INVITED: Toward an Open-Source Digital Flow: First Learnings from the OpenROAD Project

Tutu Ajayi[1], Vidya A. Chhabria[2], Mateus Fogaça[3], Soheil Hashemi[4], Abdelrahman Hosny[4], Andrew B. Kahng[5], Minsoo Kim[5], Jeongsup Lee[1], Uday Mallappa[5], Marina Neseem[4], Geraldo Pradipta[2], Sherief Reda[4], Mehdi Saligane[1], Sachin S. Sapatnekar[2], Carl Sechen[6], Mohamed Shalan[7], William Swartz[6], Lutong Wang[5], Zhehong Wang[1], Mingyu Woo[5] and Bangqi Xu[5]

[1]U. Michigan, [2]U. Minnesota, [3]PGMicro/UFRGS, [4]Brown U., [5]UC San Diego, [6]U. Texas-Dallas, [7]American U. in Cairo

ABSTRACT

We describe the planned Alpha release of OpenROAD, an open-source end-to-end silicon compiler. OpenROAD will help realize the goal of "democratization of hardware design", by reducing cost, expertise, schedule and risk barriers that confront system designers today. The development of open-source, self-driving design tools is in and of itself a "moon shot" with numerous technical and cultural challenges. The open-source flow incorporates a compatible open-source set of tools that span logic synthesis, floorplanning, placement, clock tree synthesis, global routing and detailed routing. The flow also incorporates analysis and support tools for static timing analysis, parasitic extraction, power integrity analysis, and cloud deployment. We also note several observed challenges, or "lessons learned", with respect to development of open-source EDA tools and flows.

ACM Reference Format: Tutu Ajayi, Vidya A. Chhabria, Mateus Fogaça, Soheil Hashemi, Abdelrahman Hosny, Andrew B. Kahng, Minsoo Kim, Jeongsup Lee, Uday Mallappa, Marina Neseem, Geraldo Pradipta, Sherief Reda, Mehdi Saligane, Sachin S. Sapatnekar, Carl Sechen, Mohamed Shalan, William Swartz, Lutong Wang, Zhehong Wang, Mingyu Woo and Bangqi Xu. INVITED: Toward an Open-Source Digital Flow: First Learnings from the OpenROAD Project. In *The 56th Annual Design Automation Conference 2019 (DAC '19), June 2–6, 2019, Las Vegas, USA.* ACM, New York, NY, USA, 4 pages. https://doi.org/110.1145/3316781.3326334

1 INTRODUCTION

Even as hardware design tools and methodologies have advanced over the past decades, the semiconductor industry has failed to control product design costs. Today, barriers of cost, expertise and unpredictability (risk) block designers' access to hardware implementation in advanced technologies. Put another way: hardware system innovation is stuck in a local minimum of (i) complex and expensive tools, (ii) a shortage of expert users capable of using these tools in advanced technologies, and (iii) enormous cost and risk barriers to even attempting hardware design.

DAC '19, June 2–6, 2019, Las Vegas, NV, USA

ACM ISBN 978-1-4503-6725-7/19/06…$15.00
https://doi.org/10.1145/3316781.3326334

Particularly in the digital integrated-circuit (IC) domain, layout automation has been integral to the design of huge, extremely complex products in advanced technology nodes. However, a shortfall of *design capability* – i.e., the ability to scale product quality concomitant with the scaling of underlying device and patterning technologies – has been apparent for over a decade in even the most advanced companies [4]. Thus, to meet product and schedule requirements, today's leading-edge system-on-chip (SoC) product companies must leverage specialization and divide-and-conquer across large teams of designers: each individual block of the design is handled by a separate subteam, and each designer has expertise in a specific facet of the design flow. Many development teams do not have resources to execute such a strategy, and hence see typical hardware design cycles of 12-36 months.

