Using Open-Source EDA Tools in an Industrial Design Flow

Daniela Sánchez Lopera, Prajwal Kashyap, Nicolas Gerlin, Sven Wenzek, Wolfgang Ecker
Outline

• Motivation
• Background – Digital design flow
• What is OpenROAD?
• Our design flow
• Use Cases
• Results
• Summary & Conclusion
Motivation: Open-Source Software for EDA?

Advantages:
• Extensibility
• Accessibility
• Scalability

For industries even more:
• Playground for:
  • Students, researchers & inhouse trainings
• Suitable for:
  • Experiments
  • Collecting huge amount of data
  • Analysing design and flow
• Enabling:
  • Machine Learning (ML) applications
  • Innovation
Background
Digital Design Flow

Images generated by OpenROAD using random die configuration and open-source PDK
What is OpenROAD?

- RTL-to-GDSII framework for design exploration and physical design implementation
- Three existing "flow controllers"
  - OpenROAD-flow-scripts¹
  - OpenLANE²
  - Robust Design Flow (RDF)-2021³

¹ https://github.com/The-OpenROAD-Project/OpenROAD-flow-scripts
² https://github.com/The-OpenROAD-Project/OpenLane
³ https://github.com/ieee-ceda-datc/datc-rdf
What is OpenROAD?

Related work

ML applications optimizing tool configurations:
- LSOracle\textsuperscript{4}
- OpenABC-D\textsuperscript{5}
- VeriGOOD-ML\textsuperscript{6}

ML applications learning from OpenROAD outcomes:
- Congestion\textsuperscript{7}
- Arrival times and slack\textsuperscript{8}

\textsuperscript{7} Ghose, Amur, et al. "Generalizable Cross-Graph Embedding for GNN-based Congestion Prediction", ICCAD 2021
What is OpenROAD?
Challenges
Lück, C., Sánchez Lopera, D., Wenzek, S., & Ecker, W. Industrial Experience with Open-Source EDA Tools. MLCAD 2022

Adaptations on source code to cope with:
• Infrastructure restrictions: No super user, no Docker
• Proprietary PDKs
• Parallelization on compute farm
What is OpenROAD?

Envisioned use-cases

- Design Space Exploration
- Data generation for ML models
  - Design metric prediction

Sánchez Lopera D., Ecker W., Applying GNNs to Timing Estimation at RTL, ICCAD 2022.

MetaRTL: Ecker, W., and Schreiner, J. "Introducing Model-of-Things (MoT) and Model-of-Design (MoD) for simpler and more efficient hardware generators. VLSI-SoC 2016."
What is OpenROAD?

Envisioned use-cases

• Design Space Exploration

• Data generation for ML models
  • Design metric prediction

Sánchez Lopera D., Ecker W., Applying GNNs to Timing Estimation at RTL, ICCAD 2022.

MetaRTL: Ecker, W., and Schreiner, J. "Introducing Model-of-Things (MoT) and Model-of-Design (MoD) for simpler and more efficient hardware generators. VLSI-SoC 2016.

What is OpenROAD?

Envisioned use-cases

- Design Space Exploration
- Data generation for ML models:
  - Design metric prediction

Lück, C., Sánchez Lopera, D., Wenzek, S., & Ecker, W. Industrial Experience with Open-Source EDA Tools. MLCAD 2022

How good are our ground truth labels coming from open-source tools?

Sánchez Lopera D., Ecker W., Applying GNNs to Timing Estimation at RTL, ICCAD 2022.

**MetaRTL**: Ecker, W., and Schreiner, J.. "Introducing Model-of-Things (MoT) and Model-of-Design (MoD) for simpler and more efficient hardware generators. VLSI-SoC 2016."
## OpenROAD vs Commercial Tools

<table>
<thead>
<tr>
<th>Feature</th>
<th>Open-source</th>
<th>Commercial</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extensibility</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>Accessibility</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>Scalability</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>Customer Support</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Reliability</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>Technology &amp; Engineering</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>Workforce development</td>
<td>x</td>
<td>x</td>
</tr>
</tbody>
</table>

