2022 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE
MUNICH, GERMANY
DECEMBER 6 - 7, 2022

Using Open-Source EDA Tools in an Industrial Design Flow

Daniela Sánchez Lopera*†, Sven Wenzek§, Wolfgang Ecker*†
*Infineon Technologies AG
†Technische Universität München, Germany
‡Technische Universität Kaiserslautern, Germany
§EPOS Embedded Core & Power Systems
Email: Firstname.Lastname@infineon.com
Email: Firstname.Lastname@epos-d.com

***Abstract*— Commonly, semiconductor manufacturers employ commercial electronic design automation tools to get production-ready results. These tools are costly and closed source. Thus, they are not suited for academic and research purposes. Seminal works have presented open-source alternatives for stages of the design flow. However, executing standalone tools is time-consuming, and iterations of the design flow are costly. To overcome this, an open-source toolchain for digital layout is presented: OpenROAD. Even though OpenROAD has been described extensively in the literature, its performance w.r.t commercial tools has not been evaluated. To address this, this paper generates different implementations of RISC-V designs using an in-house hardware generation framework. Those designs are synthesized using OpenROAD and a commercial tool and mapped to a commercial and open-source technology library. Experiments compare optimization capabilities, runtimes, and the Quality of Results (QoRs) along the design flow. Commercial tools provide higher QoRs, lower runtimes, and better optimization capabilities. However, OpenROAD represents a good trade-off between performance, license costs, framework transparency, and support to the academic community. The comparison of OpenROAD and commercial tools is relevant to analyze the validity of machine learning models trained with OpenROAD outcomes, and to promote the use of open-source tools.**

Keywords*—*Electronic Design Automation*, *OpenROAD*, *Quality of Results*, *Machine Learning

I. Introduction

Typically, the chip design flow from initial specification to Graphic Data System (GDS) is conducted using commercial Electronic Design Automation (EDA) tools to obtain high Quality of Results (QoRs) and production-ready layouts. However, commercial tool licenses are expensive. Thus, they are not appropriate for academic and research purposes, which need many and long tool runs. Even though commercial tool vendors provide support to customers, their tools are black boxes, and innovation in the design flow itself is constrained. To overcome these limitations, seminal works have introduced EDA open-source tools, such as ABC [1] and KLayout [2].

The digital design flow applies the "Divide and Conquer" strategy, going from specifications to layouts. Nevertheless, using standalone tools is time-consuming, and design flow iterations are costly. To overcome these issues, a DARPA IDEA program presented in 2018 the OpenROAD[1] project [3]: an open-source toolchain for digital layout aiming to run 24 hours with no human-in-the-loop. OpenROAD interconnects standalone open-source tools to build the design flow from Register Transfer Level (RTL) to GDS. Figure 1 summarizes the stages of the design flow and their respective open-source tool alternatives.

OpenROAD popularity is growing and is being used inside flow controllers such as OpenROAD-flow-scripts[2], OpenLane[3] [4], and the Robust Design Flow (RDF)-2021[4] [5]. These flow controllers use OpenROAD as the main building block and connect it with other tools to add optimizations, improve design exploration or automate data collection. The OpenROAD flow controllers bring many benefits: adjustable source code, easy installation, and open exchange with the community. For research and academia, OpenROAD contribution is even higher. Innovation is made through OpenROAD code extension and modification, but also through Machine Learning (ML) models learning from OpenROAD-generated objects and reports [6]–[9].

Even though the number of ML applications using OpenROAD outcomes continues to increase, to the best of our knowledge, no studies compare the performance of OpenROAD w.r.t the commercial tools. To solve this, this paper compares OpenROAD with a commercial alternative within the industrial design flow setup shown in Figure 1. The main contributions of this paper are:

- Different RISC-V implementations are generated using an in-house RTL generator.
- A proprietary and an open-source Process Design Kit (PDK) are used.
- Experiments compare the QoRs along the design flow, runtimes, and optimization capabilities.

[1]https://github.com/The-OpenROAD-Project/OpenROAD
[2]https://github.com/The-OpenROAD-Project/OpenROAD-flow-scripts
[3]https://github.com/The-OpenROAD-Project/OpenLane
[4]https://github.com/ieee-ceda-datc/datc-rdf