To overcome the above limitations and keep pace with the exponential increases in SoC complexity associated with Moore's Law, the DARPA IDEA program aims to develop a fully automated "no human in the loop" circuit layout generator that enables users with no electronic design expertise to complete physical design of electronic hardware. The **OpenROAD** ("Foundations and Realization of Open, Accessible Design") project [19] was launched in June 2018 as part of the DARPA IDEA program. OpenROAD's overarching goal is to bring down the barriers of cost, expertise and unpredictability that currently block system creators' access to hardware implementation in advanced technologies. OpenROAD seeks to develop a **fully autonomous**, **open-source** tool chain for silicon compilation across die, package and board, with initial focus on the RTL-to-GDSII phase of system-on-chip design. More specifically, we aim to deliver tapeout-capable tools in source code form, with permissive licensing, so as to seed a future "Linux of EDA" (i.e., *electronic design automation*).

The contributions and approach of OpenROAD seek to establish a new paradigm for EDA tools, academic-industry collaboration, and academic research itself. OpenROAD aims to finally surmount ingrained, "cultural" and "critical mass / critical quality" barriers to establishing an open-source ethos in the EDA field. The remainder of this paper will describe briefly the main tools in the OpenROAD's Alpha flow release as well as lessons learned from our endeavors.

2 MAIN TOOLS FOR LAYOUT GENERATION

OpenROAD's silicon compilation tool chain consists of a set of open-source tools that takes RTL Verilog, constraints (.sdc), liberty (.lib) and technology (.lef) files as input, and aims to generate a tapeout-ready GDSII file. Figure 1 illustrates the flow of tools corresponding

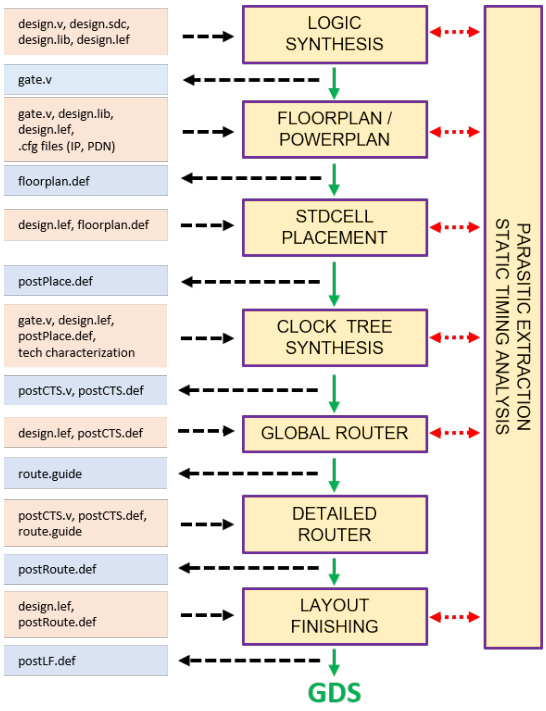

Figure 1: The OpenROAD flow.

to individual OpenROAD tasks. These include logic synthesis (LS), floorplan (FP) and power delivery network (PDN), placement, clock tree synthesis (CTS), routing and layout finishing.

A. Logic Synthesis: The major gap in open-source LS is timing awareness and optimization. Our Alpha release brings three improvements to the open-source YOSYS/ABC synthesis tools [15]. First, we use reinforcement learning techniques to enable autonomous design space explorations for timing-driven logic optimization. To produce best synthesis scripts that are tuned to individual circuits, we develop a reinforcement learning agent that automatically generates step-by-step synthesis scripts to meet target timing constraints while minimizing total area. Second, we improve the basic buffering algorithm in ABC and enable physical-aware buffering and gate sizing by integrating the RePlAce [5] placement tool into the logic synthesis flow, whereby global placement-based wire capacitance estimates are used within gate sizing and buffering to improve timing results. Third, we incorporate the ability to handle a subset of commonly-used SDC commands in the synthesis flow.

