Lossless Compression Algorithms for Direct-Write Lithography Systems

Hsin-I Liu

Video and Image Processing Lab
Department of Electrical Engineering and Computer Science
University of California at Berkeley
Optical Lithography

- Lithography is applied to create patterns on the wafer in semiconductor manufacturing
- Current approach: Mask is applied in optical lithography systems
  - cost of mask is increasing......
# From Mask to Maskless Lithography

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Technology Node (nm)</td>
<td>90</td>
<td>65</td>
<td>45</td>
<td>32</td>
<td>22</td>
<td>16</td>
<td>11</td>
<td>8</td>
</tr>
</tbody>
</table>

Source: ITRS 2004
Cost of Masks in Optical Lithography

ITRS 2009
A micromirror array is used to replace the optical mask
- Reduce the cost of mask by x times
- Increase patterning flexibility

Focus of research:
- Fabricate micromirror → array
- Modify the layout pattern for proximity effect correction → OPC or EPC

However......
• Each micromirror is controlled individually and dynamically
• Layout image is rasterized into pixel based
  → Data delivery problem for real-time manufacturing
• Update the pixel value for
  - Different portion of layout images
  - Overcome the voltage attenuation problem
Data Delivery Issue

- Data rate for 45nm minimum feature to achieve 1 wafer layer/minute throughput
  \[
  \frac{\text{wafer \cdot layer}}{60 \text{ s}} \times \frac{\pi/4 \cdot (300 \text{ mm})^2}{\text{wafer \cdot layer}} \times \frac{\text{pixel}}{(22 \text{ nm})^2} \times \frac{5 \text{ bits}}{\text{pixel}} = 12 \text{ Tb/s}
  \]
- Estimated needed compression: \(12 \text{ Tb/s} \div 1.2 \text{ Tb/s} = 10\)
- Board to chip communication: 1.2 Tb/s
  - e.g. 128 pins @ 6.4 GHz
- Throughput requirement can be reduced to 3-5 wafer layers per hour → still need compression
- Lossless compression is applied to
  - Reduce storage space
  - Lower I/O throughput overhead

![Diagram of data delivery system](chart.jpg)
Data Compression Requirements

- Lossless compression
- Achieve ~10 compression efficiency
- Asymmetric compression algorithms
  - Offline encoding
  - Real-time decoding → decoder is implemented in hardware and integrated into the writer system
Block GC3 - Compression Algorithm for Rasterized, Flattened Layout

Block Golomb context copy code (Block GC3)

- Prediction from Context - JBIG
  1. Predict a pixel value from neighboring pixels (P)
  2. Good for non-repetitive layouts

[H. Liu ‘06]
Block GC3 - Context Predict

<table>
<thead>
<tr>
<th>Prediction</th>
<th>Prediction Error</th>
<th>Empirical Error Prob.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>0.6%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7.1%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3.9%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.0%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.0%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2.2%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3.7%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.3%</td>
</tr>
</tbody>
</table>

\[ x = b - a + c \]

- if \( x < 0 \) then \( z = 0 \)
- if \( x > \text{max} \) then \( z = \text{max} \)
- otherwise \( z = x \)
Block GC3 - Copy

- Copying - ZIP, 2D-LZ
  1. Copy from left or above
  2. Good for repetitive layouts
Block GC3 - Segmentation

- Layout images are divided into prediction and copy regions
  - Determined within 8 x 8 block
- Errors from prediction and copy are transmitted from Encoder to decoder
- All the information is further compressed
Block GC3 Encoder/Decoder Architecture

- Outperform the existing techniques
- Simple decoder design

[V. Dai ’05]
Golomb Run-Length Code

- A simple code for binary stream

```
000100000000001100101......
```

Bucket Size (B): maximum # of zeroes in a row

\[ B = 4 \]

Two kind of codes: 

- \((0)\) \(\rightarrow\) B zeros in a row

\((1, n)\) \(\rightarrow\) n zeros in a row followed by a one

\[(1,3) (0) (0) (1,2)(1,0)(1,2) (1,1)\]

Compression achieved

Additional information introduced
Golomb Run-Length Code

- A simple code for binary stream

\[
000100000000001100101\ldots
\]

Bucket Size (B): maximum # of zeroes in a row

\[B = 4\]

Two kind of codes: 

- \((0)\) \rightarrow B zeros in a row
- \((1, n)\) \rightarrow n zeros in a row followed by a one

\[(1,3) (0) (0) (1,2) (1,0) (1,2) (1,1)\]

Golomb code achieves its best compression efficiency in i.i.d. random variables

\[\rightarrow\] achieves inefficient compression with highly skewed bitstream such as error location

simple decoder design
Complexity vs. Compression Ratio of Compression Schemes

[Desirable operating point]