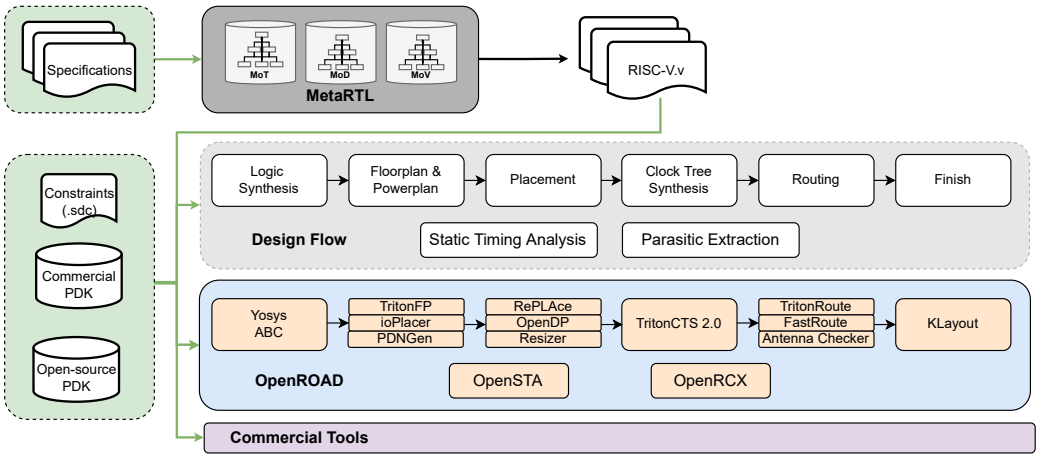


Figure 1: Industrial Design Flow from Specifications to GDSII

Results confirm that the QoRs of commercial tools is higher. However, OpenROAD represents a good trade-off between QoRs, costs, and framework transparency. OpenROAD is an enabler of research in EDA to foster new algorithms in the design flow and to collect ML training data with distributions not so far from the obtained with commercial tools.

This paper is organized as follows: Section II briefly reviews the digital design flow. Section III surveys related works using OpenROAD. Section IV describes the industrial design flow used. The different implementations of the RISC-V are presented in Section V. The synthesis results using commercial and open-source tools are shown in Section VI. Finally, Section VII concludes by summarizing the results and mentioning future directions.

II. Digital Design Flow

This section briefly reviews the stages of the chip design flow shown in the gray box of Figure 1.

Having the RTL description of a design in any of the Hardware Description Languages (HDLs), the first step in the design flow is **logic synthesis** to map the RTL blocks to cell gates selected from a technology library. Given the timing constraints, the tool performs logic mapping while optimizing Power, Performance, and Area (PPA). During **floor planning**, the gate-level netlist is mapped to an initial physical description of the chip. To that end, the floor plan size, core and die area, utilization ratio, and type of power distribution are defined. Then, the major RTL blocks (macros), input/output ports, and the power grid are placed [10]. In **placement**, standard cells are located in specific regions of the layout while minimizing PPA targets and cost metrics such as congestion and wire length.

Previous stages considered ideal clock distributions. **Clock Tree Synthesis (CTS)** is routing the clock signal from its source to all the sequential elements while minimizing skew and meeting delay targets [10]. Given the metal layers of the PDK, the nets are horizontally and vertically routed until all the components are connected while minimizing area and wire length. **Routing** is a complex task due to the high number of cells, the geometrical constraints, and the electrical issues, which can increase the delays or induce noise signals [10]. Afterward, sign-off checks are performed: **Design Rule Check (DRC)** determines if the generated layout satisfies design rules such as metal width and spacing for different layers. **Layout Versus Schematic (LVS)** verifies whether the functionality of the gate-level netlist and the layout is the same. And, **Parasitic Extraction** extracts resistances, capacitances, and inductances to simulate and compare final and pre-layout results. Finally, intermediate files are merged to generate the final GDS layout [11].

III. Related Work

In the following, we mention state-of-the-art works already using OpenROAD.

As OpenROAD reduces design efforts and allows a better design-space exploration, the number of studies using this toolchain is increasing. For instance, in [12], the future research directions on placement are linked to the availability of open-source tools and PDKs, such as OpenROAD and the SkyWater Technology Foundry [13]. OpenROAD is also being used by the OpenLane flow and the RDF-2021. OpenLane is an open-source RTL-to-Graphic Data System II (GDSII) flow developed by Efabless. It adds custom tools for better design space exploration. Thus, it is a valuable tool for academic projects, such as developing a RISC-V 32-bit processor [14]. The RDF-2021 flow developed by the IEEE CEDA Design Automation Technical Committee (DATC) [15] is a framework for EDA-flow-related research. The latest RDF version includes Chisel/FIRRTL [16], a hardware generation tool built in Scala, and METRICS2.1 [17], a metric system to collect data and enable ML applications.

OpenROAD opens the door to ML applications optimizing tool configurations and learning from tool outcomes [6]–[9], [18]–[20]. LSOracle [18] has been added as a plugin to Yosys to optimize logic mapping.

Table I: Brief comparison of used PDKs

PDK	License Type	PVT			# Lines Lib. File [$\times 10^3$]	# Standard Cells
130 nm	open-source	1.0 P	1.80 V	25°C	333.5	753
40 nm	proprietary	1.3 P	1.08 V	-40°C	14678.9	852

In [6], a dataset called OpenABC-D is built to predict netlists QoRs for given tool parameters. OpenABC-D is leveraged by the Bulls-Eye [19] framework, which predicts the quality of synthesized netlist and includes simulated annealing to search the design space. Similarly, VeriGOOD-ML [20] generates Verilog code and validates it using OpenROAD outcomes. OpenROAD is used to build datasets for predicting design metrics as congestion [7], arrival times and slack [8], and PPA of Coarse-grained Reconfigurable Arrays (CGRAs) [9].

OpenROAD published papers describe the advances of the open-source flow and some attempts of benchmarking [17], [21], [22]. In [17], the AutoTuner framework finding optimal OpenROAD parameters is evaluated over three designs and two different SkyWater libraries. In [21], the number of commits and citations are proposed as metrics to overcome the challenge of open-source tools' progress quantification. A first comparison was conducted between a commercial and the OpenROAD placer. Similar to [21], [22] discusses design successes from OpenROAD and community engagement. They compare the capacitance and resistance values provided by OpenRCX and OpenSTA with results from commercial parasitic extraction and timing analysis tools. Even though OpenROAD has been described and used extensively in the literature, to the best of our knowledge, no studies compare its performance w.r.t commercial tools.