A Mixed Open-Source and Proprietary EDA Commons for Education and Prototyping Invited Paper. Andrew B. Kahng. ICCAD 2022
OpenROAD vs Commercial Tools

<table>
<thead>
<tr>
<th>Feature</th>
<th>Open-source</th>
<th>Commercial</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extensibility</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>Accessibility</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>Scalability</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>Customer Support</td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>Reliability</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>Technology &amp; Engineering</td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>Workforce development</td>
<td>x</td>
<td>x</td>
</tr>
</tbody>
</table>

- **Open and free software**
- **Years of experience and billions of investments**

A Mixed Open-Source and Proprietary EDA Commons for Education and Prototyping Invited Paper. Andrew B. Kahng. ICCAD 2022
## OpenROAD vs Commercial Tools

<table>
<thead>
<tr>
<th>Feature</th>
<th>Open-source</th>
<th>Commercial</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extensibility</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>Accessibility</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>Reliability</td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>Technology &amp; Engineering</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>Workforce development</td>
<td>x</td>
<td>x</td>
</tr>
</tbody>
</table>

Q1: But how do they compare w.r.t PPA results?

Open and free software

A Mixed Open-Source and Proprietary EDA Commons for Education and Prototyping Invited Paper. Andrew B. Kahng. ICCAD 2022
Our Design Flow (1)

Inputs

- Specifications
- Constraints (.sdc)
- Commercial PDK
- Open-Source PDK
Our Design Flow (1)

Inputs

<table>
<thead>
<tr>
<th>PDK</th>
<th>Type</th>
<th># Lines Lib. File</th>
<th># Standard Cells</th>
</tr>
</thead>
<tbody>
<tr>
<td>40nm</td>
<td>Proprietary</td>
<td>14678.9 K</td>
<td>852</td>
</tr>
<tr>
<td>130nm</td>
<td>Open-source</td>
<td>333.5 K</td>
<td>753</td>
</tr>
</tbody>
</table>

Our Design Flow (2)
RTL Generation

Advantages of using MetaRTL on top of OpenROAD:
- Using one programming language for hardware generation, synthesis flow and machine learning$^{12}$
- Generation of properties for formal verification$^{13}$

$^{12}$ K. Devarajegowda, et al., "Python based framework for HDSLs with an underlying formal semantics“. ICCAD 2017
$^{13}$ K. Devarajegowda, et al., "How to Keep 4-Eyes Principle in a Design and Property Generation Flow“, MBMV 2019
Our Design Flow (3)
RTL2GDS Flow

Specifications

MetaRTL

RISC-V

Constraints (.sdc)

Commercial PDK

Open-source PDK

Logic Synthesis → Floorplan & Powerplan → Placement → Clock Tree Synthesis → Routing → Finish

Design Flow

Static Timing Analysis

Parasitic Extraction
Our Design Flow (4)

RTL2GDS Tools

- Specifications
- Constraints (.sdc)
- Commercial PDK
- Open-source PDK

Design Flow:
- Logic Synthesis
- Floorplan & Powerplan
- Placement
- Clock Tree Synthesis
- Routing
- Finish

Static Timing Analysis
Parasitic Extraction

Commercial Tools:
- Yosys
- ABC
- TritonFP
- ioPlacer
- PDKGen
- RePLAce
- OpenDP
- Resizer
- TritonCTS 2.0
- TritonRoute
- FastRoute
- Antenna Checker
- KLayout

OpenROAD:
- OpenSTA
- OpenRCX
## Generated Use-Cases
### RISC-V - RV32IMCX

<table>
<thead>
<tr>
<th>Designs</th>
<th>Extension Units</th>
<th># Lines of Code</th>
<th># Components</th>
<th># Input bits</th>
<th># Output bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>RISC-V¹</td>
<td>CRC, PFC</td>
<td>16496</td>
<td>810</td>
<td>71</td>
<td>157</td>
</tr>
<tr>
<td>RISC-V²</td>
<td>Exception</td>
<td>28377</td>
<td>1430</td>
<td>170</td>
<td>164</td>
</tr>
<tr>
<td>RISC-V³</td>
<td>MAC</td>
<td>39487</td>
<td>2271</td>
<td>171</td>
<td>164</td>
</tr>
<tr>
<td>RISC-V⁴</td>
<td>Event Counters</td>
<td>16391</td>
<td>844</td>
<td>70</td>
<td>157</td>
</tr>
<tr>
<td>RISC-V⁵</td>
<td>CRC, PFC, MAC, Event Counters, Exception</td>
<td>42121</td>
<td>2403</td>
<td>170</td>
<td>165</td>
</tr>
</tbody>
</table>
# Generated Use-Cases

