communication order, we form M frames of N blocks per frame, where $MN \geqslant 116{,}326$. Statistical multiplexing effectively allows us to average the compression ratio over each frame. We can then compute the minimum over all frames and denote this value as $C_{\min,N}$. Note that by definition $C_{\min,1}=C_{\min}$ and $C_{\min,116{,}328}=C_{\mathrm{avg}}$. $C_{\min,N}$, r_{wafer}, and $r_{\mathrm{comm,max}}$ are related through this equation: $r_{\mathrm{wafer}}=r_{\mathrm{comm,max}}\times C_{\min,N}/d_{\mathrm{wafer}}$.

Using different values for N, we compute the $C_{\min,N}$ and $r_{\mathrm{comm,max}}$ for Block C4, Metall, and a target throughput of 60 wafer layers per hour. These results are summarized in Table 4. In columns are the number of decoder paths N, the minimum frame compression ratio $C_{\min,N}$, the maximum board-to-chip communications throughput $r_{\mathrm{comm,max}}$, the wafer throughput r_{wafer}, and the number of HT3 links needed to support the communications throughput. Clearly, $C_{\min,N}$ increases as the number of decoder paths N increases. At $N=1000$, $C_{\min,N}=4.9$, which is very close to $C_{\mathrm{avg}}=5.2$, demonstrating the strength of the statistical multiplexing approach. The corresponding maximum communication throughput is 1.41 Tb/s, which can be met with $\lceil 1.41/0.32 \rceil=5$ HT3 links.

3.3 *Adding Buffering to the Datapath*

Another way to smooth the data throughput is to introduce on-chip memory buffer at the output of the communications channel before decompressing the data in Fig. 1. This buffer absorbs variations in data throughput caused by in-

Table 4 Effect of statistical multiplexing using N parallel decoder paths on block C4 compression ratio and communication throughput for metal1.

N	$C_{\min,N}$	$r_{\mathrm{comm,max}}$ (Tb/s)	r_{wafer} (wafer layer/h)	No. of HT3 links
1	1.7	4.07	60	13
2	2.3	3.01	60	10
10	2.5	2.77	60	9
100	3.3	2.10	60	7
1000	4.9	1.41	60	5
116,328	5.2	1.33	60	5

terblock variations of compression ratios. For blocks with high compression ratios, excess communication throughput is used to fill the buffer. For blocks with low compression ratio, data are drained from the buffer to supplement the communication channel. Intuitively, the larger the buffer is, the more variations it can absorb, and the lower is the required maximum communication throughput. On the other hand, the primary advantage of spending area on a buffer in the first place is to save on chip area devoted to communication. Therefore, there is a trade-off between the area needed by the buffer and the additional area saved by reducing the number of communication links.

We can roughly estimate the amount of buffer to add using the following steps. Suppose we add sufficient buffer equivalent to the minimum compressed block. For Metall, this buffer is $(1000\times 1000\times 6 \text{ bits})/1.7=3.5$ Mb in size for Block C4. Now suppose, in communication order, we group blocks pairwise and compute each pair's compression ratio, followed by computing the minimum over all pairs $C_{\text{min,pair}}$. This number is guaranteed to be higher than C_{min} and lower than C_{avg}. Empirically for Metall, $C_{\text{min,pair}}=2.3$ for Block C4, assuming raster scan order. For this system, the following inequality holds: $r_{\text{wafer}} \geqslant r_{\text{comm,max}} \times C_{\text{min,pair}}/d_{\text{wafer}}$. That is, at the very least, we should be able to replace C_{min} with the higher $C_{\text{min,pair}}$ for relating wafer throughput to the maximum communication throughput. Continuing our previous example for Metall with a target wafer throughput of 60 wafers per hour, the result is $r_{\text{comm,max}} \leqslant 3.01$ Tb/s, equivalent to $\lceil 3.01/0.32 \rceil=10$ HT3 links. Compared to the 13 HT3 links for zero buffering, this is a reduction of three links for 3.5 Mb of buffering, which seems to be worthwhile trade-off. Clearly, more systematic analysis of such trade-offs are necessary for any future practical maskless lithography systems.

3.4 *Distribution of Low-Compression Blocks*

The computation of $r_{\text{comm,max}}$ in the previous paragraph is a conservative upper bound, in that it focuses on the worst case, where low compression-ratio blocks may be clustered together. Thus, we require that any drain on the buffer caused by a low compression ratio block to be immediately refilled by the adjacent block. If low compression blocks are spread far apart from each other by coincidence, then $r_{\text{comm,max}}$ may be significantly lowered. Furthermore, if the writing system allows for limited reordering of the blocks, then this could be used to intentionally spread the low compression ratio blocks apart. As an example, some maskless lithography systems are written in a step-and-scan mode, where multiple blocks form a frame that is written in a single scan.[20] In this case, blocks may be reordered within a frame to smooth the data rate.

