Heterogeneous Acceleration for Design Rule Checking

Zhuolun He, Bei Yu

Department of Computer Science & Engineering
The Chinese University of Hong Kong

Nov. 01, 2023
Outline

1. Introduction
2. Efficient DRC
3. Parallel Sweepline for DRC
4. Heterogeneous DRC
CPU-GPU Paradigm is Promising

37,888 GPUs (8,335,360 cores) + 9,472 CPUs (606,208 cores) ⇒ one ExaFLOPS ($10^{18}$)
GPU-accelerated EDA

• Placement
  • Global Placement: DreamPlace [DAC’19], X-Place [DAC’22]
  • Detailed Placement: ABCDPlace [TCAD’20]

• Routing
  • pattern routing: FastGR [TCAD’22]
  • maze routing: GAMER [TCAD’22]
  • Steiner tree construction [ICCAD’22]

• Static Timing Analysis [ICCAD’20] [DAC’21]

• gate-level logic simulation [DAC’22], circuit simulation [DAC’21], logic optimization [DAC’22], capacitance extraction [DATE’13]

• ...


This Talk: Design Rule Checking

- 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
Algebraic Design Rule Checking

Example: if type1 and type3 are ‘outside’, type2 is ‘inside’, it represents width check
Layout Data Structures

Binning

Corner Stitching [TCAD’84]
Parallel DRC

- Various parallelism
  - region-based, hierarchy-based, edge-based, ···
  - task parallelism, data + task

- Various platforms
  - SIMD, multiprocessors, GPU, specialized hardware, ···

<table>
<thead>
<tr>
<th></th>
<th>Multiprocessor</th>
<th>GPU</th>
<th>Hardware</th>
<th>Distributed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Region-Data-</td>
<td></td>
<td></td>
<td></td>
<td>[VLSID’94]</td>
</tr>
<tr>
<td>Hierarchy-</td>
<td>[ICPP’84]</td>
<td>[ICCCAD’22]</td>
<td>[DAC’84]</td>
<td>[DAC’11]</td>
</tr>
<tr>
<td>Edge-</td>
<td>[DAC’88]</td>
<td>[DAC’23]</td>
<td>[VLSID’20]</td>
<td>[CS’92]</td>
</tr>
<tr>
<td>Task-</td>
<td></td>
<td></td>
<td></td>
<td>[TR’86]</td>
</tr>
<tr>
<td>Task- and Data-</td>
<td>[JPDC’96]</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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
(We only consider horizontal edges.)

**Problem (Distance Check)**

Given a set $\mathcal{H}$ of horizontal segments in $\mathbb{R}^2$, report the segment pairs from $\mathcal{H}^2$ whose horizontal projection is nonempty, and vertical distance is smaller than $\delta$. Formally, we want to report:

$$\{(l_1, r_1] \times y_1, [l_2, r_2] \times y_2) \in \mathcal{H}^2\}$$

s.t. $[l_1, r_1] \cap [l_2, r_2] \neq \emptyset, |y_1 - y_2| < \delta$
Parallel prefix sums

\[ a[] = (4, 5, 3, 6, 2, 5, 1, 1, 0) \]

Suppose we have 3 threads.

1. Batching: each thread computes sums of 3 consecutive elements.
\[ s = (?, ?, 12, ?, ?, 13, ?, ?, 2) \]

2. Sweeping: sweep the partial sums
\[ s = (?, ?, 12, ?, ?, 25, ?, ?, 27) \]

3. Refining: compute other prefix sums
\[ s = (4, 9, 12, 18, 20, 25, 26, 27, 27) \]
Key idea: the prefix structure contains a set $S$ of segments that are below current segment within $\delta$ in $y$-direction.

Remains to check if each pair of segments overlap in the $x$-direction.