IV. Industrial Design Flow from Specifications to GDS

This section describes the industrial setup shown in Figure 1.

A. Specification-to-RTL Flow

As recognized in [5], the flow controllers using OpenROAD highly benefit from hardware generation frameworks, which generate different designs and micro-architectures. In the RDF-2021, Chisel/FIRRTL is incorporated. This paper employs an in-house generation framework called *MetaRTL* [23]. MetaRTL uses metamodeling and applies the Model Driven Architecture (MDA) principle to RTL generation [24]. In general, this framework provides three layers of a design: the initial specifications or Model of Things (MoT), the intermediate representation or Model of Design (MoD), and finally, the mapping to an HDL or Model of View (MoV). When integrated into OpenROAD, MetaRTL benefits are twofold: Python is the programming language used for hardware generation, OpenROAD flow controllers, and ML applications. Second, formal properties are generated, and it is possible to verify MetaRTL designs following a four-eyes principle [25]. The formal verification capabilities of our in-house framework motivate even further a complete flow from specification to GDS. However, formal verification is beyond the scope of this paper.

B. RTL-to-GDS Flow

This paper compares a commercial tool and the flow controller OpenROAD-flow-scripts. Figure 1 shows the open-source tools within OpenROAD. Yosys [26] and ABC perform logic synthesis and are built standalone. Yosys is an open-source Verilog synthesis tool that provides the opportunity to add extensions and customizations. Nevertheless, it does not include a timing analysis engine. Yosys uses ABC [1] for logic minimization and technology mapping. The resulting gate-level netlist is the input to the OpenROAD toolchain. While commercial tools are already installed, parallelized, and ready to use within an industrial flow, OpenROAD and its flow controller are set within our multi-user compute farm [27].

C. Process Design Kits (PDKs)

A PDK is a set of data files describing the fabrication process of a chip for the design tools. A PDK, developed by a foundry, contains technology details, design rules, standard cell libraries, layout information, and SPICE models [28]. This paper uses two PDKs: the open-source SkyWater 130 nm and a proprietary 40 nm. Table I compares relevant aspects of both technologies. Commercial tools convert the open-source PDK to a format compatible with the commercial synthesis tool.

V. Use Case: RISC-V

Using MetaRISC [25], five RISC-V implementations are generated using different extensions, as listed in Table II. Each design consists of a 32-bit 5-stage pipeline without privileged mode with support for compressed instructions and multiplication without division (RV32IMCX).

Table II: Features of the generated and synthesized RISC-V designs

Design	Extension Units	# LoC	# Components	# Input Bits	# Output Bits
RISC-V[1]	CRC, PFC	16496	810	71	157
RISC-V[2]	Exception	28377	1430	170	164
RISC-V[3]	MAC	39487	2271	171	164
RISC-V[4]	Event Counters	16391	844	70	157
RISC-V[5]	CRC, PFC, MAC, Event Counters, Exception	42121	2403	170	165

The RISC-V[1] uses an on-chip control flow integrity unit called Program Flow Check (PFC). It computes a signature with Cyclic Redundancy Check (CRC) by taking the current instruction and the last signature as input for each clock cycle. The software can compare a pre-computed signature with this run-time one to ensure that the control flow is not corrupted, following the principle described in [29]. The RISC-V[2] adds an exception unit to the CPU to handle exceptions and interrupts as defined in the RISC-V privileged specifications. New signals for exception detection are added all over the pipeline and forwarded to this unit responsible for the trap handling. The RISC-V[3] incorporates a Multiply-Accumulate (MAC) unit, widely used in signal processing and machine learning applications. RISC-V[4] includes Event Counters for hardware monitoring, i.e., a cycle and an instruction counter have been added. A 32-bit Control and Status Register (CSR) has been added to activate all the possible counters, and two 32-bit CSRs represent each counter. RISC-V[5] includes all the features mentioned above.

VI. Synthesizing RISC-V Designs

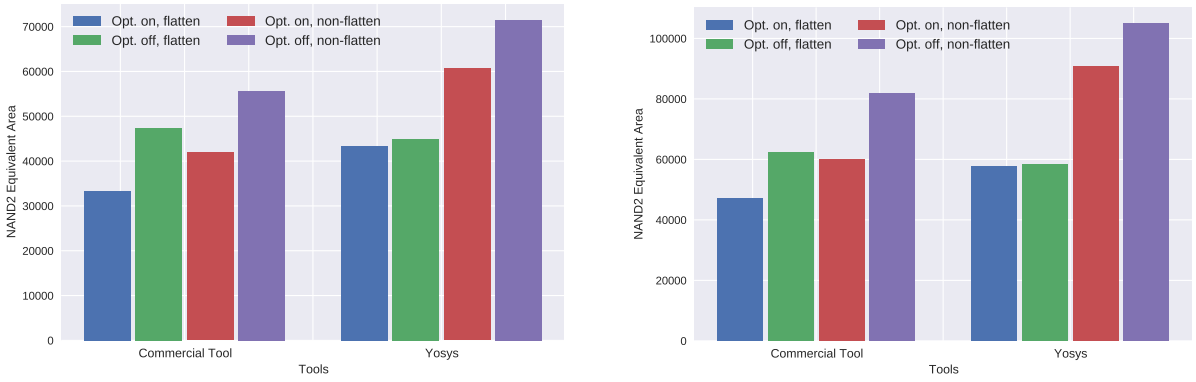
The design flow in Figure 1 has been applied to generate and synthesize the RISC-V implementation listed in Table II. In the following, results over the different circuits are presented. The clock period is fixed to 25 ns, the maximum latency is 1 ns, and the minimum is 0.1 ns. The setup skew is maintained at 0.5 ns, the setup transition at 0.2 ns, and the hold transition at 0.1 ns.

