Efficient Design Rule Checking with GPU Acceleration

Wei Zhong¹, Zhenhua Feng¹, Zhuolun He², Weimin Wang¹, Yuzhe Ma³, Bei Yu²

¹Dalian University of Technology
²Chinese University of Hong Kong
³HKUST(GZ)

Apr. 18, 2023
Introduction
Design rule checking (DRC) is the process to verify that a design layout conforms to a set of predefined design rules.
• Width check
• Space check
• Enclosing check

(a) Example of width check; (b) Example of space check; (c) Example of enclosing check.
Algorithms
The overview of our GPU-accelerated design rule checking flow.

- Clip Layout into Multiple Regions
- Merge Contacting Polygons in Each Region
- Copy Merged Region to GPU
- Sort Horizontal And Vertical Edges of The Polygon in The Region
- Check Inside of Polygon
- Box Overlap Check to Find The Conflict Polygon Pairs
- Scanline Between Polygons to Find Candidate Edge Pairs
- Check Out Conflicting Edge Pairs among Candidate Edge Pairs
- Scanline to Find The Candidate Edge Pairs

Overview of our GPU-accelerate DRC.
The method of transferring layout information from the CPU to the GPU.

Encoding and decoding of a layout. Polygon edges are separated and stored uniformly in an edge matrix. The compact representation reduces data transfer overhead.
• Scanline inside polygon.
• Overlap checking.
• Scanline Between Polygons.
Programming architecture for the scanline algorithm.
Suppose there are two vertical edges $e_1 = x_1 \times [y_{11}, y_{12}]$ and $e_2 = x_2 \times [y_{21}, y_{22}]$. If their distance is smaller than the threshold $\epsilon$:

$$|x_1 - x_2| < \epsilon,$$  \hspace{1cm} (1)

they will be recorded as a candidate edge pair. Similarly, for horizontal $e_3 = [x_{31}, x_{32}] \times y_3$ and $e_4 = [x_{41}, x_{42}] \times y_4$, they will be recorded when the condition $|y_3 - y_4| < \epsilon$ holds.
For any two polygons $P^{(i)}$ and $P^{(j)}$, threshold $\epsilon$, if any one of the following formulas is satisfied, they will be regarded as a pair of polygons with the possibility of violation:

\[
P_{ll}^{(i)} \cdot x < P_{ur}^{(j)} \cdot x + \epsilon, \quad (2)
\]
\[
P_{ll}^{(j)} \cdot x < P_{ur}^{(i)} \cdot x + \epsilon, \quad (3)
\]
\[
P_{ll}^{(i)} \cdot x < P_{ur}^{(j)} \cdot y + \epsilon, \quad (4)
\]
\[
P_{ll}^{(j)} \cdot x < P_{ur}^{(i)} \cdot y + \epsilon. \quad (5)
\]

Two typical overlapping examples.
Polygon A and Polygon B are detected using the scanline algorithm between polygons.
Experimental result
Experimental setup

- A 64-bit Ubuntu Linux machine with TITAN RTX GPU and 3.5GHz Intel Core i9-10920X CPU.
- The compilers include CUDA NVCC 10.2 and GNU GCC 5.4.0.
- 4096 threads for all kernel configurations and 1 CPU core for all host operations.
- Baseline is KLayout 0.26.6.
- The layout benchmarks in the experiments are generated by OpenROAD project with default settings.
Space check results with (a) DRC script and (b) ours; Enclosing check results with (c) DRC script and (D) ours.
## Runtime Performance

**Table**: Enclosing check in Metall

<table>
<thead>
<tr>
<th>Design</th>
<th>gcd</th>
<th>aes</th>
<th>bp_be</th>
<th>bp</th>
</tr>
</thead>
<tbody>
<tr>
<td>8 CPU threads</td>
<td>33.522</td>
<td>13194.039</td>
<td>58477.239</td>
<td>90250.85</td>
</tr>
<tr>
<td>16 CPU threads</td>
<td>34.212</td>
<td>13074.176</td>
<td>51671.131</td>
<td>85792.708</td>
</tr>
<tr>
<td>24 CPU threads</td>
<td>34.52</td>
<td>13072.36</td>
<td>49047.536</td>
<td>74497.754</td>
</tr>
<tr>
<td>Ours</td>
<td>0.343</td>
<td>27.932</td>
<td>257.056</td>
<td>409.381</td>
</tr>
</tbody>
</table>