B. Floorplan and PDN: Floorplanning and early power delivery network (PDN) synthesis are performed by TritonFPlan, which has two major components. The first component is macro-packing seeded by a mixed-size (macros and standard cells) global placement from RePlAce and an input IO placement. The macro-packing uses simulated annealing (SA) based on Parquet [1] [21]. The SA uses a B*-tree representation of the macro placement and has the bicriteria objective of maximizing standard-cell placement area while minimizing wirelength. The SA solution is snapped to legal locations with respect to top-level PDN, while comprehending constraints such as macro-specific halo and vertical/horizontal channel widths. Multiple floorplan solutions are created from the original mixed-size global placement. The second component of TritonFPlan creates a DRC-correct PDN for each macro-packing solution, following a safe-by-construction approach. The rules for metal and via geometries are extracted from user specified *config* files. RePlAce is used to determine the best of these floorplan (with PDN) solutions according to estimated total wirelength including placed standard cells. Only rectangular floorplans are currently supported. Realization of endcap, tapcell, and IO-pin methodologies is ongoing.[1]

C. Placement and PDN refinement: RePlAce [5] is a BSD-licensed open-source analytical placer based on the electrostatics analogy. In OpenROAD, RePlAce is used for (i) physical synthesis, (ii) mixed-size (macros and cells) placement during floorplanning, (iii) standard-cell placement within a given floorplan, and (iv) clock buffer legalization during clock tree synthesis (CTS) [22]. RePlAce is timing-driven, taking industry-standard LEF/DEF, Verilog, SDC and Liberty formats. RePlAce incorporates (i) FLUTE [8] to estimate Steiner wirelength; (ii) a fast RC estimator for parasitics estimation; and (iii) OpenSTA [20] for static timing analysis during placement. RePlAce applies a signal net reweighting iteration [9] based on OpenSTA's analysis to improve timing. Note that RePlAce does not currently change (i.e., buffer or size) the netlist provided by physical-aware synthesis (LS).

After placement, the PDN is further refined based on improved estimates of spatial current distribution. The key goal is to enable single-pass, correct-and-safe-by-construction refinement of the PDN. The floorplan-stage PDN is based on using a single pitch throughout the chip; after placement, this PDN is selectively depopulated. The chip area is tiled into regions, and for each region, one of a set of depopulated PDN wiring templates (cf. the "config" files noted above) is chosen. These templates are stitchable so that they obey design rules when abutted. The PDN tool takes as input a set of predefined templates, an early (floorplanning-stage) placed DEF for a design, and available power analysis information (e.g., our OpenSTA tool can provide instance-based power reporting). A trained convolutional neural network (CNN) then selects a safe template for each region.

D. Clock Tree Synthesis: TritonCTS [10, 22] performs clock tree synthesis (CTS) for low-power, low-skew and low-latency clock distribution, based on the GH-Tree (generalized H-Tree) paradigm of Han *et al.* [10]. A dynamic programming algorithm finds a clock tree topology with minimum estimated power, consistent with given latency and skew targets. The capacitated k-means algorithm from [13] is adapted to perform sink clustering. TritonCTS has interfaces with the placer (RePlAce) for legalization of inserted clock buffers.

E. Global Routing: UTD-BoxRouter is a modified version of BoxRouter 2.0 [6]. The tool reads LEF and placed DEF. It defines global routing cells known as gcells and performs global routing minimizing the congestion and overflow within the cells while minimizing wire length and vias. The tool generates the route guides

[1]Note that TritonFPlan requires the user to specify several *config* files, e.g., IP*.cfg to capture macro packing rules, and PDN.cfg to capture safe-by-construction metal and via geometry information using a regular grid. These files are part of one-time tool enablement that circumvents the inability of academic open-source tools and developers to see unencrypted foundry enablements.

necessary for subsequent detailed router execution. This global router first solves the 2D routing routing problem through the use of prerouting, integer linear programming and negotiation-based A* search for routing stability. This global router performs a 2D-to-3D mapping using a layer assignment using an integer linear programming algorithm that is aware of vias and blockages.

F. Detailed Routing: TritonRoute [14] takes as input LEF and placed DEF, then performs detailed routing for both signal nets and clock nets given a global routing solution in route guide format [17]. Prior to the detailed routing, (i) TritonRoute preprocesses the global routing solution using breadth-first search to reduce the probability of loops generated in later stage while net connectivity is preserved; and (ii) TritonRoute identifies unique instances considering orientation and routing track offsets, and generates pin access patterns to aid connections to pins.