[H. Liu '06]
Full-Chip Test

- 24% of the images have CR < 5

[AMD CPU 65 nm Metal-1] [A. Zakhor ’09]
Full-Chip Test

LDPC Decoder - Block C4

Compression Ratio vs. Probability Density Function

- Metal 1 (80% density)
- Metal 2 (88% density)
- Metal 1 (90.5% density)

ST ASIC 65 nm

[A. Zakhor ‘09]
Block Diagram of Block GC3 Decoder

- High parallelism for hardware implementation → Data flow architecture
Data Flow of Decoder

- Address Generator
  - address
- Region Decoder
  - l/a, d
  - predict/copy
  - pixel value
- Huffman
  - error value
- Linear Prediction
  - pixel value
- History Buffer
  - pixel value
- Golomb
  - error location
- mux
  - output
After the decoding, the pixel value is stored back to history buffer.
Data Flow of Decoder - Copy

- After the decoding, the pixel value is stored back to history buffer
Data Flow of Decoder - Error

- After the decoding, the pixel value is stored back to history buffer
## Decoder Performance - FPGA

<table>
<thead>
<tr>
<th>Device</th>
<th>Xilinx Virtex II Pro 70</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of slice flip-flops</td>
<td>3,233 (4%)</td>
</tr>
<tr>
<td>Number of 4 input LUTs</td>
<td>3,086 (4%)</td>
</tr>
<tr>
<td>Number of block RAMs</td>
<td>36 (10%)</td>
</tr>
<tr>
<td>System clock rate</td>
<td>100 MHz</td>
</tr>
<tr>
<td>System throughput rate</td>
<td>0.99 (pixels/clock cycle)</td>
</tr>
<tr>
<td>System output data rate</td>
<td>495 Mb/s</td>
</tr>
</tbody>
</table>

- The hardware performance can be improved
  - Update FPGA devices
  - Apply ASIC implementation
# Decoder Performance - ASIC

<table>
<thead>
<tr>
<th>Block</th>
<th>Area (um²)</th>
<th>Throughput (output/cycle)</th>
<th>Power (mW)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Golomb</td>
<td>1,136</td>
<td>1</td>
<td>0.2</td>
</tr>
<tr>
<td>Huffman</td>
<td>848</td>
<td>1/codeword+2</td>
<td>0.21</td>
</tr>
<tr>
<td>Linear Prediction</td>
<td>455</td>
<td>1</td>
<td>0.16</td>
</tr>
<tr>
<td>Address Generator</td>
<td>362</td>
<td>0.99</td>
<td>0.03</td>
</tr>
<tr>
<td>Region Decoder</td>
<td>18,370</td>
<td>1</td>
<td>7.26</td>
</tr>
<tr>
<td>Control/Merge</td>
<td>749</td>
<td>1</td>
<td>0.22</td>
</tr>
<tr>
<td>Memory</td>
<td>46,960</td>
<td>1</td>
<td>13.27</td>
</tr>
<tr>
<td>Block GC3</td>
<td>69,288</td>
<td>0.99</td>
<td>21.48</td>
</tr>
</tbody>
</table>

- 85% of area results from 1.7 KB of memory
- System clock rate: up to 500 MHz
- System throughput: 0.99
- System output rate: up to 2.47 Gb/s
- 200 decoders to achieve 500 Gb/s → 3 wafer layers per hour
Apply Block GC3 to reduce I/O overhead

<table>
<thead>
<tr>
<th>I/O Type</th>
<th>Data rate</th>
<th># of link for 500 Gb/s</th>
<th># of link with Block GC3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cell I/O</td>
<td>6.4 Gb/s</td>
<td>80</td>
<td>12</td>
</tr>
<tr>
<td>Hyper Transport 3.1</td>
<td>6.4 Gb/s</td>
<td>80</td>
<td>12</td>
</tr>
<tr>
<td>Optical link</td>
<td>3 Gb/s</td>
<td>167</td>
<td>26</td>
</tr>
<tr>
<td>Intel 65 nm interface</td>
<td>10 Gb/s</td>
<td>50</td>
<td>8</td>
</tr>
<tr>
<td>Intel 45 nm interface</td>
<td>25 Gb/s</td>
<td>20</td>
<td>3</td>
</tr>
</tbody>
</table>

- 200 Block GC3 decoders is 14 mm²
- Reduced I/O interface is more practical for direct-write applications
Writer Chip Architecture

- DRAM array directly controls the micromirror array above
- Throughput of the chip: 3 wafer·layer/hour (500Gb/s)
Encoding complexity of Block GC3

- Find best copy distance → the most computational challenging part of encoding
Find the Best Copy Distance

• For an m x n image with block size M, the complexity is

\[ O \left( \frac{mn}{M^2} (d_x + d_y) \right) \]

