OpenDRC: An Efficient Open-Source Design Rule Checking Engine with Hierarchical GPU Acceleration

Zhuolun He\textsuperscript{1,2}, Yihang Zuo\textsuperscript{2}, Jiaxi Jiang\textsuperscript{2}, Haisheng Zheng\textsuperscript{2}, Yuzhe Ma\textsuperscript{3}, Bei Yu\textsuperscript{1}

\textsuperscript{1}CUHK \hspace{1cm} \textsuperscript{2}Shanghai AI Lab \hspace{1cm} \textsuperscript{3}HKUSZ(GZ)
Design Rule Checking\(^1\)...

- verifies a deck of layout constraints
- consists of complex rules nowadays
  - geometric, inter-layer, conditional rules...
- is ultra time-consuming in the design flow

\(^1\)Figure from ASAP7 design rule manual
There are several ways to improve DRC efficiency:

• **better algorithms/data structures**
  - sweepline, quad-tree, r-tree, ...

• **parallel computing**
  - region-based, design hierarchy, edge-based, task parallelism, ...
  - on SIMD engines, specialized hardware, distributed systems, GPUs, ...

• **approximation methods**
  - hotspot detection, violation type/number prediction, ...
  - other ML-enhanced DRC schemes
Open-source EDA tools have been inspiring and empowering the evolution of cutting-edge EDA research.

Open-source ‘design rule checkers’ often appear in

- detailed routers (e.g., TritonRoute\textsuperscript{2})
  - not for verification purpose
  - tightly coupled with path searching
- layout editors (e.g., Magic\textsuperscript{3}, KLayout\textsuperscript{4})
  - GUI-centric, not optimized for standalone checking
  - different data structure for efficient editing


We feel that a new (open-source) design rule checking engine is necessary!

This work proposes OpenDRC, which

- aims for extremely high efficiency
- supports hierarchical designs
- provides GPU acceleration
- is available at https://github.com/opendrc/opendrc
OpenDRC Overall Flow

Sequential Mode
Parallel Mode
Rules from API
Adaptive partition
Layout in BVH
Hierarchical DRC

• Core data structure: Layer-wise *Bounding Volume Hierarchy* (BVH)
  • does not flatten the layout; preserves design hierarchy
  • maintains *minimum bounding rectangle* (MBR) of cells

• Redundant check elimination

![Diagram showing examples of some are trivial duplications, some results could be reused, and some violations cannot happen.]

• some are trivial duplications: \((a^M, b_1^N)\) and \((b_1^M, a^N)\) when \(M = N\)

• some results could be reused: \((a^M, a^N)\) and \((b_2^M, b_2^N)\) when \(b_2 = a\)

• some violations cannot happen: \((a^M, b_3^N)\)
Intuition (due to row-based placement):

- Layouts can be partitioned into non-overlapping regions (rows) along the y-axis
- By such grouping, x-coordinates of cells in a row are also likely to be separated

Solution: adaptive row-based partition

- solvable as interval merging problem in $\Theta(k + N)$ or $\Theta(k \log k)$
- $k$ is the number of cells, $N$ is the number of unique coordinates ($k \gg N$)
Interval Merging

**Require:** A set $S$ of intervals to be merged

**Ensure:** Non-overlapping intervals covering the domain of $S$

1. Initialize an array $A$ with indices
2. for all interval $[l, r] \in S$ do
3. Update $A[l] \leftarrow \max(A[l], r)$
4. end for
5. Initialize current interval end $e \leftarrow -1$
6. for the $i$-th element $\in A$ do
7. if $i > e$ then
8. Create a new interval and reset $e$
9. end if
10. Update current interval end $e \leftarrow \max(e, A[i])$
11. end for

▷ Step 1: Initialize
▷ Step 2: Merge
▷ Step 3: Scan

▷ moving across interval boundary
• Standard sweepline algorithm\(^5\) (but we use interval tree)

Parallel Mode w/ GPU Acceleration

- parallel DRC row-by-row
- edges of polygons are packed into a flattened array and transferred to device memory
- parallel sweepline\(^6\)
  1. parallel scan to determine check range
  2. parallel edge to edge(s) check

\[ T_1 = [a, a] \]
\[ T_2 = [a, c] \]
\[ T_3 = [b, d] \]
\[ T_4 = [d, e] \]
\[ T_5 = [e, f] \]
\[ T_6 = [f, g] \]
\[ T_7 = [g, h] \]
\[ T_8 = [h, i] \]

Segments sorted by y-coordinates

Prefix | Violation
---|---
\(T_1\) | report \((a, b)\)
\(T_2\) | report \((b, d)\)
\(T_3\) | report \((g, h)\)
\(T_8\) | report \((h, i)\)