Figure 3 is a visualization of the compression ratio distribution of Block C4 for the Metall layer. Brighter pixels are blocks with low compression ratios, and darker pixels are blocks with high compression ratios. Note that repetitive memory arrays on the bottom half are relatively dim. Block C4 compresses these repetitive regions effectively. The less regular but relatively dense layout are clustered in distinct bright regions in the middle. This geographic distribution should be taken into consideration when deciding on the mechanism to smooth interblock variations.

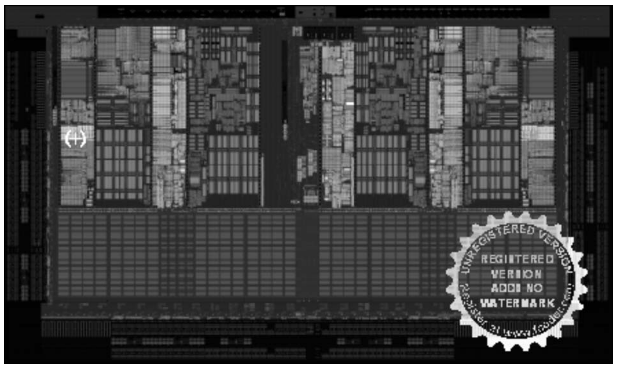


Fig. 3 A visualization of the compression ratio distribution of block C4 for the Metal1 layer. Brighter pixels are blocks with low compression ratios, while darker pixels are blocks with high compression ratios. The minimum 1.7 compression ratio block is marked by a white crosshair (+).

3.5 *Modulating the Writing Speed*

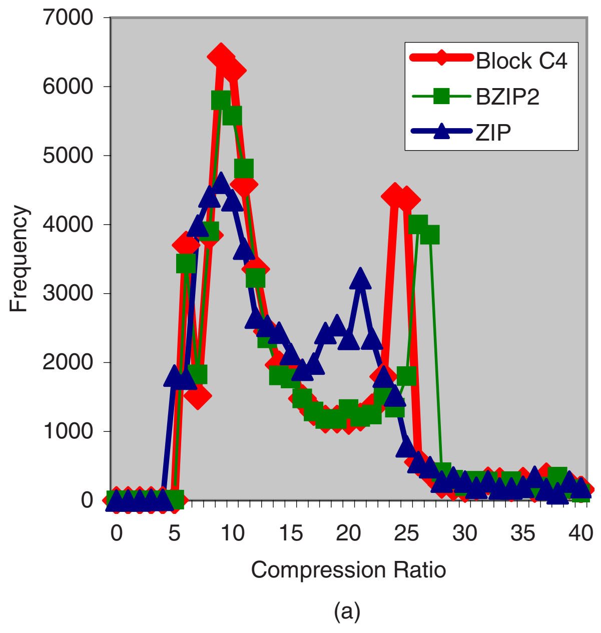
Another possibility is to modulate the writing speed of the maskless lithography writers to match the interblock variations in compression ratio. For example, it is conceivable to divide blocks into discrete classes based on the range of compression ratios into which they fall. The lithography writers would then switch between a discrete number of writing speeds depending on the class of block. The "high" compression ratio blocks are written with "high" speed, whereas "low" compression ratio blocks are written with "low" speed. Because of overhead in switching speeds, it may not be feasible to vary the writing speed on a block-by-block basis. In this case, the writers would change speed based on the minimum compression ratio within a contiguous group of blocks.

Whichever mechanism is used to smooth the data throughput, the effectiveness depends on the distribution of compression ratios across all blocks of a layer. Intuitively, the higher the number of low compression ratio blocks there are the more difficult it is to lower the maximum communication throughput. Let us examine the distribution of these variations.

4 Distribution of Compression Ratios

Figure 4(a) shows the histogram of compression ratios for the full-chip Poly layer for Block C4, ZIP, and BZIP2. The horizontal axis is the compression ratio bins ranging from 0 to 40 in increments of 1. The vertical axis is the count of the number of blocks that fall into each bin. The histogram of Block C4 is plotted in red with diamond markers, BZIP2 in green with square markers, and ZIP in blue with triangular markers. The first observation to be made about this histogram is that the distribution of compression ratios is multimodal and non-Gaussian. Second, note that the distribution has an extremely long tail beyond 30. In general, the layout contains a large amount of blank regions filled by a few large polygons. The information content in these regions are low and compress easily.