A. Logic Synthesis

For logic synthesis, we compare flattened and hierarchical synthesis. Moreover, Yosys offers two configurations: *ABC_AREA* and *ABC_SPEED*, for reducing area and delay, respectively. In the case of the commercial tool, synthesis is run without and with optimizations.

1) Comparing Area

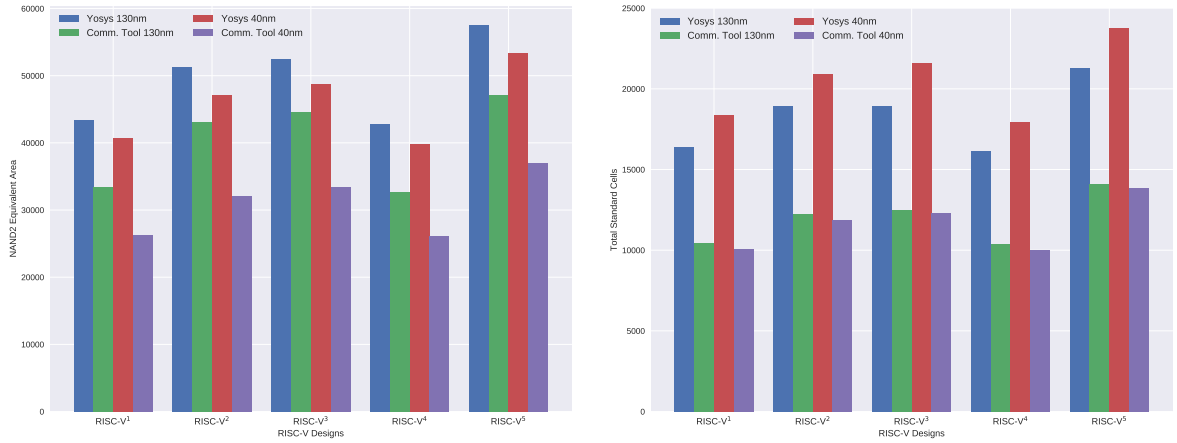
To compare the total area among two PDKs, we employ the NAND2 equivalent area, using the area of the smallest NAND cell of each technology with two inputs and one output. Figure 2 shows how the different configurations influence the results for the RISC-V[1] and the RISC-V[5] designs. For simplicity, Figure 2 shows results when using the open-source PDK. Similar results are obtained using the 40 nm technology: The lowest area is captured when enabling optimizations (ABC_SPEED target in Yosys) for flattened synthesis. For non-flattened designs, the commercial tool outperforms Yosys with and without optimizations.



(a) RISC-V[1] (b) RISC-V[5]

Figure 2: NAND2 equivalent area after logic synthesis with different configurations

All designs are synthesized using the best configurations, i.e., flattened synthesis and enabled optimizations. Figure 3a shows that the total area when using Yosys is, on average, 1.24× and 1.49× higher than the commercial tool for the 130 nm and 40 nm technologies, respectively. Figure 3b shows that the commercial tool uses lesser number of standard cells independently of the technology. On average, Yosys needs 1.54× more standard cells for the 130 nm technology and 1.77× for the 40 nm technology w.r.t the commercial tool.



(a) NAND2 equivalent area (b) Total standard cells

Figure 3: NAND2 equivalent area and total standard cells after logic synthesis with flattened and optimized setup

2) Comparing WNS

The Worst Negative Slack (WNS) and the Total Negative Slack (TNS) reported after logic synthesis by the commercial tool are zero for all the cases, indicating that the logic mapping is timing-driven. The reported WNS by OpenSTA after Yosys is also zero, but only when performing non-flattened synthesis. For flattened synthesis, the WNS is negative, as shown in Figure 4, for all RISC-Vs and both PDKs. The reported slack worsens with the complexity of the designs.