OpenDRC Software Architecture

- **Interface Layer**
  - GDSII Parser
  - Rule Def.
  - Result IO

- **Application Layer**
  - Task Scheduler
  - Algo Dispatcher

- **Algorithm Layer**
  - Area
  - Width
  - Spacing
  - Enclosure

- **Infrastructure Layer**
  - Database
  - Geo.
  - Utility
  - GPU Libs
auto db = odrc::gdsii::read(/* path-to-gdsii */);
auto e = odrc::engine();
e.add_rules({
    db.polygons().is_rectilinear(),
    db.layer(19).width().greater_than(18),
    db.layer(20).polygons().ensures(
        [](const auto& p){return !p.name.empty();}
    )
});
e.check(db);
### Experimental Evaluation (Intra-Polygon Rules)

<table>
<thead>
<tr>
<th>Design</th>
<th>Rule</th>
<th>KLayout flat</th>
<th>KLayout deep</th>
<th>KLayout tile</th>
<th>X-Check Seq.</th>
<th>X-Check Par.</th>
<th>OpenDRC Seq.</th>
<th>OpenDRC Par.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>aes</td>
<td>M1.W1</td>
<td>3.45</td>
<td>12.69</td>
<td>0.49</td>
<td>0.41</td>
<td>0.02</td>
<td>0.03</td>
<td>M1.A1</td>
</tr>
<tr>
<td></td>
<td>M2.W1</td>
<td>1.37</td>
<td>3.83</td>
<td>0.23</td>
<td>0.14</td>
<td>0.04</td>
<td>0.04</td>
<td>M2.A1</td>
</tr>
<tr>
<td></td>
<td>M3.W1</td>
<td>2.52</td>
<td>2.98</td>
<td>0.36</td>
<td>0.11</td>
<td>0.03</td>
<td>0.03</td>
<td>M3.A1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ethmac</td>
<td>M1.W1</td>
<td>11.88</td>
<td>45.84</td>
<td>1.56</td>
<td>1.21</td>
<td>0.07</td>
<td>0.08</td>
<td>M1.A1</td>
</tr>
<tr>
<td></td>
<td>M2.W1</td>
<td>3.76</td>
<td>10.72</td>
<td>0.52</td>
<td>0.42</td>
<td>0.10</td>
<td>0.11</td>
<td>M2.A1</td>
</tr>
<tr>
<td></td>
<td>M3.W1</td>
<td>6.36</td>
<td>7.64</td>
<td>0.36</td>
<td>0.31</td>
<td>0.08</td>
<td>0.08</td>
<td>M3.A1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>gcd</td>
<td>M1.W1</td>
<td>0.13</td>
<td>0.44</td>
<td>0.13</td>
<td>0.11</td>
<td>&lt; 0.01</td>
<td>&lt; 0.01</td>
<td>M1.A1</td>
</tr>
<tr>
<td></td>
<td>M2.W1</td>
<td>0.05</td>
<td>0.08</td>
<td>0.05</td>
<td>&lt; 0.01</td>
<td>&lt; 0.01</td>
<td>&lt; 0.01</td>
<td>M2.A1</td>
</tr>
<tr>
<td></td>
<td>M3.W1</td>
<td>0.06</td>
<td>0.07</td>
<td>0.06</td>
<td>&lt; 0.01</td>
<td>&lt; 0.01</td>
<td>&lt; 0.01</td>
<td>M3.A1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ibex</td>
<td>M1.W1</td>
<td>3.60</td>
<td>12.38</td>
<td>0.50</td>
<td>0.43</td>
<td>0.02</td>
<td>0.03</td>
<td>M1.A1</td>
</tr>
<tr>
<td></td>
<td>M2.W1</td>
<td>1.30</td>
<td>3.61</td>
<td>0.24</td>
<td>0.14</td>
<td>0.03</td>
<td>0.04</td>
<td>M2.A1</td>
</tr>
<tr>
<td></td>
<td>M3.W1</td>
<td>2.38</td>
<td>2.88</td>
<td>0.36</td>
<td>0.10</td>
<td>0.03</td>
<td>0.03</td>
<td>M3.A1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>jpeg</td>
<td>M1.W1</td>
<td>13.32</td>
<td>55.35</td>
<td>1.68</td>
<td>1.39</td>
<td>0.08</td>
<td>0.08</td>
<td>M1.A1</td>
</tr>
<tr>
<td></td>
<td>M2.W1</td>
<td>3.05</td>
<td>8.77</td>
<td>0.46</td>
<td>0.40</td>
<td>0.10</td>
<td>0.10</td>
<td>M2.A1</td>
</tr>
<tr>
<td></td>
<td>M3.W1</td>
<td>4.86</td>
<td>6.14</td>
<td>0.59</td>
<td>0.29</td>
<td>0.08</td>
<td>0.08</td>
<td>M3.A1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>sha3</td>
<td>M1.W1</td>
<td>3.48</td>
<td>12.36</td>
<td>0.49</td>
<td>0.43</td>
<td>0.02</td>
<td>0.03</td>
<td>M1.A1</td>
</tr>
<tr>
<td></td>
<td>M2.W1</td>
<td>1.10</td>
<td>2.95</td>
<td>0.21</td>
<td>0.12</td>
<td>0.03</td>
<td>0.03</td>
<td>M2.A1</td>
</tr>
<tr>
<td></td>
<td>M3.W1</td>
<td>1.79</td>
<td>2.15</td>
<td>0.30</td>
<td>0.09</td>
<td>0.02</td>
<td>0.02</td>
<td>M3.A1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>uart</td>
<td>M1.W1</td>
<td>0.15</td>
<td>0.40</td>
<td>0.15</td>
<td>0.11</td>
<td>&lt; 0.01</td>
<td>&lt; 0.01</td>
<td>M1.A1</td>
</tr>
<tr>
<td></td>
<td>M2.W1</td>
<td>0.06</td>
<td>0.12</td>
<td>0.06</td>
<td>&lt; 0.01</td>
<td>&lt; 0.01</td>
<td>&lt; 0.01</td>
<td>M2.A1</td>
</tr>
<tr>
<td></td>
<td>M3.W1</td>
<td>0.08</td>
<td>0.09</td>
<td>0.08</td>
<td>&lt; 0.01</td>
<td>&lt; 0.01</td>
<td>&lt; 0.01</td>
<td>M3.A1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Average</td>
<td></td>
<td>37.7×</td>
<td>82.1×</td>
<td>9.6×</td>
<td>4.5×</td>
<td>0.9×</td>
<td>1.0×</td>
<td>37.6×</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>13.0×</td>
</tr>
</tbody>
</table>