An alternative view of the same data is presented in Fig. 4(b). In this case, we plot the cumulative distribution of blocks on the vertical axis against the compression ratio on the horizontal axis. Figure 4(b) is essentially the normal-



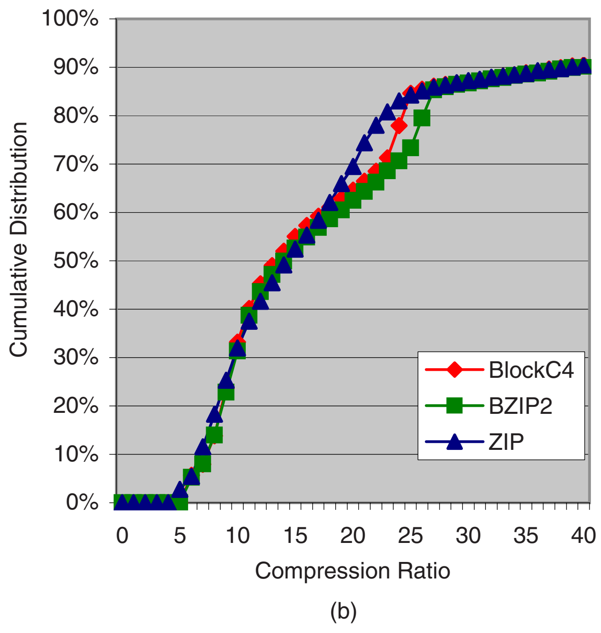


Fig. 4 Compression ratios for block C4, BZIP2, and ZIP for the poly layer: (a) Histogram and (b) cumulative distribution function.

ized integral of the plot in Fig. 4(a). The cumulative distribution function (CDF) of the compression ratio of Block C4 is plotted in red with diamond markers, BZIP2 in green with square markers, and ZIP in blue with triangular markers. A point on the CDF curve represents the percentage of blocks Y with compression ratio less than X. Generally speaking, when the curve shifts to the right, the overall compression efficiency of a layer is improved.

Of particular interest are compression ratio bins at the low end of the spectrum, because these are our throughput bottlenecks. In Fig. 4(b), 25.3% of ZIP blocks, 22.8% of BZIP2 blocks, and 23.7% of Block C4 blocks have a compression ratio of <10. Therefore, in the low end of the compression spectrum, Block C4 and BZIP2 have about the same compression efficiency and both have better efficiency than ZIP. In addition, even though the reported minimum compression ratio in Table 2 for Block C4 and BZIP2 are 4.4 and 3.1, respectively, the CDF curve clearly shows that very few blocks have compression ratios of <5. In fact, for this Poly layer, only seven of the 116,328 blocks have compression ratio's of <5 for Block C4 and BZIP2. These seven blocks are clustered in two separate regions, and within a region, no two blocks are adjacent to each other. The total size for these seven blocks compressed by Block C4 is 9.1 Mb. Therefore, if we have enough memory buffer to simply store all seven compressed blocks, then we can effectively use five as the minimum compression ratio for Poly. On the other hand, $2.8\% \approx 1800$ of ZIP blocks have compression ratio of <5. Because there are more variations, the system has to work harder to absorb them.

An alternative to absorbing the variation is to reexamine the compression algorithm to look for ways to compress these difficult blocks more efficiently. Figures 5 and 6 are samples of such hard-to-compress blocks for Poly and Metal1 layout. The key observation to make is that these blocks are dense in polygon count and yet are not regularly repeated structures, although some repetition does exist. Metal1 is more dense and less repetitive and therefore has significantly lower compression ratio than Poly. Increasing the buffer size of Block C4 from 1.7 to 656 kB does improve the compression efficiency, but not by a commensurate amount. For the Poly block in Fig. 5, the Block C4 compression ratio improves from 4.4 to 5.1, and for the Metal1 block in Fig. 6, the Block C4 compression ratio improves from 1.7 to 1.9.

Another way to gauge the difficulty of compressing the blocks in Figs. 5 and 6 is to compute the entropy. Entropy is the theoretical minimum average number of bits needed

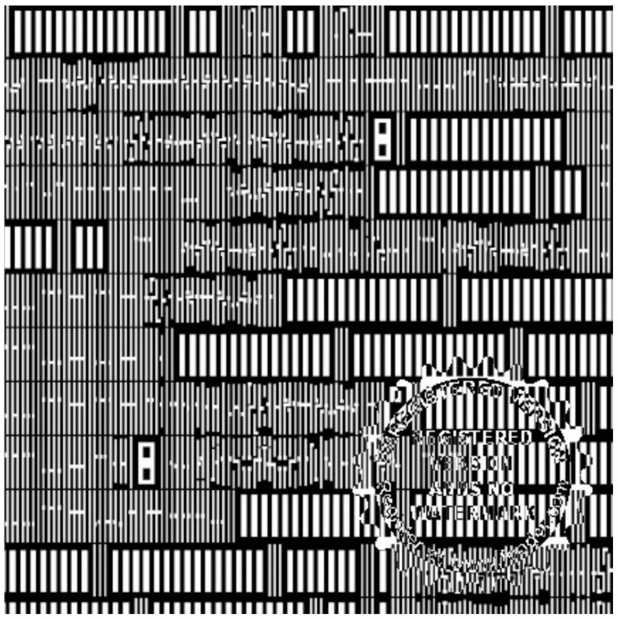


Fig. 5 A block of the poly layer, which has a compression ratio of 2.3, 4.0, and 4.4 for ZIP, BZIP2, and block C4, respectively.

