OpenROAD: Toward a Self-Driving, Open-Source Digital Layout Implementation Tool Chain

Tutu Ajayi[5], David Blaauw[5], Tuck-Boon Chan[2], Chung-Kuan Cheng[1], Vidya. A. Chhabria[6], Kyojin Choo[5], Matteo Coltella[3], Sorin Dobre[2], Ronald Dreslinski[5], Mateus Fogaça[1], Soheil Hashemi[4], Abdelrahman Hosny[4], Andrew B. Kahng[1], Minsoo Kim[1], Jiajia Li[2], Zhaoxin Liang[6], Uday Mallappa[1], Paul Penzes[2], Geraldo Pradipta[6], Sherief Reda[4], Austin Rovinski[5], Kambiz Samadi[2], Sachin S. Sapatnekar[6], Lawrence Saul[1], Carl Sechen[7], Vaishnav Srinivas[2], William Swartz[7], Dennis Sylvester[5], David Urquhart[3], Lutong Wang[1], Mingyu Woo[1] and Bangqi Xu[1]
[1]UC San Diego, [2]Qualcomm, [3]Arm, [4]Brown University, [5]U. Michigan, [6]U. Minnesota, [7]U. Texas-Dallas

***Abstract*—We describe the scope and initial efforts of OpenROAD, a project in the DARPA IDEA program that pursues open-source tools for 24-hour, "no human in the loop" digital layout generation across integrated circuit, package and board domains. If successful, OpenROAD will help realize the IDEA goal of "democratization of hardware design", by reducing cost, expertise, schedule and risk barriers that confront system designers today. Several novel technical directions follow directly from the IDEA program's 24-hour, no-humans goals. These include (i) enablement of pervasive machine learning in and around design tools and flows, (ii) parallel search and optimization to exploit available cloud resources, (iii) partitioning and problem decomposition to reduce solution latency, and (iv) layout generation methodologies that provide "freedoms from choice" without undue loss of design quality. Further, the development of open-source, self-driving design tools is in and of itself a "moon shot" with numerous technical and cultural challenges.**

I. Introduction

Even as hardware design tools and methodologies have advanced over the past decades, the semiconductor industry has failed to control product design costs, as depicted in Figure 1. 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.

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 [2]. 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. DoD researchers and development teams do not have resources to execute such a strategy, and hence see typical hardware design cycles of 12-36 months.

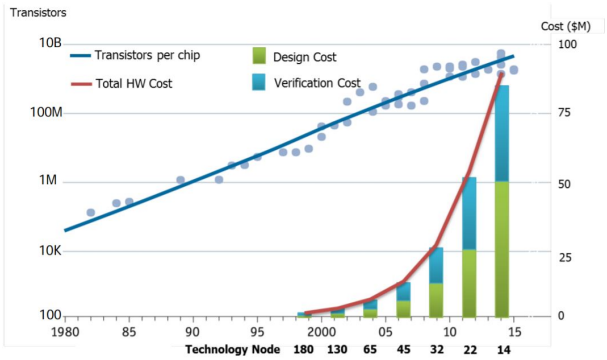

Fig. 1. Design technology crisis.

A. IDEA and the OpenROAD Project

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 [17] 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. With a team of performers that includes Qualcomm, Arm, and multiple universities led by UC San Diego, OpenROAD seeks to develop a **fully autonomous**, ***open-source*** tool chain for digital layout generation 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*).

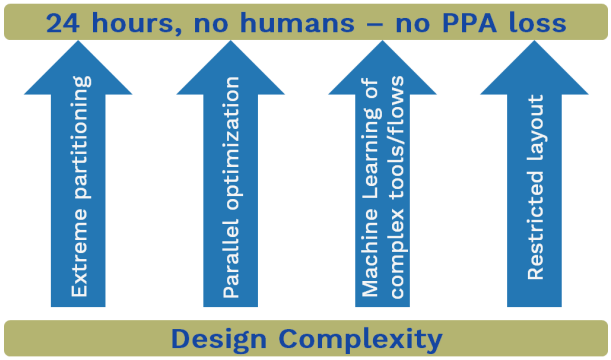

Fig. 2. Design complexity.