The flow proceeds sequentially through track assignment and detailed routing stages. First, track assignment uses a fast greedy heuristic to determine tracks for each global routing segment. Second, clip-based initial detailed routing solves a multi-terminal, multi-net switchbox routing problem. Clips can be routed in parallel. In each clip, nets are routed sequentially using a multi-terminal A* algorithm. Third, multiple iterations of search and repair are performed to reduce wirelength and via count, as well as to help DRC convergence.

3 ANALYSIS AND SUPPORT TOOLS

OpenROAD uses a number of analysis and infrastructure tools that are used throughout the flow.

A. Static Timing Analysis: OpenSTA [20] is a GPL3 open-sourced version of the commercial Parallax timer. The Parallax timing engine has been offered commercially for nearly two decades, and has been incorporated into over a dozen EDA and IC companies' timing analysis tools. OpenSTA is publicly available on GitHub [20]. It supports multiple advanced foundry nodes and standard timing report styles.

B. Parasitic Extraction: The parasitic extraction (PEX) tool processes a foundry process design kit (PDK) to build linear regression models for wire resistance, ground capacitance, and coupling capacitances to wires on the same layer, or in the adjacent layers above and below. A basic use case is for a tool in the OpenROAD flow (e.g., CTS, global routing, static timing analysis) to call PEX, providing an input DEF file that consists of the wire of interest and its neighbors. The output is provided as a SPEF file that contains the extracted parasitics. Anticipated evolutions include interfacing the PEX functions to a possible future IDEA-wide physical design database, and extending the model-fitting approach to achieve low-overhead parasitic estimators for use in timing-driven placement and crosstalk estimation during global routing.

C. Cloud Infrastructure: For users to take advantage of OpenROAD tools, a cloud infrastructure effort aims to provide an end-to-end user experience. In our cloud deployment, users subscribe their Git repo to our cloud system. Once a design change is pushed to the Git repo, the design is automatically compiled by the current version of the OpenROAD flow and the user receives a notification by email when the flow is complete. The user can then download the outcome files through a web browser. If needed, the user can also monitor the progress of the flow on our web-based front end. Our cloud deployment is elastic as it leverages more computing resources when more users log into the service.

D. Integration and Testing: The individual tools described above comprise a tool chain that produces an implemented design ready for final verification and fabrication. Initial platform support is targeted for CentOS 6, with tool- and flow-specific support. Here, we leverage a testcase suite based around existing designs that have previously been taped out; these designs range across complexity and process. Our suite of testcases also includes a number of SoCs that are currently in development. A continuous integration test suite validates the tools individually during development and tracks regression metrics and feature impact.

4 MAJOR FUTURE EXTENSIONS

Other elements of the OpenROAD project under development include pervasive machine learning optimizations across the flow, early SoC planning, and chip-package-PCB co-design.

A. METRICS 2.0: To enable large-scale application of machine learning and ultimately a self-driving OpenROAD flow, we are developing METRICS 2.0 [11], which can serve as a unified, comprehensive design data collection and storage infrastructure (see [18]). A METRICS 2.0 dictionary provides a standardized list of metrics suitable for collection during tool/flow execution that capture key design parameters as well as outcomes from various tools in the design flow. Examples of these metrics include "number of instances", "number of congested global cells" and "total runtime". We also propose a system architecture based on JavaScript Object Notation (JSON) for data logging, and MongoDB database [7] for data storage and retrieval of the metrics. METRICS 2.0 will leverage machine-learning frameworks such as TensorFlow, which provides interfaces to read and write into MongoDB, and enables fast deployment of machine learning algorithms. The outcomes from the machine-learning algorithms will be used to tune the operation of the tools in the OpenROAD flow.

B. Early SoC Planning: To improve turnaround time, we plan to initiate the OpenROAD tool chain with reliable tentative floorplans as flow starting points, to minimize the likelihood of run failures. Early floorplan estimates for the SoC can be enhanced by embedding physical implementation information in each IP (e.g., using the vendor extension mechanism within industry-standard IP-XACT descriptions), and by making use of technology- and tool chain- specific parameters and statistical models. Combining and elaborating such information enables early area and performance estimates.