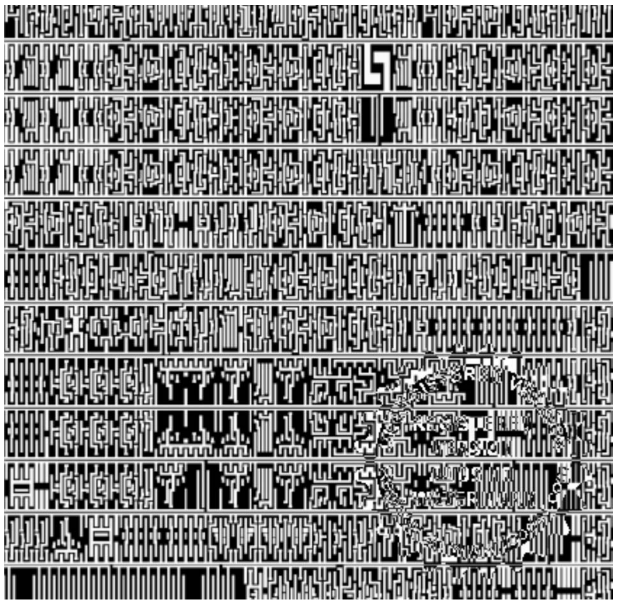


Fig. 6 A block of the M1 layer which has a compression ratio of 1.1, 1.4, and 1.7 for ZIP, BZIP2, and block C4, respectively.

to losslessly represent each pixel, assuming pixels are independently and identically distributed. This assumption does *not* hold for layout pixel data. Nonetheless, entropy still serves as a useful point of reference. For Fig. 5, the entropy is 3.7 bits/pixel, which corresponds to a compression ratio of 6/3.7 bits/pixel=1.6. For Fig. 6, the entropy is 4.8 bits/pixel, which corresponds to a compression ratio of 6/4.8 bits/pixel=1.3. Huffman coding realizes a compression ratio very close to entropy: 1.6 and 1.2 for Figs. 5 and 6, respectively.

Another alternative is to systematically change the layout so as to improve its compression efficiency. It is usually possible to preserve the same design intent using a different physical layout. If the design can be made more "compression friendly" in these difficult blocks, then the compression efficiency can be improved.

For completeness of analysis, Fig. 7 shows CDF plots of Contact, Active, and Via1, and Fig. 8 shows the same for Metal 1 and Metal2 layers. Examining these plots, Block C4 clearly has higher compression efficiency for Contact, Active, and Metal1 layers than both BZIP2 and ZIP. For the Via1 and Metal2 layers, the compression efficiency of Block C4 is comparable to BZIP2, particularly in the region of compression ratios of <10. Both Block C4 and BZIP2 have higher efficiency than ZIP.

Comparing the curves between levels, clearly Metal1 is the most difficult to compress. For a given low compression ratio threshold, for example, 5, Metal1 has the largest percentage of blocks falling below that threshold (i.e., 24% for Block C4). Metal2 follows with 0.81% for Block C4. The remaining layers contain no blocks below that threshold. Table 5 lists the complete numbers for all layers and compression algorithms using a low compression ratio threshold of 5. The reason Metal1 and Metal2 are particularly challenging is simple. These layers are the primary wiring layers connecting device to device, and as anyone who has untangled cables behind a personal computer can attest, wires quickly turn into a complex mess if not carefully managed. Intuitively, this means that the wiring layers tend to be more dense and less regular than the other chip design layers, making them the most difficult to compress. The density of polygon corners makes it difficult for context prediction to achieve good compression, and the irregularity of the design makes it difficult for copying to achieve good compression. The Block C4 segmentation algorithm is stuck between the proverbial rock and a hard place. Nonetheless, to the extent that some compression has been achieved, the algorithm does benefit from having both prediction and copying. As an example, turning off copying reduces the Block C4 compression ratio to 1.4 from 1.7 for the Metal1 block shown in Fig. 6.

Another question we can ask is, if we can exclude the 100 most difficult to compress blocks out of 116,328 blocks, either via buffering or some other mechanism, what is the minimum compression ratio for each layer? The result is shown in Table 6. For Metal1, Metal2, and Active, there is little change. However, for Poly, Contact, and Via, there is a significant improvement. For these layers, the minimum compression ratio is pessimistic due to a small number of special cases. If these small number of variations can be absorbed by the maskless lithography system or by