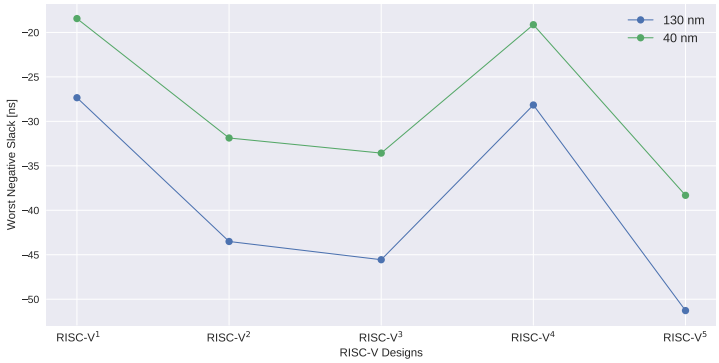
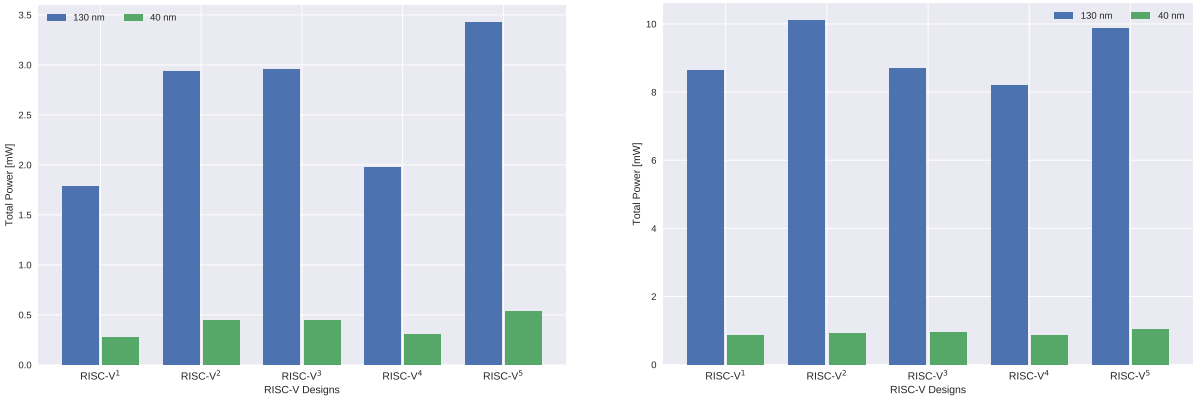


Figure 4: WNS per design after flattened logic synthesis using Yosys

3) Comparing Total Power

The reported total power after using the commercial tool is, on average, 3.81× and 2.39× lower than the value reported by OpenROAD for the 130 nm and 40 nm technology, respectively. Best results are obtained using flattened synthesis, ABC_AREA, and enabled optimizations. Figure 5 shows that the maximum power consumed is around 3.5 mW for the commercial tool and approximately 10.1 mW for Yosys when using the 130 nm technology.



(a) Commercial tool (b) Yosys

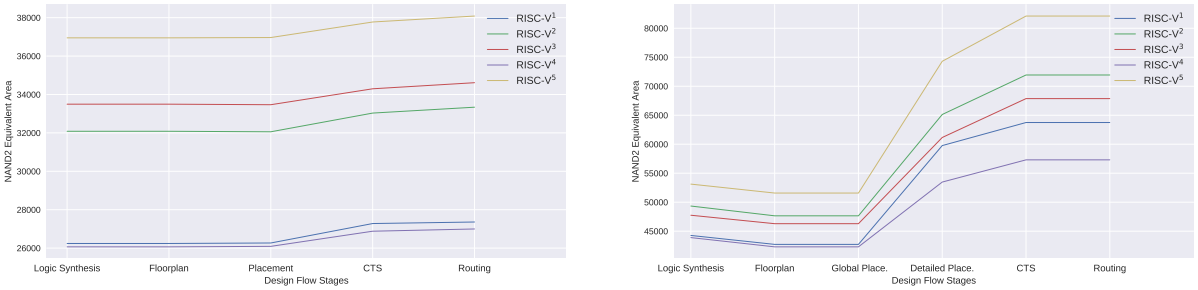
Figure 5: Total power after logic synthesis using the best configurations for both tools

B. Physical Synthesis

The gate-level netlist obtained after logic synthesis is the input to a commercial tool and OpenROAD. We compare their results for the 40 nm technology and all RISC-Vs designs. For simplicity, the best configurations are used, i.e., flattened synthesis, ABC_AREA, and enabled optimizations.

1) Comparing Area

Figure 6 shows the total NAND2 equivalent area per stage. Results show that the commercial tool results after logic synthesis are already optimized for floor planning and placement so that the total area after those stages does not increase. On average, the resulting netlist from the commercial tool occupies a lower area by a factor of $2.14\times$ and a lesser number of standard cells by a factor of $2.41\times$ w.r.t the OpenROAD netlist.

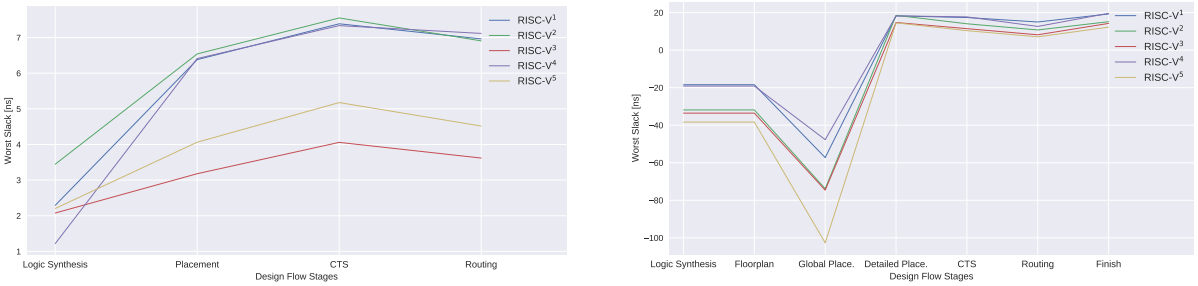


(a) Commercial tool (b) OpenROAD

Figure 6: Post-route NAND2 equivalent area per each stage using both tools

2) Comparing WNS

After routing, the WNS and TNS are zero for both tools when running flattened synthesis with enabled optimizations. Figure 7 shows the critical path positive slack during different stages. On average, for all designs, the post-routing critical path in OpenROAD has a slack $2.87\times$ higher than the reported by the commercial tool.