Segments sorted by $y$-coordinates:

<table>
<thead>
<tr>
<th>Prefix</th>
<th>Violation</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_1 = [a, a]$</td>
<td>report $(a, b)$</td>
</tr>
<tr>
<td>$T_2 = [a, c]$</td>
<td></td>
</tr>
<tr>
<td>$T_3 = [b, d]$</td>
<td>report $(b, d)$</td>
</tr>
<tr>
<td>$T_4 = [d, e]$</td>
<td></td>
</tr>
<tr>
<td>$T_5 = [e, f]$</td>
<td>report $(g, h)$</td>
</tr>
<tr>
<td>$T_6 = [f, g]$</td>
<td></td>
</tr>
<tr>
<td>$T_7 = [g, h]$</td>
<td>report $(h, i)$</td>
</tr>
<tr>
<td>$T_8 = [h, i]$</td>
<td></td>
</tr>
</tbody>
</table>

Vertical Sweeping
General strategies:

- Concurrent GPU computation and CPU computation.
- Concurrent GPU computation between streams.
- Overlap data transfer and computation.
- Minimize data transfer overhead.
- Avoid GPU invocation for small data batch.
Techniques in OpenDRC

Background: In OpenDRC, layout is adaptively partitioned into rows.

- **S1**: GPU computation of the previous row and CPU preprocessing of the next row can be executed concurrently.
- **S2**: GPU computation for horizontal edges and vertical edges can be executed concurrently by different streams.
- **S3**: Data movement for one batch of data and GPU sorting of another can be overlapped.
Techniques in OpenDRC

Background: In OpenDRC, layout is adaptively partitioned into rows.

- S1: GPU computation of the previous row and CPU preprocessing of the next row can be executed concurrently.
- S2: GPU computation for horizontal edges and vertical edges can be executed concurrently by different streams.
- S3: Data movement for one batch of data and GPU sorting of another can be overlapped.
- S4: Differentiate horizontal and vertical edges.
- S5: No GPU computation will be invoked if a row has only a limited number of objects.
## Result

<table>
<thead>
<tr>
<th>Design</th>
<th>Size</th>
<th>Rows</th>
<th>CPU</th>
<th>GPU</th>
<th>GPU\S1</th>
<th>GPU\S2</th>
<th>GPU\S3</th>
<th>GPU\S4</th>
<th>GPU\S5</th>
<th>GPU\all</th>
</tr>
</thead>
<tbody>
<tr>
<td>aes</td>
<td>294052</td>
<td>277</td>
<td>2128</td>
<td>189</td>
<td>187</td>
<td>193</td>
<td>213</td>
<td>192</td>
<td>186</td>
<td>202</td>
</tr>
<tr>
<td>ethmac</td>
<td>1007152</td>
<td>507</td>
<td>14318</td>
<td>436</td>
<td>441</td>
<td>455</td>
<td>430</td>
<td>520</td>
<td>440</td>
<td>534</td>
</tr>
<tr>
<td>ibex</td>
<td>303004</td>
<td>277</td>
<td>2404</td>
<td>187</td>
<td>194</td>
<td>197</td>
<td>194</td>
<td>202</td>
<td>193</td>
<td>212</td>
</tr>
<tr>
<td>jpeg</td>
<td>1182541</td>
<td>537</td>
<td>19692</td>
<td>455</td>
<td>465</td>
<td>480</td>
<td>458</td>
<td>526</td>
<td>503</td>
<td>575</td>
</tr>
<tr>
<td>sha3</td>
<td>301382</td>
<td>277</td>
<td>2298</td>
<td>180</td>
<td>178</td>
<td>192</td>
<td>177</td>
<td>185</td>
<td>191</td>
<td>217</td>
</tr>
<tr>
<td>Average</td>
<td>19.22</td>
<td>1.00</td>
<td>1.01</td>
<td>1.05</td>
<td>1.03</td>
<td>1.09</td>
<td>1.04</td>
<td>1.18</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Removing each strategy: 1% to 9% speed-down
- Still $16.3 \times$ faster than CPU when removing all five
- Heterogeneous acceleration is promising
- Review efforts for efficient DRC
  - Algebraic
  - Layout data structures
  - parallel DRC
- Heterogeneous acceleration for DRC
THANK YOU!