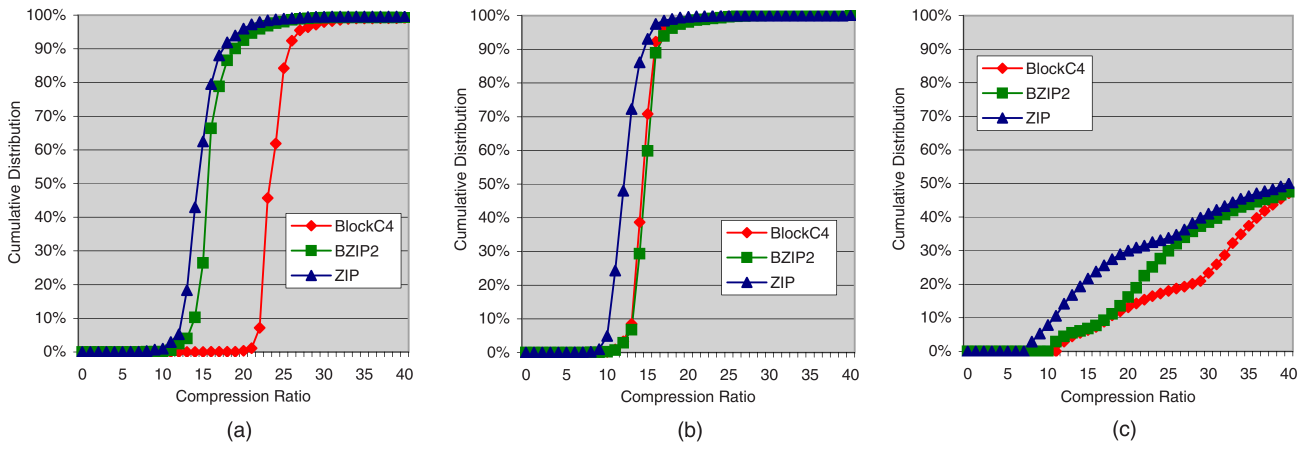


Fig. 7 CDF of compression ratios for block C4, BZIP2, and ZIP for (a) Contact, (b) Via 1, and (c) Active.

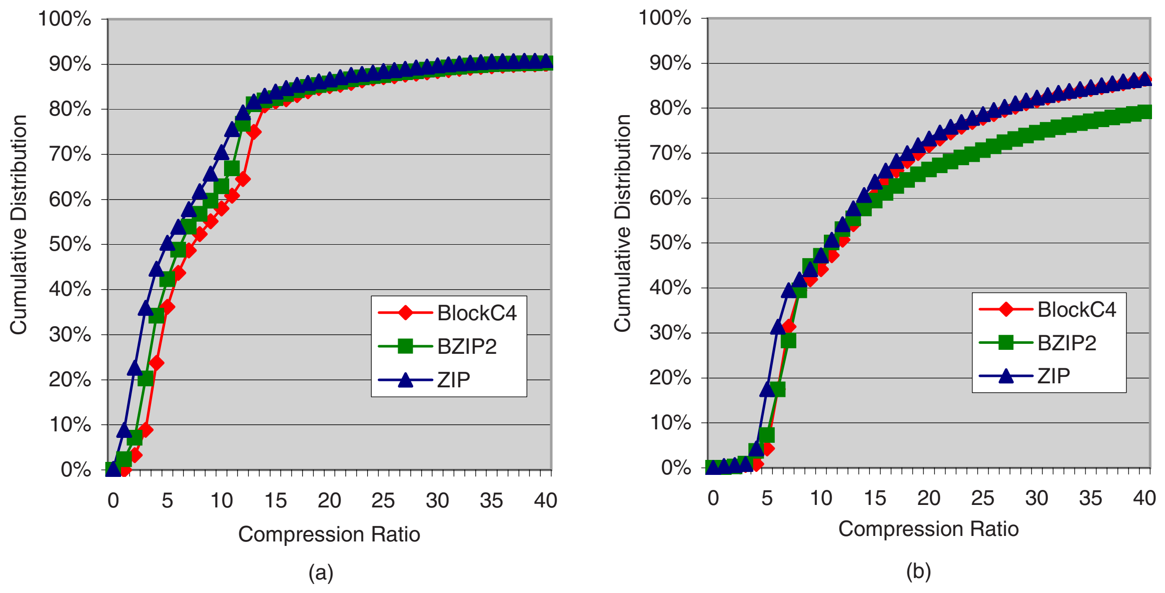


Fig. 8 CDF of compression ratios for block C4, BZIP2, and ZIP for (a) Metal1 and (b) Metal 2.

systematically altering the design to be more compression friendly, then the overall wafer throughput can be improved significantly.

5 Comparison of Encoding and Decoding Times

Examining the encoding times in Table 2, clearly ZIP is the fastest, BZIP2 is about three times slower than ZIP, and Block C4 about 200 times slower than BZIP2. All run times are reported for an Athlon64 3200+ Windows XP desktop with 1 GB of memory. BlockC4 is implemented in C# language. ZIP and BZIP2 are commercially available software, written and optimized in C code.