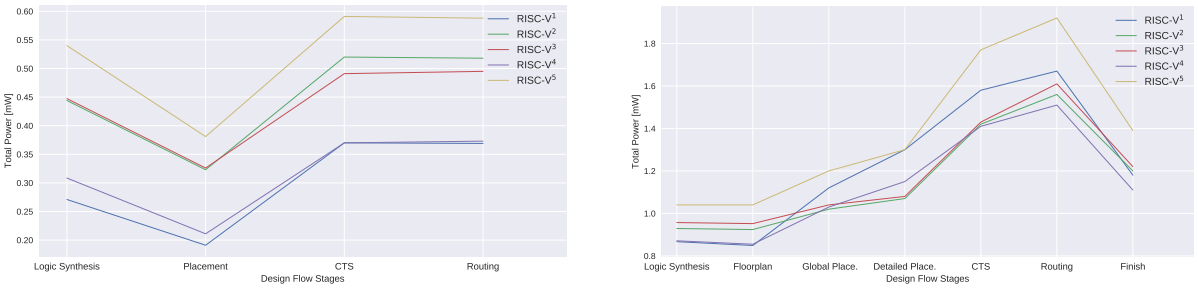


(a) Commercial tool (b) OpenROAD

Figure 7: Post-route critical path slack per each stage using both tools

3) Comparing Total Power

Figure 8 shows the variation of the total power consumption along the design flow. OpenROAD post-routing layout consumes more power by a factor of $2.66\times$ w.r.t the reported values from the commercial tool.



(a) Commercial Tool (b) OpenROAD

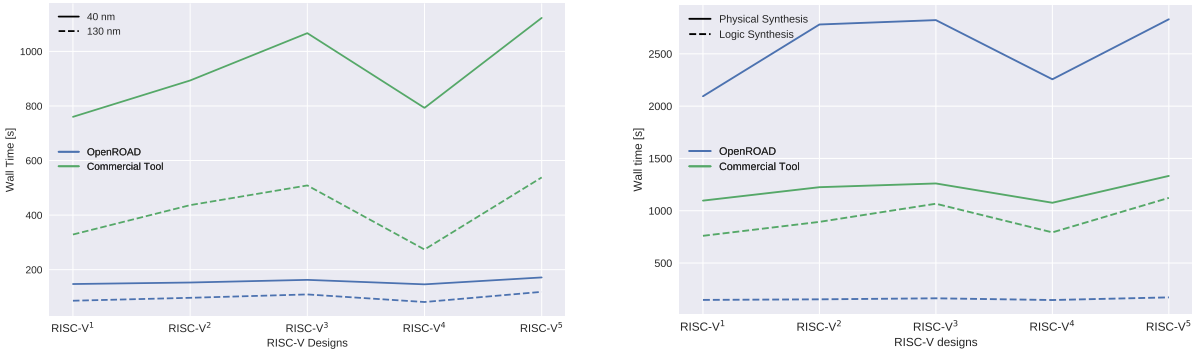
Figure 8: Post-route total power consumption per each stage using both tools

C. Runtimes

For our experiments, both tools are sequentially run on a Linux CPU Intel® Xeon® Gold 6248R at 3.00 GHz and 80 GiB system memory. For fairness in the comparison, the multi-threading options are disabled for both tools. In other cases, the commercial tool is already wrapped up and sped up so that each stage is executed in

a different machine of the compute farm, based on memory and speed requirements. OpenROAD also provides the option to use multiple cores to improve runtime.

Figure 9 shows the wall times for logic and physical synthesis. On average, Yosys (OpenROAD) is $5.91\times$ and $4.18\times$ faster than the commercial tool for the 130 nm and 40 nm technologies. The runtime during logic synthesis increases with the complexity of the technology used, as shown in Figure 9a. Figure 9b shows the wall time for floor planning until routing. On average, the physical synthesis stages consume $2.13\times$ more time in OpenROAD than in the commercial tool. In general, the commercial tool performs logic and physical synthesis faster.