Three innovative base technologies underlie the OpenROAD team's strategy to achieve no-human-in-loop (NHIL), 24-hour turnaround time (TAT). First, ***machine learning*** based modeling and prediction of tool and flow outcomes will enable the tool auto-tuning and design-adaptivity required for NHIL, new optimization cost functions in EDA tools, and new tool knobs that tools may expose to users. Second, ***extreme partitioning*** strategies for decomposition will enable thousands of tool copies running on cloud resources to maximize success within human, CPU, schedule bounds. Quality loss from decomposition is recovered with improved predictability of flow steps, along with stronger optimizations. Third, ***parallel/distributed search and optimization*** will leverage available compute resources (e.g., cloud) to maximize design outcomes within resource limits, and in the face of noise and chaos in the behavior of complex metaheuristics. A complementary precept is to reduce design and tool complexities through "freedoms from choice" in layout generation; this can increase predictability and avoid iterations in the design process. The synergy of base technologies and restrictions of the layout solution space is illustrated in Figure 2.

B. A New Paradigm

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. To start the project, we bring (i) significant initial software IP including donated source code bases, and a commercial static timing analysis tool; (ii) a significant set of academic software IP and skillsets; (iii) leading SoC and IP know-how and guidance from industry partners Qualcomm and Arm; (iv) an in-built Internal Design team (U. Michigan) to provide de facto product engineering and alpha customer functions; and (v) a broad agenda of industry and academic outreach. Furthermore, OpenROAD derives its "Base Technologies" efforts directly from the IDEA program requirements (no-humans, 24-hours, no loss of PPA quality). We view the cohesive integration of machine learning, problem partitioning and decomposition, and parallel/distributed search and optimization as essential to reaching the IDEA target.

This paper. The remainder of this paper will outline the current status of OpenROAD's GitHub-deployed tools and flow. Early proof points and calibrations in the realm of digital IC layout generation ("RTL-to-GDSII") have been obtained in multiple foundry design enablements including 16nm FinFET technology.