Part of the reason that Block C4 is so much slower is the inherent complexity of the Copy/Context prediction segmentation code, and another part is the lack of code optimization. All three algorithms have fairly stable and predictable run times that are independent of the layer. This is a significant advantage over the layer-dependent and extremely long runtimes of C4 seen previously.[5]

It is also worth noting that each of the 116,328 blocks are independently compressed and decompressed, which supports the highly parallel decoder implementation. Similarly, the total compression run time can be reduced by parallelizing the run using multiple CPUs and dividing the blocks evenly between CPUs.

Examining decoding times, ZIP is again the fastest, but here Block C4 is faster than BZIP2 by a factor of 2. Considering Block C4's decode buffer requirement is two orders of magnitude less than BZIP2, it is clearly the best choice for hardware implementation. Block C4 is a highly asymmetric algorithm in terms of encoder versus decoder

Table 5 Percentage of blocks with compression ratio of <5.

Statistic	Layer	ZIP	BZIP2	Block C4
Percentage of blocks with compression ratio of <5 (lower is better) (%)	Poly	0.03	0.00	0.00
	Metal1	44.63	34.20	23.72
	Metal2	4.33	3.75	0.81
	Contact	0.02	0.00	0.00
	Active	0.00	0.00	0.00
	Via	0.01	0.00	0.00

Table 6 Minimum compression ratio excluding the lowest 100 compression ratio blocks.

Statistic	Layer	ZIP	BZIP2	Block C4
Min. compression ratio over all blocks	Poly	2.6	3.1	4.4
	Metal1	0.96	1.3	1.7
	Metal2	1.0	1.3	2.1
	Contact	2.7	4.3	4.8
	Active	8.1	11.1	12.8
	Via1	2.2	3.6	4.5
Min. compression ratio excluding the lowest 100 compression ratio blocks	Poly	4.1	5.2	5.2
	Metal1	1.0	1.4	1.8
	Metal2	1.4	2.5	2.5
	Contact	8.1	10.0	19.8
	Active	8.1	11.1	12.9
	Via	8.2	10.5	11.0

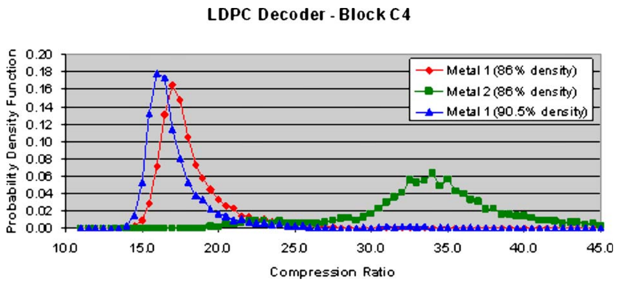


Fig. 9 Histogram of compression ratios for block C4 for Metal1 (86%), Metal2 (86%), and Metal1 (90%) layers of the LDPC chip.

complexity because segmentation is not required by the decoder, and consequently, its decoding speed is about 800 times faster than its encoding speed.

6 Distribution of Compression Ratios for an ASIC

Thus far, our full chip characterization of Block C4 has been focused on an industry microprocessor. Because maskless lithography is likely to impact low-volume application-specific integrated circuit (ASIC) manufacturing before it is used for high-volume general purpose chips, such as a microprocessor, it would be interesting to see whether the low compression ratio for Metal1 and Metal2 layers seen for the microprocessor carries over to an ASIC. To this end, we have applied Block C4 to a low density parity check (LDPC) decoder ASIC chip in the 65-nm technology, with layout placement and routing generated using Synopsys Astro. Assuming a pixel size of 32 nm, each block is 1024×1024 pixels, or 32×32 μm. Figure 9 shows the histogram of compression ratio for Metal1 and Metal2 layers. For the Metal1 layer, we have applied the routing tool twice in order to generate two different layout densities, namely, 86 and 90%. The chip contains 1291 cells. The gds sizes for Metal1 (86%), Metal1 (90%), and Metal2 are 536, 517, and 779 MB, respectively.

As expected, the compression ratio for Metal1 drops as the density goes up. In addition, the distribution of compression ratio for Metal1 is to the left of that of Metal2, indicating that Metal1 blocks are harder to compress than those of Metal2. Metal1 contains optimally dense wires inherent to each standard cell and between neighboring cells, while Metal2 wires are used to connect nearby cells. Thus, despite the presence of easily compressible power and ground rails on the Metal1 layer, Metal1 is consistently more difficult to compress than Metal2, which often contains large blank spaces in areas, where intercell routing is straightforward.

More importantly, the minimum compression ratio for Metal1 (86%), Metal1 (90%), and Metal2 (86%) are 14.3, 13.2, and 18.7, respectively. These minimum compression ratios are considerably higher than those of the microprocessor considered earlier. The 1024×1024 blocks corresponding to minimum compression ratio for M1 (90%) and M2 (86%) are shown in Figs. 10(a) and 10(b), respectively.