**Speedup**
- gcd: 100.641×
- aes: 468.01×
- bp_be: 190.80×
- bp: 181.98×

**Average**: 201.1×
Table: Enclosing check in Meta12

<table>
<thead>
<tr>
<th>Design</th>
<th>gcd</th>
<th>aes</th>
<th>bp_be</th>
<th>bp</th>
</tr>
</thead>
<tbody>
<tr>
<td>8 CPU threads</td>
<td>5.547</td>
<td>1977.047</td>
<td>2859.979</td>
<td>3332.67</td>
</tr>
<tr>
<td>16 CPU threads</td>
<td>5.732</td>
<td>1997.85</td>
<td>2435.594</td>
<td>2321.697</td>
</tr>
<tr>
<td>24 CPU threads</td>
<td>5.552</td>
<td>1976.503</td>
<td>2320.845</td>
<td>2298.961</td>
</tr>
<tr>
<td>Ours</td>
<td>0.291</td>
<td>30.493</td>
<td>132.717</td>
<td>250.022</td>
</tr>
<tr>
<td>Speedup Average</td>
<td>19.08×</td>
<td>64.82×</td>
<td>17.49×</td>
<td>11.21×</td>
</tr>
<tr>
<td>Average</td>
<td>22.19×</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Runtime Performance

### Table: Space check in Metall

<table>
<thead>
<tr>
<th>Design</th>
<th>gcd</th>
<th>aes</th>
<th>bp_be</th>
<th>bp</th>
</tr>
</thead>
<tbody>
<tr>
<td>8 CPU threads</td>
<td>10.99</td>
<td>376.244</td>
<td>5950.007</td>
<td>14865.705</td>
</tr>
<tr>
<td>16 CPU threads</td>
<td>11.131</td>
<td>3692.87</td>
<td>4540.09</td>
<td>8833.2</td>
</tr>
<tr>
<td>24 CPU threads</td>
<td>10.989</td>
<td>3690.08</td>
<td>4226.62</td>
<td>7565.84</td>
</tr>
<tr>
<td>Ours</td>
<td>0.316</td>
<td>19.091</td>
<td>250.799</td>
<td>471.71</td>
</tr>
</tbody>
</table>

| Speedup Average  | 34.78×  | 193.29×  | 16.85×  | 16.04×  |
|                  |        |        | 36.71×   |        |
## Runtime Performance

**Table:** Space check in Metal2

<table>
<thead>
<tr>
<th>Design</th>
<th>gcd</th>
<th>aes</th>
<th>bp_be</th>
<th>bp</th>
</tr>
</thead>
<tbody>
<tr>
<td>8 CPU threads</td>
<td>6.378</td>
<td>2732.5534</td>
<td>3870.166</td>
<td>5015.233</td>
</tr>
<tr>
<td>16 CPU threads</td>
<td>6.168</td>
<td>2703.47</td>
<td>3365.211</td>
<td>3767.176</td>
</tr>
<tr>
<td>24 CPU threads</td>
<td>6.174</td>
<td>2666.539</td>
<td>3114.918</td>
<td>3621.914</td>
</tr>
<tr>
<td>Ours</td>
<td>0.399</td>
<td>28.591</td>
<td>121.279</td>
<td>238.966</td>
</tr>
<tr>
<td>Speedup Average</td>
<td>15.47×</td>
<td>93.26×</td>
<td>25.68×</td>
<td>15.16×</td>
</tr>
<tr>
<td>Average</td>
<td></td>
<td></td>
<td>27.38×</td>
<td></td>
</tr>
</tbody>
</table>
Conclusion
• An efficient data packing method customized for layout data transfer.
• A GPU-based scanline algorithm that can perform parallel scanning of complex graphics.
• Speedup of 36× when performing space check on the Metal1 layer.
• Speedup of 201× when performing enclosing check of the Metal1 layer.
THANK YOU!