## RISC-V

<table>
<thead>
<tr>
<th>Designs</th>
<th>Extension Units</th>
<th># Lines of Code</th>
<th># Components</th>
<th># Input bits</th>
<th># Output bits</th>
<th>Complexity Flag</th>
</tr>
</thead>
<tbody>
<tr>
<td>RISC-V(^1)</td>
<td>CRC, PFC</td>
<td>16496</td>
<td>810</td>
<td>71</td>
<td>157</td>
<td>+</td>
</tr>
<tr>
<td>RISC-V(^2)</td>
<td>Exception</td>
<td>28377</td>
<td>1430</td>
<td>170</td>
<td>164</td>
<td>++</td>
</tr>
<tr>
<td>RISC-V(^3)</td>
<td>MAC</td>
<td>39487</td>
<td>2271</td>
<td>171</td>
<td>164</td>
<td>++</td>
</tr>
<tr>
<td>RISC-V(^4)</td>
<td>Event Counters</td>
<td>16391</td>
<td>844</td>
<td>70</td>
<td>157</td>
<td>+</td>
</tr>
<tr>
<td>RISC-V(^5)</td>
<td>CRC, PFC, MAC, Event Counters, Exception</td>
<td>42121</td>
<td>2403</td>
<td>170</td>
<td>165</td>
<td>+++</td>
</tr>
</tbody>
</table>
Post Routing Results
Results (1)
Post Routing – Area

\[
Avg.\ ratio = \frac{1}{5} \sum_{Commercial\ Tool}^{Yosys/OpenROAD} RISC-V
\]
Averaging the results for all 5 RISC-Vs, OpenROAD occupies more area:

- NAND2 Eq. Area: 2.1x more area
- # Standard Cells: 2.4x more cells
Results (2)
Post Routing – Worst Slack

Averaging the results for all 5 RISC-Vs, OpenROAD worst slack after routing is:
• Critical path worst slack: 2.9x higher
Results (3)

Post Routing – Total Power

Commercial Tool

OpenROAD

Averaging the results for all 5 RISC-Vs, OpenROAD consumes more power:
• Total power: 2.7x more
Runtimes
Results (4)
Wall times – Routing

• Under fair conditions:
  • No multi-threading
  • Same CPU:
    Linux CPU Intel® Xeon® Gold 6248R
    at 3.00 GHz and 80 GiB system memory
Sweeping clock
Results (5)
Sweeping clock - Logic Synthesis - Area

OpenROAD keeps area constant while varying clock period
Results (6)
Sweeping clock - Logic Synthesis - Power

Higher clock, less total power
OpenROAD needs very high clock periods for meeting timing requirements after logic synthesis.
Results (8)
Sweeping clock – Routing - Worst Slack

OpenROAD needs higher clocks for meeting timing requirements after routing
Q1: But how do they compare w.r.t PPA results?
## Summary

<table>
<thead>
<tr>
<th>Stage</th>
<th>Ratio [Open-Source Tool/Commercial Tool]</th>
<th>NAND2 Eq. Area</th>
<th># Standard Cells</th>
<th>WS</th>
<th>Total Power</th>
</tr>
</thead>
<tbody>
<tr>
<td>Post logic synthesis</td>
<td></td>
<td>1.5</td>
<td>1.8</td>
<td>&lt;0</td>
<td>2.4</td>
</tr>
<tr>
<td>Post routing</td>
<td></td>
<td>2.1</td>
<td>2.4</td>
<td>2.9</td>
<td>2.7</td>
</tr>
</tbody>
</table>

Avg = 2.5x

* For PDK 40nm
* For flatten synthesis
* For clock 25ns
## Summary

<table>
<thead>
<tr>
<th>Stage</th>
<th>Ratio [Open-Source Tool/Commercial Tool]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>NAND2 Eq.</td>
</tr>
<tr>
<td>Post logic synthesis</td>
<td>1.49</td>
</tr>
<tr>
<td>Post routing</td>
<td>2.14</td>
</tr>
</tbody>
</table>

* For PDK 40nm
* For flatten synthesis
* For clock 25ns

How good are our ground truth labels coming from open-source tools?
Conclusion

• We outline our industrial flow from initial specifications to GDS using different RISC-Vs as use cases.