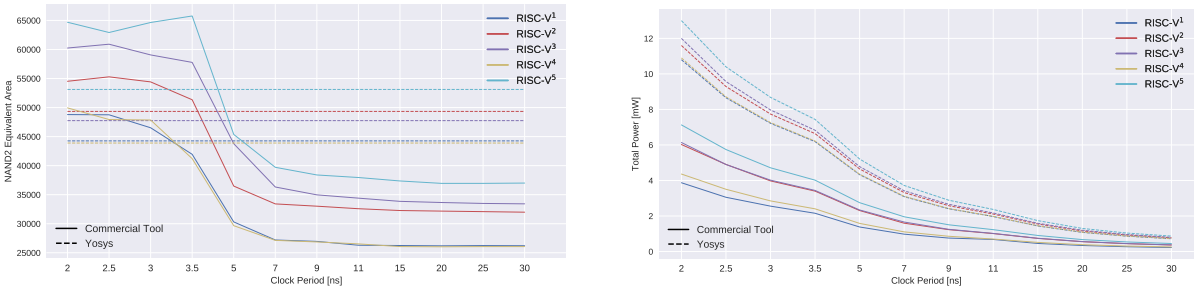


(a) Logic synthesis (b) Routing

Figure 9: Wall time after logic synthesis and routing

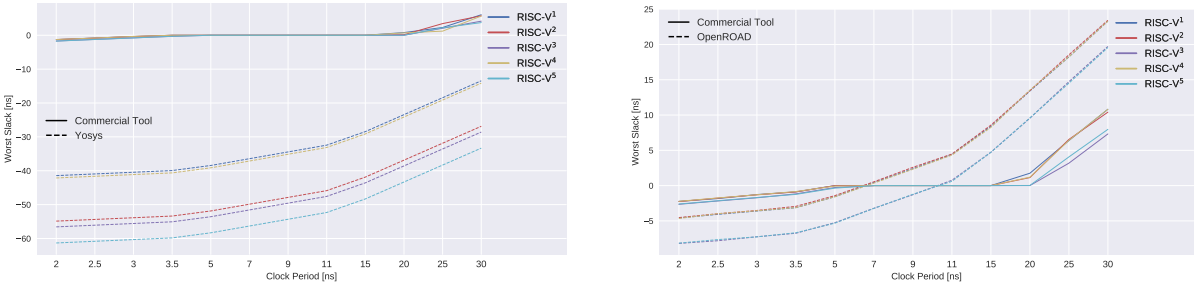
D. PPA Analysis for Different Clocks

The above experiments fixed the clock to 25 ns to avoid post-routing timing violations. Figure 10 and Figure 11a show the PPA results after logic synthesis when sweeping the clock period. In the commercial tool, the total area is reduced when increasing the clock period until a threshold. Yosys does not consider timing constraints, and its total area across different clock periods stays constant. The commercial tool achieves a total power lower than 2 mW from a clock period equal to 7 ns. For OpenROAD, this happens for clock periods higher than 15 ns. Using a 3.5 ns clock period, the commercial tool reaches a zero worst slack for all RISC-Vs. This is never reached after running Yosys. Results after detailed placement show similar trends w.r.t area and power. Regarding the worst slack, the netlists have already been repaired, and no timing violations appear for a clock period higher than 10 ns with OpenROAD and 5 ns with the commercial tool, as shown in Figure 11.



(a) NAND2 equivalent area (b) Total power

Figure 10: Total power and area after logic synthesis with different clock periods



(a) Logic synthesis (b) Routing

Figure 11: Worst slack after logic synthesis and routing with different clock periods

VII. SUMMARY

We outline our industrial flow from initial specifications to GDS using different RISC-Vs as use cases. We compare OpenROAD performance with a commercial EDA tool. Averaging the reported post-routing factors comparing PPA results for a 25 ns clock period, the commercial tool outperforms OpenROAD by a factor of 2.5. The commercial tool is, on average, 588 s faster without any parallelization, and it could meet the timing constraints for lower clock periods than OpenROAD. We expect this to change in the future based on the rapid development of OpenROAD and its community commitment manifested as more than 140 citations to OpenROAD publications [30], 1490 commits in 2022, and 19 active pull requests [31]. In contrast to commercial tools, open-source EDA tools open the door to academic research, benchmarking, ML techniques, and the transfer of research to real-world scenarios. In future work, we expect to include LSOracle and AutoTuner in our analysis. Moreover, we plan to use the outcomes of OpenROAD inside commercial tools and vice versa. As mentioned in [32], OpenROAD flow controllers still do not ease portability among OpenROAD and commercial tools. We expect this to change, as this would allow industries and researchers to benefit from both worlds.

ACKNOWLEDGMENT

The work described herein is partly funded by the German Federal Ministry for Economic Affairs and Energy (BMWi) as part of the research project ProgressivKI (19A21006C).