Average values are provided for comparison.
## Experimental Evaluation (Inter-Polygon Rules)

<table>
<thead>
<tr>
<th>Design</th>
<th>Rule</th>
<th>KLayout</th>
<th>X-Check</th>
<th>OpenDRC</th>
<th>Rule</th>
<th>KLayout</th>
<th>X-Check</th>
<th>OpenDRC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>flat</td>
<td>deep</td>
<td>tile</td>
<td></td>
<td>flat</td>
<td>deep</td>
<td>tile</td>
</tr>
<tr>
<td>aes</td>
<td>M1.S.1</td>
<td>4.33</td>
<td>13.78</td>
<td>0.62</td>
<td>0.17</td>
<td>0.21</td>
<td>0.06</td>
<td>V1.M1.EN.1</td>
</tr>
<tr>
<td></td>
<td>M2.S.1</td>
<td>1.55</td>
<td>4.15</td>
<td>0.29</td>
<td>0.13</td>
<td>0.09</td>
<td>0.02</td>
<td>V2.M2.EN.1</td>
</tr>
<tr>
<td></td>
<td>M3.S.1</td>
<td>2.64</td>
<td>3.25</td>
<td>0.38</td>
<td>0.12</td>
<td>0.15</td>
<td>0.02</td>
<td>V1.M2.EN.2</td>
</tr>
<tr>
<td>ethmac</td>
<td>M1.S.1</td>
<td>14.67</td>
<td>48.50</td>
<td>1.89</td>
<td>0.39</td>
<td>0.72</td>
<td>0.14</td>
<td>V1.M1.EN.1</td>
</tr>
<tr>
<td></td>
<td>M2.S.1</td>
<td>4.35</td>
<td>11.71</td>
<td>0.59</td>
<td>0.20</td>
<td>0.23</td>
<td>0.05</td>
<td>V2.M2.EN.1</td>
</tr>
<tr>
<td></td>
<td>M3.S.1</td>
<td>6.68</td>
<td>8.17</td>
<td>0.82</td>
<td>0.16</td>
<td>0.39</td>
<td>0.04</td>
<td>V1.M2.EN.2</td>
</tr>
<tr>
<td></td>
<td>M1.S.1</td>
<td>0.15</td>
<td>0.46</td>
<td>0.14</td>
<td>0.11</td>
<td>&lt; 0.01</td>
<td>0.01</td>
<td>V1.M1.EN.1</td>
</tr>
<tr>
<td>gcd</td>
<td>M2.S.1</td>
<td>0.05</td>
<td>0.09</td>
<td>0.05</td>
<td>0.11</td>
<td>&lt; 0.01</td>
<td>&lt; 0.01</td>
<td>V2.M2.EN.1</td>
</tr>
<tr>
<td></td>
<td>M3.S.1</td>
<td>0.06</td>
<td>0.07</td>
<td>0.06</td>
<td>0.11</td>
<td>&lt; 0.01</td>
<td>&lt; 0.01</td>
<td>V1.M2.EN.2</td>
</tr>
<tr>
<td>ibex</td>
<td>M1.S.1</td>
<td>4.45</td>
<td>13.15</td>
<td>0.63</td>
<td>0.17</td>
<td>0.22</td>
<td>0.06</td>
<td>V1.M1.EN.1</td>
</tr>
<tr>
<td>jpeg</td>
<td>M2.S.1</td>
<td>1.49</td>
<td>3.96</td>
<td>0.25</td>
<td>0.13</td>
<td>0.09</td>
<td>0.02</td>
<td>V2.M2.EN.1</td>
</tr>
<tr>
<td></td>
<td>M3.S.1</td>
<td>2.50</td>
<td>3.08</td>
<td>0.39</td>
<td>0.12</td>
<td>0.14</td>
<td>0.02</td>
<td>V1.M2.EN.2</td>
</tr>