• Averaging the reported post-routing PPA factors for a 25 ns clock period, the commercial tool outperforms OpenROAD by a factor of 2.52x.

• The commercial tool is faster without any parallelization, and it meets timing constraints for lower clock periods than OpenROAD.
Conclusion

**But** OpenROAD is evolving fast: more than 1.4K commits in 2022 and 19 active pull requests\(^\text{14}\).

*Future:*

- Commercial and open-source EDA working together to enable research and progress on the field
- Analyze generated output of commercial tools using OpenROAD and vice versa


Questions?

Thank you!
What is OpenROAD?

Related work

OpenROAD published papers describe some attempts of benchmarking:

• Number of commits, citations, community engagement\textsuperscript{9}
• Comparing results of their AutoTuner with two different SkyWater libraries\textsuperscript{10}
• Comparing the OpenROAD placer, OpenSTA and OpenRCX w.r.t commercial tools\textsuperscript{11}

\textsuperscript{10}A. B. Kahng, "Looking into the Mirror of Open Source: Invited Paper", ICCAD 2019
\textsuperscript{11}A. B. Kahng and T. Spyrou, “The OpenROAD Project: Unleashing Hardware Innovation”, GOMAC 2021
What is OpenROAD?
Envisioned use-cases

• *Design Space Exploration (DSE)*

Lück, C., Sánchez Lopera, D., Wenzek, S., & Ecker, W. Industrial Experience with Open-Source EDA Tools. MLCAD 2022
Post Logic Synthesis Results
Results (1)
Post Logic Synthesis – Area

NAND2 Equivalent Area

![Bar chart showing NAND2 Equivalent Area for different RISC-V versions and Yosys/Comm. Tool configurations.](chart.png)
Results (1)
Post Logic Synthesis – Area

NAND2 Equivalent Area

Averaging the results for all 5 RISC-Vs, Yosys/OpenROAD occupies higher area:
• For 130nm: 1.2x more area
• For 40nm: 1.5x more area
Results (2)
Post Logic Synthesis – Area

Total Standard Cells

Averaging the results for all 5 RISC-Vs, Yosys/OpenROAD occupies higher area:
• For 130nm: 1.5x more cells
• For 40nm: 1.8x more cells
Results (3)

Post Logic Synthesis – Worst Negative Slack (WNS)

For both PDKs: 130nm and 40nm

<table>
<thead>
<tr>
<th>Synthesis Type</th>
<th>Commercial Tool</th>
<th>OpenROAD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flatten</td>
<td>0</td>
<td>&lt;0</td>
</tr>
<tr>
<td>Non-flatten</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>(hierarchical)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Results (3)
Post Logic Synthesis – Worst Negative Slack (WNS)

For both PDKs: 130nm and 40nm

<table>
<thead>
<tr>
<th>Synthesis Type</th>
<th>Commercial Tool</th>
<th>OpenROAD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flatten</td>
<td>0</td>
<td>&lt;0</td>
</tr>
<tr>
<td>Non-flatten (hierarchical)</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

![Graph showing Worst Negative Slack for RISC-V designs in 130nm and 40nm PDKs]
Results (4)
Post Logic Synthesis – Total Power

Commercial Tool

OpenROAD
Results (4)

Post Logic Synthesis – Total Power

Averaging the results for all 5 RISC-Vs, Yosys/OpenROAD consumes more power:

- For 130nm: 3.8x more power
- For 40nm: 2.4x more power
Results (4)

Post Logic Synthesis – Total Power

Averaging the results for all 5 RISC-Vs, Yosys/OpenROAD consumes more power:

- For 130nm: 3.8x more power
- For 40nm: 2.4x more power