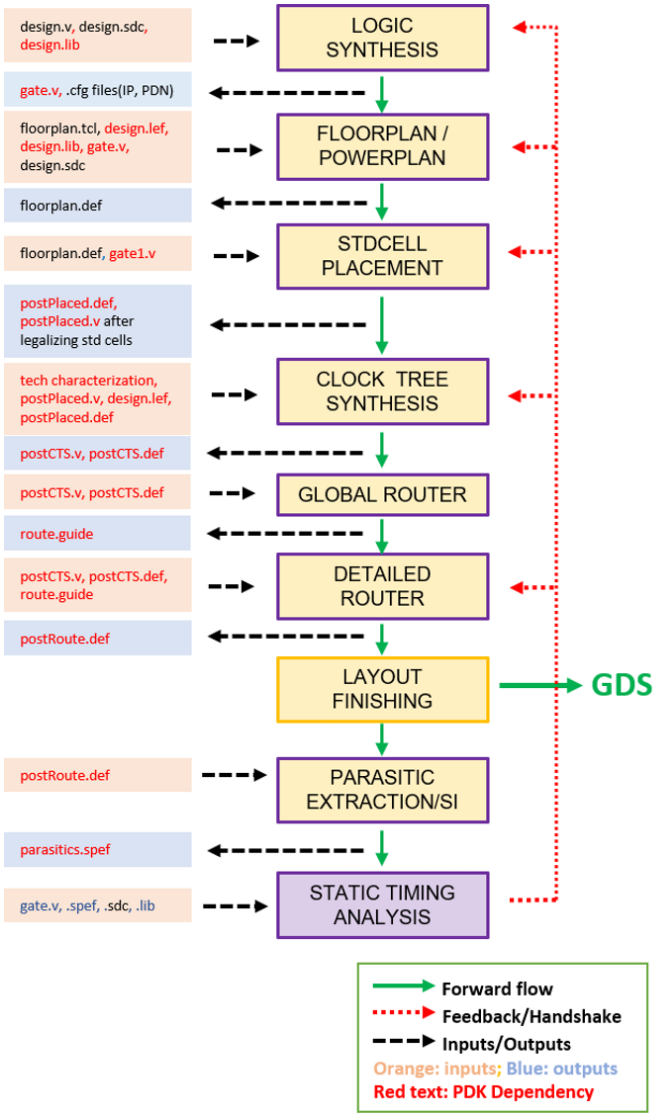

Fig. 3. The OpenROAD flow.

II. Current Status: Layout Tool Chain

OpenROAD's layout generation 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 tapeout-ready GDSII file. Figure 3 illustrates the flow of tools corresponding to individual OpenROAD tasks. These include logic synthesis (LS), floorplan (FP) and power delivery network (PDN) generation, placement, clock tree synthesis (CTS), routing and layout finishing.

A. Logic Synthesis

The major gap in open-source LS is timing awareness and optimization. OpenROAD has explored two avenues toward enablement of timing-driven synthesis. First, we use machine

learning techniques to enable autonomous design space explorations for timing-driven logic optimization. It is often the case that synthesis scripts contain tens of commands in order to make a design meet its timing and area goals. These scripts are crafted by human experts. To produce best synthesis scripts that are tuned to individual circuits, we design machine learning agents that automatically generate step-by-step synthesis scripts to meet target timing and delay goals. Second, we enable physical-aware logic synthesis by integrating the RePlAce [20] placement tool into the logic synthesis flow, whereby global placement-based wire capacitance estimates are used within logic synthesis to improve timing results. Existing academic tools are oblivious to the outcomes of subsequent steps in the design flow, and our ultimate goal is to feed back wiring estimates as they are refined in physical design steps (e.g., standard-cell placement and global routing) to improve synthesis results.

B. Floorplan and PDN

Floorplanning and power delivery network synthesis are performed by TritonFPlan, which has two major components. The first component is integer programming-based macro block packing that is aware of macro-level connectivity and is seeded by a mixed-size (blocks and standard cells) global placement. The second component is Tcl-based power delivery network (PDN) generation following a safe-by-construction approach. TritonFPlan requires the user to specify several *config* files, e.g., IP_global.cfg and IP_local.cfg capture macro packing rules, and PDN.cfg captures safe-by-construction metal and via geometry information. These *config* files are necessitated by the inability of academic open-source tool developers (or, their tools) to see complete unencrypted design enablements from the foundry. We discuss this below in Section IV. The TritonFPlan tool uses mixed-size placer (RePlAce) for its initial global placement. The generated macro global locations provide a starting point from which multiple floorplan solutions are created. For each of the generated floorplan solutions with fixed macros and PDN, we use our placer (RePlAce) again, to determine the best floorplan according to an estimated total wirelength criterion. Limitations include support of only rectangular floorplans, and macro counts less than 100.

C. Placement

RePlAce [3, 20] is a BSD-licensed open-source analytical placer based on the electrostatics analogy. In OpenROAD, RePlAce is used for mixed-size (macros and cells) placement during floorplanning, for standard-cell placement within a given floorplan, and during clock tree synthesis (CTS) [18] for clock buffer legalization. Timing-driven placement is achieved with integration of FLUTE [5] and OpenSTA [16], along with a signal net reweighting iteration [6]. The timing-driven TD-RePlAce tool takes input in standard LEF/DEF, Verilog, SDC and Liberty formats, and incorporates a fast RC estimator for parasitics extraction. Ongoing efforts aim to enable routability-driven mode using commercial format (LEF/DEF/Verilog). Figure 4 shows the RePlAce placement of a small RISC-V based block (foundry 16nm technology) produced by the University of Michigan internal design advisors subteam.