7 Discussion

In summary, compression can play an important role in most layers and its shortcomings can be mitigated through careful engineering of the overall maskless lithography data path and design layout. In addition, Block C4 has shown itself as a strong candidate for implementation in the maskless lithography datapath shown in Fig. 1, with the lowest decoder buffering requirement of 1.7 KB,[1] low decoder complexity in software, high-compression efficiency, and a reasonable and predictable compression speed in software.

To use Block C4 in a maskless lithography datapath, we need to transform the implementation of a Block C4 decoder from software to silicon hardware. These developments, including decomposition of the algorithm into functional blocks, area, and power are discussed in Refs. 5 and 16.

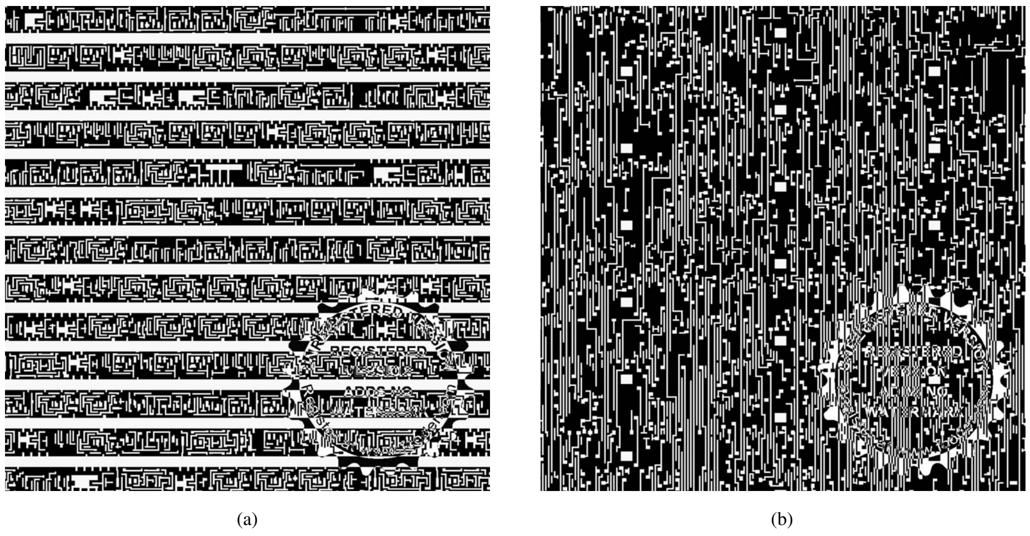


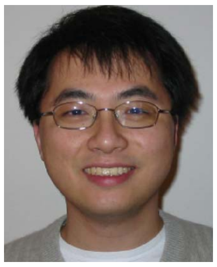
Fig. 10 Lowest compression ratio blocks of LDPC chip for (a) Metal1 (90%) with CR of 13.2 and (b) Metal2 (86%) with CR of 18.7.

Acknowledgments

This research is conducted under the Research Network for Advanced Lithography, supported jointly by SRC (01-MC-460) and DARPA (MDA972-01-1-0021). The authors gratefully acknowledge the contributions of Zhengya Zhang, Hsin-I Liu, Prof. Borivoje Nikolic, Brian Richards, the students, faculty, and sponsors of the Berkeley Wireless Research Center, the National Science Foundation Infrastructure Grant No. 0403427, and the wafer fabrication donation from STMicroelectronics.