C. SoC-PCB-PKG co-design: We plan to develop layout generation flows that can co-optimize across the SoC, package (PKG) and PCB domains. Today, SoC, PKG and PCB tools and flows are largely disjoint; large time is required to converge across the three designs with manual iterations. To deliver fast turnaround time in the PKG and PCB domains, a Unified Planning Tool (UPT) that seamlessly coordinates among the three databases and enables quick iterations is essential. The UPT would include optimization engines to evaluate the complex tradeoffs across the three design spaces. Nearer-term efforts pursue development of open-source PKG routing and PCB place-and-route tools.

5 LESSONS LEARNED

At this writing, OpenROAD is nine months in from contract signing. The Alpha release of OpenROAD's RTL-to-GDSII (netlist to routed DEF) flow will be in July 2019, and the v1.0, 16/14nm-capable release will be a year later. Several lessons learned make it likely that we will reallocate resources to solve critical gaps (open-source back-end database, program management, software engineering quality, etc.). A partial list follows.

The right mindsets are needed. The need for the EDA and IC design ecosystem to embrace open-source as long-overdue culture change is described in [3, 12]. From the proposal stage on, all PIs in the DARPA IDEA program (and, especially, all PIs in OpenROAD) signed up to a "this is not business as usual" compact, with a clear understanding that "the metric is (tapeout-capable, liberally-licensed) working code, not papers".

Project expectations must be consistent. GitHub issues filed (and numerous other communications) indicate that the expectations of "users" of our free, open-source tools are heavily informed by previous experience with commercial EDA offerings (which embody billions of dollars of R&D investment). Our research and development trajectory aims at a *required deliverable* of "no human in the loop", "self-driving tools and flows" with "24-hour turnaround time". This is at odds with requests for "poor man's Innovus" functionality, or better PPA calibrations against the outputs of commercial SP&R flows. Managing this contradiction, e.g., with stronger "product management", will be necessary.

Basic, structural impediments to a FOSS EDA ecosystem must be recognized and solved. Several obstacles to efficient progress were identified early in the project. (i) Since universities are not foundry-qualified, there are aspects of (encrypted) design enablement that our tools simply cannot read. Until such qualification exists, workarounds consist of one-time "calibrations" or "characterizations" that require qualified commercial tools to be run by licensed mutual customers of tool vendors and foundry. (ii) Since IDEA does not develop "golden" signoff tools (physical verification, extraction, STA, etc.), our tools must guardband heavily to remain "correct and safe by construction". Eventual program PPA goals are challenging in this light. (iii) Since universities are not commercial entities, typically bug reports are received without testcases (due to lack of common NDAs in place). Normal commercial NDAs are blocked at universities by mutual exclusions, export control-related policies, "non-trusted" nature of universities and students, and any number of other factors. There are no public enablements that (i) have complexity commensurate with tool development for advanced production nodes, or (ii) are accessible to both commercial and non-commercial entities.

6 CONCLUSION

We have reviewed scope and status of OpenROAD, a DARPA IDEA project that aims to develop a self-driving, open-source digital layout implementation tool chain. We plan to make source code available at **http://github.com/The-OpenROAD-Project**, and a cloud deployment is available at **http://flow.theopenroadproject.org**. Separately, outreach will be an important element, e.g., contests [16], workshops [24], and soliciting global collaborations [12]. We welcome all feedback, participation and contributions.

ACKNOWLEDGMENTS

We thank Andreas Olofsson of DARPA for providing guidance. We also thank Colin Holehouse, Matteo Coltella and David Urquhart of ARM, and project PIs and participants at Qualcomm, UC San Diego and the University of Michigan. The OpenROAD project is supported by DARPA (HR0011-18-2-0032). Mr. Fogaça's studies are financed by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