REFERENCES

[1] R. Brayton and A. Mishchenko, "ABC: An academic industrial-strength verification tool," in *Computer Aided Verification*, 2010, pp. 24–40.
[2] M. Koefferlein, "KLayout layout viewer and editor," 2018. [Online]. Available: https://klayout.de/
[3] T. Ajayi, D. Blaauw, T. Chan, C. Cheng, V. Chhabria *et al.*, "OpenROAD: Toward a self-driving, open-source digital layout implementation tool chain," *Proceedings. GOMACTECH*, pp. 1105–1110, 2019.
[4] A. A. Ghazy and M. Shalan, "OpenLANE: The open-source digital ASIC implementation flow," 2020.
[5] J. Chen, I. H.-R. Jiang, J. Jung, A. B. Kahng, S. Kim *et al.*, "DATC RDF-2021: Design flow and beyond ICCAD special session paper," in *2021 IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, 2021, pp. 1–6.
[6] A. B. Chowdhury, B. Tan, R. Karri, and S. Garg, "OpenABC-D: A large-scale dataset for machine learning guided integrated circuit synthesis," 2021.
[7] A. Ghose, V. Zhang, Y. Zhang, D. Li, W. Liu, and M. Coates, "Generalizable cross-graph embedding for GNN-based congestion prediction," in *2021 IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, 2021, pp. 1–9.
[8] Z. Guo, M. Liu, J. Gu, S. Zhang, D. Z. Pan, and Y. Lin, "A timing engine inspired graph neural network model for pre-routing slack prediction," in *Proceedings of the 59th ACM/IEEE Design Automation Conference (DAC)*. ACM, 2022, p. 1207–1212.
[9] C. Tan, C. Xie, A. Li, K. J. Barker, and A. Tumeo, "AURORA: Automated refinement of coarse-grained reconfigurable accelerators," in *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2021, pp. 1388–1393.
[10] A. Teman, "Digital VLSI design." [Online]. Available: https://www.eng.biu.ac.il/temanad/digital-vlsi-design/
[11] "Signoff-checks." [Online]. Available: https://signoffsemiconductors.com/signoff-checks/
[12] A. B. Kahng, "Advancing placement," in *Proceedings of the 2021 International Symposium on Physical Design (ISPD)*. ACM, 2021, p. 15–22.
[13] "Welcome to SkyWater SKY130 PDK's Documentation!" [Online]. Available: https://skywater-pdk.readthedocs.io/en/main/
[14] "RISC-V chip designed with open source tools eeNews Europe," May 2022. [Online]. Available: https://www.eenewseurope.com/en/risc-v-chip-designed-with-open-source-tools/
[15] "Design Automation Technical Committee | IEEE Council on Electronic Design Automation." [Online]. Available: https://ieee-ceda.org/technical-committee/datc
[16] J. Bachrach, H. Vo, B. Richards, Y. Lee, A. Waterman *et al.*, "Chisel: Constructing hardware in a Scala embedded language," in *Proceedings of the 49th Annual Design Automation Conference (DAC)*. ACM, 2012, p. 1216–1225.
[17] J. Jung, A. B. Kahng, S. Kim, and R. Varadarajan, "METRICS2.1 and flow tuning in the IEEE CEDA Robust Design Flow and OpenROAD ICCAD special session paper," in *2021 IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, 2021, pp. 1–9.
[18] W. L. Neto, M. Austin, S. Temple, L. G. Amarù, X. Tang, and P.-E. Gaillardon, "LSOracle: A logic synthesis framework driven by artificial intelligence: Invited paper," *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 1–6, 2019.
[19] A. Chowdhury, B. Tan, R. Carey, T. Jain, R. Karri *et al.*, "Too big to fail? Active few-shot learning guided logic synthesis," 2022.
[20] H. Esmaeilzadeh, S. Ghodrati, J. Gu, S. Guo, A. B. Kahng *et al.*, "VeriGOOD-ML: An open-source flow for automated ML hardware synthesis," in *2021 IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, 2021, pp. 1–7.
[21] A. B. Kahng, "Looking into the mirror of open source: Invited paper," in *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2019, pp. 1–8.
[22] A. B. Kahng and T. Spyrou, "The OpenROAD project: Unleashing hardware innovation," in *Proc. GOMAC*, 2021.
[23] J. Schreiner, R. Findenigy, and W. Ecker, "Design centric modeling of digital hardware," in *2016 IEEE International High-Level Design Validation and Test Workshop (HLDVT)*, 2016, pp. 46–52.
[24] F. Truyen, "The fast guide to model driven architecture," *Cephas Consulting Corp*, 2006.
[25] K. Devarajegowda, W. Ecker, and W. Kunz, "How to keep 4-eyes principle in a design and property generation flow," in *MBMV 2019; 22nd Workshop - Methods and Description Languages for Modelling and Verification of Circuits and Systems*, 2019, pp. 1–6.
[26] C. Wolf, J. Glaser, and J. Kepler, "Yosys- A Free Verilog Synthesis Suite," 2013.
[27] C. Lück, D. Sánchez Lopera, S. Wenzek, and W. Ecker, "Industrial experience with open-source EDA tools," in *Proceedings of the 2022 ACM/IEEE Workshop on Machine Learning for CAD*, 2022, pp. 143–143.
[28] "A guide to advanced process design kits." [Online]. Available: https://semiengineering.com/a-guide-to-advanced-process-design-kits/
[29] A. Benso, S. Di Carlo, G. Di Natale, and P. Prinetto, "A watchdog processor to detect data and control flow errors," in *9th IEEE On-Line Testing Symposium (IOLTS)*. IEEE, 2003, pp. 144–148.
[30] "Publications." [Online]. Available: https://theopenroadproject.org/publications/
[31] "Commit activity." [Online]. Available: https://github.com/The-OpenROAD-Project/OpenROAD/graphs/commit-activity
[32] H. Liew, D. Grubb, J. Wright, C. Schmidt, N. Krzysztofowicz *et al.*, "Hammer: A modular and reusable physical design flow tool: Invited," in *Proceedings of the 59th ACM/IEEE Design Automation Conference (DAC)*. ACM, 2022, p. 1335–1338.