References

1. V. Dai and A. Zakhor, "Advanced low-complexity compression for maskless lithography data," *Proc. SPIE* **5374**, 610–618 (2004).
2. V. Dai and A. Zakhor, "Lossless compression techniques for maskless lithography data," *Proc. SPIE* **4688**, 583–594 (2002).
3. V. Dai, "Binary lossless layout compression algorithms and architectures for direct-write lithography systems," MS Thesis, Department of Electrical Engineering and Computer Sciences, University of California–Berkeley; see ⟨http://www-video.eecs.berkeley.edu/papers/vdai/ms-thesis.pdf⟩ (2000).
4. V. Dai and A. Zakhor, "Lossless layout compression for maskless lithography systems," *Proc. SPIE* **3997**, 467–477 (2000).
5. H. Liu, V. Dai, A. Zakhor, and B. Nikolic, "Reduced complexity compression algorithms for direct-write maskless lithography systems," *J. Micro/Nanolith. MEMS MOEMS* **6**, 013007 (2007).
6. N. Chokshi, D. S. Pickard, M. McCord, R. F. W. Pease, Y. Shroff, Y. Chen, W. G. Oldham, and D. Markle, "Maskless extreme ultraviolet lithography," *J. Vac. Sci. Technol. B* **17**(6), 3047–3051 (1999).
7. E. M. Stone, J. D. Hintersteiner, W. A. Cebuhar, R. Albright, N. K. Eib, A. Latypov, N. Baba-Ali, S. K. Poultney, and E. H. Croffie, "Achieving mask-based imaging with optical maskless lithography," in *Emerging Lithographic Technologies X*, *Proc. SPIE* **6151**, 665–676 (2006).
8. A. Murray, F. Abboud, F. Raymond, and C. N. Berglund, "Feasibility study of new graybeam writing strategies for raster scan mask generation," *J. Vac. Sci. Technol.* **11**, 2390 (1993).
9. J. Chabala, F. Abboud, C. A. Sauer, S. Weaver, M. Lu, H. T. Pearce-Percy, U. Hofmann, M. Vernon, D. Ton, D. M. Cole, and R. J. Naber, "Extension of graybeam writing for the 130 nm technology node," *Proc. SPIE* **3873**, 36–48 (1999).
10. D. H. Dameron, C. Fu, and R. F. W. Pease, "A multiple exposure strategy for reducing butting errors in a raster-scanned electron-beam exposure system," *J. Vac. Sci. Technol. B* **6**(1), 213–215 (1988).
11. P. C. Allen, "Laser pattern generation technology below 0.25um," *Proc. SPIE* **3334**, 460–468 (1998).
12. H. Martinsson and T. Sandstrom, "Rasterizing for SLM-based mask making and maskless lithography," *Proc. SPIE* **5567**, 557–564 (2004).
13. M. J. Weinberger, G. Seroussi, and G. Sapiro, "The LOCO-I lossless image compression algorithm: principles and standardization into JPEG-LS," *IEEE Trans. Image Process.* **9**(8), 1309–1324 (2000).
14. J. Ziv and A. Lempel, "A universal algorithm for sequential data compression," *IEEE Trans. Inf. Theory* **IT-23**(3), 337–343 (1977).
15. M. Burrows and D. J. Wheeler, "A block-sorting lossless data compression algorithm," Technical Report No. 124, Digital Equipment Corp., Palo Alto, CA (1994).
16. V. Dai, "Data compression for maskless lithography systems: architecture, algorithms and implementation," Ph.D. Thesis, University of California–Berkeley (2008).
17. B. Nikolic, B. Wild, V. Dai, Y. Shroff, B. Warlick, A. Zakhor, and W. G. Oldham, "Layout decompression chip for maskless lithography," *Proc. SPIE* **5374**(1), 1092–1099 (2004).
18. HyperTransport Consortium, ⟨http://www.hypertransport.org⟩.
19. K. Chang, S. Pamarti, K. Kaviani, E. Alon, X. Shi, T. J. Chin, J. Shen, G. Yip, C. Madden, R. Schmitt, C. Yuan, F. Assaderaghi, and M. Horowitz, "Clocking and circuit design for a parallel I/O on a first-generation CELL processor," presented at *Int. Solid-State Circuit Conf.*, IEEE, Piscataway, NJ (2005).
20. B. Wild, "Data handling circuitry for maskless lithography systems," MS Thesis, University of California–Berkeley (2001).



Vito Dai received his BS in electrical engineering from California Institute of Technology in 1998, and his MS and PhD in electrical engineering at the University of California at Berkeley in 2000 and 2008, respectively. He is currently a senior process engineer in the OPC/DFM Automation Group at GLOBALFOUNDRIES, where he works on developing new methodologies for improving the lithographic printability of designs through 2-D pattern analysis for the leading-edge semiconductor manufacturing.

Avideh Zakhor joined the faculty at University of California, Berkeley in 1988 where she is currently a professor of electrical engineering and computer sciences. Her areas of interest include theories and applications of signal, image and video processing, 3-D computer vision, and multimedia networking. She has won a number of best paper awards, including the IEEE Signal Processing Society in 1997 and 2009, IEEE Circuits and Systems Society in 1997 and 1999, international conference on image processing in 1999, Packet Video Workshop in 2002, and IEEE Workshop on Multimodal Sentient Computing in 2007. She holds six U.S. patents, and is the coauthor of three books with her students. She has been a PI or co-PI of four different MURI projects. Prof. Zakhor received her BS from California Institute of Technology, Pasadena, and her SM and PhD from Massachusetts Institute of Technology, Cambridge, all in electrical engineering, in 1983, 1985, and 1987 respectively. She was a General Motors scholar from 1982 to 1983, a Hertz fellow from 1984 to 1988, received the Presidential Young Investigators award, and Office of Naval Research young investigator award in 1992. In 2001, she was elected as an IEEE fellow and received the Okawa Prize in 2004. She cofounded OPC technology in 1996, which was later acquired by Mentor Graphics in 1998, Truvideo in 2000, and UrbanScan Inc. in 2005, which was acquired by Google in 2007.

George Cramer received his BE in electrical engineering from The Cooper Union in 2007. He completed his MS at U.C. Berkeley in Electrical Engineering in 2009, and is a member of the Video and Image Processing Laboratory. His research interests include analog integrated circuits, communications, and energy.