REFERENCES

[1] S. N. Adya and I. L. Markov, "Fixed-Outline Floorplanning: Enabling Hierarchical Design", *IEEE Trans. on VLSI* 11(6) (2003), pp. 1120-1135.
[2] C. J. Alpert, W.-K. Chow, K. Han, A. B. Kahng, Z. Li, D. Liu, S. Venkatesh, "Prim-Dijkstra Revisited: Achieving Superior Timing-driven Routing Trees", *Proc. ISPD*, 2018, pp. 10-17.
[3] A. M. Caldwell, A. B. Kahng and I. L. Markov, "Toward CAD-IP Reuse: The MARCO GSRC Bookshelf of Fundamental CAD Algorithms", *IEEE Design & Test of Computers* 19(3) (2002), pp. 70-79.
[4] W.-T. J. Chan, A. B. Kahng, S. Nath and I. Yamamoto, "The ITRS MPU and SOC System Drivers: Calibration and Implications for Design-Based Equivalent Scaling in the Roadmap", *Proc. ICCD*, 2014, pp. 153-160.
[5] C.-K. Cheng, A. B. Kahng, I. Kang and L. Wang, "RePlAce: Advancing Solution Quality and Routability Validation in Global Placement", *IEEE Trans. on CAD* (2018), to appear. DOI: 10.1109/TCAD.2018.2859220
[6] M. Cho, K. Lu, K. Yuan and D. Z. Pan, "BoxRouter 2.0: Architecture and Implementation of a Hybrid and Robust Global Router", *Proc. ICCAD*, 2007, pp. 503-508.
[7] K. Chodorow, *MongoDB: The Definitive Guide: Powerful and Scalable Data Storage*, O'Reilly Media, Inc., 2013.
[8] C. Chu and Y.-C. Wong, "FLUTE: Fast Lookup Table Based Rectilinear Steiner Minimal Tree Algorithm for VLSI Design", *IEEE Trans. on CAD* 27(1) (2008), pp. 70-83.
[9] M. Fogaça, G. Flach, J. Monteiro, M. Johann and R. Reis, "Quadratic Timing Objectives for Incremental Timing-Driven Placement Optimization", *Proc. ICECS*, 2016, pp. 620-623.
[10] K. Han, A. B. Kahng and J. Li, "Optimal Generalized H-Tree Topology and Buffering for High-Performance and Low-Power Clock Distribution", *IEEE Trans. on CAD* (2018), to appear. DOI: 10.1109/TCAD.2018.2889756
[11] S. Hashemi, C.-T. Ho, A. B. Kahng, H.-Y. Liu and S. Reda, "METRICS 2.0: A Machine-Learning Based Optimization System for IC Design", *Workshop on Open-Source EDA Technology*, 2018, pp. 21:1-21:4.
[12] A. B. Kahng, "OpenROAD: Toward a Self-Driving, Open-Source Digital Layout Implementation Tool ChainâĂŹâĂŹ, invited talk at the *Emerging Technologies for EDA Workshop*, Hsinchu, March 2019. https://vlsicad.ucsd.edu/NEWS19/OpenROAD-final-abk.pptx
[13] A. B. Kahng, J. Li and L. Wang, "Improved Flop Tray-Based Design Implementation for Power Reduction", *Proc. ICCAD*, 2016, pp. 20:1-20:8.
[14] A. B. Kahng, L. Wang and B. Xu, "TritonRoute: An Initial Detailed Router for Advanced VLSI Technologies", *Proc. ICCAD*, 2018, pp. 81:1-81:8.
[15] C. Wolf, J. Glaser and J. Kepler, "Yosys – A Free Verilog Synthesis Suite", *Proc. Austrian Workshop on Microelectronics*, 2013.
[16] ICCAD-2019 Global Routing Contest, http://iccad-contest.org/2019/
[17] ISPD-2018 Initial Detailed Routing Contest, http://www.ispd.cc/contests/18/
[18] The METRICS Initiative, MARCO/DARPA Gigascale Silicon Research Center, https://vlsicad.ucsd.edu/GSRC/metrics/
[19] The OpenROAD Project, https://theopenroadproject.org
[20] OpenSTA, https://github.com/abk-openroad/OpenSTA
[21] *Parquet*, http://vlsicad.eecs.umich.edu/BK/parquet/#DOWN
[22] TritonCTS, https://github.com/abk-openroad/TritonCTS
[23] VLSI CAD Bookshelf 2, MARCO/DARPA Gigascale Silicon Research Center, http://vlsicad.eecs.umich.edu/BK/
[24] Workshop on Open-Source EDA Technology, http://woset.org