Memory size = \( d_x \times d_y \)

• Block segmentation reduces the complexity by \( M^2 \)

• For linear writing system, horizontal/vertical copy is sufficient
Find the Best Copy Distance - Multiple Candidates

- Every block may have more than one candidates with fewest mismatches → enforce spatial coherency for better compression
- Region growing → use the fewest number of regions to represent the segmentation map
Region Growing

• 2-D region growing is an NP-complete problem
  - Use left/above segmentation info as preferences

\[
\begin{array}{c}
  a \\
  b \\
  c \\
  ?
\end{array}
\]

  If \(a = c\) then \(? = b\) else \(? = c\)

• 1-D region growing can be solved in polynomial time
  - A better solution for complex segmentation maps
Improve Compression Efficiency

• For linear writing system and ASIC layout images → average CR > 10
• For different writing system or compact layout → modify encoding scheme to improve compression efficiency
  - REBL system
REBL Direct-Write Lithography System

- Rotary writer ➔ spiral writing
- $45^\circ$ between the radius of the stage and the die

[P. Petric et. al., KLA-Tencor, 08]
• Layout pattern created by digital pattern generator (DPG)
  - 256 rows per DPG, 16 DPGs in total
  - Column by column writing mechanism
• Layout angle orientation: 15° to 75°
  \[\pm 30° + 45°\]
• E-beam proximity corrected
Lossless Compression Algorithm for REBL- Block RGC3

- Allow diagonal copying
- Reduce block size and dimension
- Apply 1-D region growing to reduce numbers of regions
- Increase memory size
- Encoding complexity

\[ O\left[ \frac{mn}{HW} (d_x \times d_y) \right] \]

Memory size = \( d_x \times d_y \)
Compression Results

<table>
<thead>
<tr>
<th>Buffer size</th>
<th>Block GC3</th>
<th>Block RGC3</th>
<th>ZIP</th>
<th>BZip2</th>
<th>JPEG-LS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.6KB 4x4</td>
<td>1.6KB 4x4</td>
<td>1.6KB 5x3</td>
<td>32KB</td>
<td>900KB</td>
<td>2.2KB</td>
</tr>
<tr>
<td>20KB 4x4</td>
<td>20KB 4x4</td>
<td>20KB 5x3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>40KB 4x4</td>
<td>40KB 4x4</td>
<td>40KB 5x3</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Layout size

<table>
<thead>
<tr>
<th>Layout size</th>
<th>2048x64</th>
<th>1024x256</th>
<th>2048x256</th>
</tr>
</thead>
<tbody>
<tr>
<td>2048x64</td>
<td>3.13</td>
<td>3.37</td>
<td>3.19</td>
</tr>
<tr>
<td>1024x256</td>
<td>3.19</td>
<td>3.30</td>
<td>3.19</td>
</tr>
<tr>
<td>2048x256</td>
<td>3.19</td>
<td>3.30</td>
<td>3.19</td>
</tr>
</tbody>
</table>

- Block RGC3 outperforms Block GC3 and others
- Larger buffer size, larger image size → better compression ratio
- 50 - 69% of improvement due to diagonal copying
  - more effective as buffer size increases

<table>
<thead>
<tr>
<th>Image size</th>
<th>H / V Copying</th>
<th>Diagonal Copying</th>
</tr>
</thead>
<tbody>
<tr>
<td>64x2048</td>
<td>3.44</td>
<td>5.22</td>
</tr>
<tr>
<td>256x2048</td>
<td>3.37</td>
<td>5.71</td>
</tr>
</tbody>
</table>