<tr>
<td>sha3</td>
<td>M1.S.1</td>
<td>15.82</td>
<td>57.36</td>
<td>2.01</td>
<td>0.43</td>
<td>0.80</td>
<td>0.16</td>
<td>V1.M1.EN.1</td>
</tr>
<tr>
<td></td>
<td>M2.S.1</td>
<td>3.48</td>
<td>9.79</td>
<td>0.49</td>
<td>0.21</td>
<td>0.20</td>
<td>0.05</td>
<td>V2.M2.EN.1</td>
</tr>
<tr>
<td></td>
<td>M3.S.1</td>
<td>5.17</td>
<td>6.70</td>
<td>0.64</td>
<td>0.16</td>
<td>0.30</td>
<td>0.03</td>
<td>V1.M2.EN.2</td>
</tr>
<tr>
<td>uart</td>
<td>M1.S.1</td>
<td>4.23</td>
<td>13.02</td>
<td>0.60</td>
<td>0.16</td>
<td>0.21</td>
<td>0.06</td>
<td>V1.M1.EN.1</td>
</tr>
<tr>
<td></td>
<td>M2.S.1</td>
<td>1.16</td>
<td>3.23</td>
<td>0.22</td>
<td>0.12</td>
<td>0.07</td>
<td>0.02</td>
<td>V2.M2.EN.1</td>
</tr>
<tr>
<td></td>
<td>M3.S.1</td>
<td>1.87</td>
<td>2.31</td>
<td>0.30</td>
<td>0.11</td>
<td>0.11</td>
<td>0.02</td>
<td>V1.M2.EN.2</td>
</tr>
<tr>
<td></td>
<td>M1.S.1</td>
<td>0.19</td>
<td>0.44</td>
<td>0.19</td>
<td>0.11</td>
<td>&lt; 0.01</td>
<td>0.01</td>
<td>V1.M1.EN.1</td>
</tr>
<tr>
<td></td>
<td>M2.S.1</td>
<td>0.07</td>
<td>0.13</td>
<td>0.07</td>
<td>0.11</td>
<td>&lt; 0.01</td>
<td>&lt; 0.01</td>
<td>V2.M2.EN.1</td>
</tr>
<tr>
<td></td>
<td>M3.S.1</td>
<td>0.08</td>
<td>0.10</td>
<td>0.08</td>
<td>0.10</td>
<td>&lt; 0.01</td>
<td>&lt; 0.01</td>
<td>V1.M2.EN.2</td>
</tr>
<tr>
<td>Average</td>
<td></td>
<td>47.6×</td>
<td>99.5×</td>
<td>12.0×</td>
<td>5.6×</td>
<td>3.2×</td>
<td>1.0×</td>
<td></td>
</tr>
</tbody>
</table>
The runtime breakdown of OpenDRC sequential *minimum spacing* checks. ‘Layout Part’ refers to adaptive layout partitioning; ‘Intvl. Tree Ops.’ refers to interval tree operations *insert*, *remove*, and *query*; ‘E2E Check’ refers to edge-to-edge checks.
We develop **OpenDRC**, a new open-source design rule checking engine

- adaptive row-based layout partition
- efficient sequential/parallel hierarchical DRC procedures
- significant speedup compared with SOTA multi-threading/GPU design rule checkers

**Future work:**

- systematic evaluation of heterogeneous computing in DRC
- data compression techniques for memory footprint reduction
- supports for general geometric shapes
THANK YOU!