25° Metal 1 layout
Results for Various Wafer Layers

<table>
<thead>
<tr>
<th>Image size</th>
<th>Buffer size</th>
<th>Metal 1 Memory 25°</th>
<th>Metal 1 Memory 35°</th>
<th>Metal 1 Memory 38°</th>
<th>Metal 1 Logic 25°</th>
<th>Poly 25°</th>
<th>Poly 35°</th>
<th>Poly 35°</th>
<th>Via 25°</th>
<th>Via 35°</th>
</tr>
</thead>
<tbody>
<tr>
<td>64x2048</td>
<td>1.7KB</td>
<td>4.92</td>
<td>5.37</td>
<td>5.14</td>
<td>--</td>
<td>8.49</td>
<td>13.14</td>
<td>12.67</td>
<td></td>
<td></td>
</tr>
<tr>
<td>256x1024</td>
<td>1.7KB</td>
<td>5.09</td>
<td>5.43</td>
<td>5.33</td>
<td>8.55</td>
<td>8.47</td>
<td>13.58</td>
<td>13.17</td>
<td></td>
<td></td>
</tr>
<tr>
<td>256x2048</td>
<td>1.7KB</td>
<td>5.10</td>
<td>5.45</td>
<td>5.35</td>
<td>--</td>
<td>8.51</td>
<td>13.62</td>
<td>13.22</td>
<td></td>
<td></td>
</tr>
<tr>
<td>64x2048</td>
<td>20KB</td>
<td>6.54</td>
<td>6.68</td>
<td>6.63</td>
<td>--</td>
<td>11.17</td>
<td>15.31</td>
<td>15.40</td>
<td></td>
<td></td>
</tr>
<tr>
<td>256x1024</td>
<td>20KB</td>
<td>6.91</td>
<td>7.08</td>
<td>7.11</td>
<td>14.06</td>
<td>12.50</td>
<td>16.14</td>
<td>16.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>256x2048</td>
<td>20KB</td>
<td>7.01</td>
<td>7.20</td>
<td>7.22</td>
<td>--</td>
<td>12.77</td>
<td>16.35</td>
<td>16.22</td>
<td></td>
<td></td>
</tr>
<tr>
<td>64x2048</td>
<td>40KB</td>
<td>6.60</td>
<td>6.79</td>
<td>6.71</td>
<td>--</td>
<td>11.91</td>
<td>15.86</td>
<td>16.11</td>
<td></td>
<td></td>
</tr>
<tr>
<td>256x1024</td>
<td>40KB</td>
<td>7.12</td>
<td>7.23</td>
<td>7.34</td>
<td>14.87</td>
<td>12.80</td>
<td>17.05</td>
<td>17.27</td>
<td></td>
<td></td>
</tr>
<tr>
<td>256x2048</td>
<td>40KB</td>
<td>7.29</td>
<td>7.41</td>
<td>7.50</td>
<td>--</td>
<td>13.17</td>
<td>17.45</td>
<td>17.79</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Higher compression ratio for via than metal 1
- Larger buffer size, larger image size ➔ better compression ratio
Encoding Time

(1) Diagonal copying
- Must compare each image block with each copy distance
  \[ O\left( \frac{1}{\text{buffer \_ size}} + \frac{1}{\beta} \right) \]  
  \[ \beta \approx 10 \]

(2) Growing regions
- Proportional to avg. # optimal copy distances per block
  \[ O\left( \frac{d_{\text{matches,block}}}{\text{block \_ size}} \right) \]

(3) Combining regions
- Proportional to avg. # optimal copy distances per region
- Inversely proportional to # of blocks per region \( \overline{N} \)
  \[ O\left( \frac{d_{\text{matches,region}}}{N \cdot \text{block \_ size}} \right) = O\left( \frac{d_{\text{matches,region}}}{\text{region \_ size}} \right) \]
### Encoding Times

<table>
<thead>
<tr>
<th>Image size</th>
<th>Buffer size</th>
<th>Diagonal copying</th>
<th>Region-growing</th>
<th>Combining regions</th>
<th>Total encoding time (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Metal1 25°</td>
<td>Via 25°</td>
<td>Metal1 25°</td>
<td>Via 25°</td>
</tr>
<tr>
<td>64×2048</td>
<td>20KB</td>
<td>95.4%</td>
<td>85.5%</td>
<td>4.3%</td>
<td>13.0%</td>
</tr>
<tr>
<td>256×1024</td>
<td>20KB</td>
<td>95.2%</td>
<td>85.1%</td>
<td>4.2%</td>
<td>13.8%</td>
</tr>
<tr>
<td>64×2048</td>
<td>40KB</td>
<td>96.1%</td>
<td>84.9%</td>
<td>3.6%</td>
<td>14.0%</td>
</tr>
<tr>
<td>256×1024</td>
<td>40KB</td>
<td>95.6%</td>
<td>81.1%</td>
<td>4.0%</td>
<td>18.0%</td>
</tr>
</tbody>
</table>

- Dominant factor ➔ Diagonal copying for best copy distance
- Encoding time proportional to buffer size, image size
Encoding time normalized to microsecond per pixel

- Smaller buffer size ⇒ lower CR and lower encode time
- Block RGC3 additional encode complexity justifiable if higher compression ratios are needed:
  - Metal 1: Higher than 3.5
  - Poly: Higher than 6
  - Via: Higher than 12
Integrating Block GC3 with Writer Systems

- Need to modify the algorithm to achieve best compression efficiency
  - May increase encoding complexity
  - Remain same decoding structure
  - Remain asymmetric compression algorithm
Summary

• Block GC3 solves data delivery problem for direct-write lithography systems

• Implement Block GC3
  - Block GC3 reduces: I/O data rate, System power → the goal

• Block RGC3 improves compression ratio for REBL system
  - Increase encoder complexity
  - Decoder complexity remains low