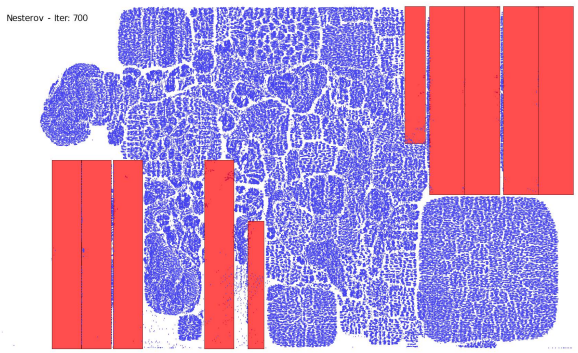

Fig. 4. Foundry 16nm RISC-V based design block from the University of Michigan, after RePlAce mixed-size placement. Red color indicates macros and blue color indicates standard cells.

D. Clock Tree Synthesis

TritonCTS [7, 18] 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 [7]. A dynamic programming algorithm finds a clock tree topology with minimum estimated power, consistent with given latency and skew targets. Linear programming is used to perform sink clustering and clock buffer placement. Leaf-level routing may be performed using either the single-trunk Steiner tree or the Prim-Dijkstra [1] algorithm.

In the layout generation flow, TritonCTS has interfaces with the placer (RePlAce) and the router (TritonRoute [9]). The placer is used for legalization of inserted clock buffers. The router maps sink pins to GCELLs that should be used for clock tree routing. TritonCTS inputs are LEF, placed DEF, placed gate-level Verilog, a configuration file and library characterization files. (For each foundry enablement, a one-time library characterization is needed. Currently, this library characterization is expected to be performed by some outside entity (foundry or tool user) using commercial EDA tools.) TritonCTS outputs are "buffered" placed DEF, "buffered" gate-level Verilog, and clock tree global routing in ISPD18 route guides format [14]. TritonCTS is publicly available on GitHub [18]. Early validations have been made using 16nm and 28nm foundry enablements. Improvements to handle multiple clock sources, non-default routing rules, etc. are ongoing.

E. Routing

TritonRoute [9] consumes LEF and placed DEF, then performs detailed routing for both signal nets and clock nets given a global routing solution in route guide format [14]. Prior to the detailed routing, TritonRoute preprocesses the global routing solution using a fast approximation algorithm [10] to ensure a Steiner tree structure for each net. Thus, congestion and wirelength are minimized while net connectivity is preserved in detailed routing stage. The detailed routing problem is then iteratively solved on a layer-by-layer basis, and each layer is partitioned into disjoint routing panels. The panel routing is formulated as a maximum weighted independent set (MWIS) problem and solved in parallel using a mixed integer linear programming (MILP)-based approach. The MWIS formulation optimally assigns tracks considering

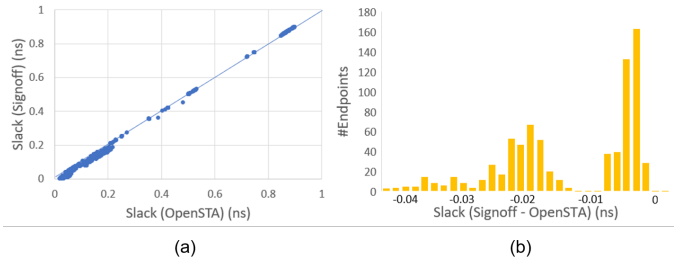

Fig. 5. Comparison between OpenSTA and a leading signoff timer (Signoff) for a small 28nm testcase. (a) Endpoint slacks of OpenSTA vs. Signoff timer. (b) Histogram of endpoint slack differences between OpenSTA and Signoff timer.

(i) intra- and inter-layer connectivity, (ii) wirelength and via minimization, and (iii) various design rules. By an alternating panel routing strategy with multiple iterations, inter-panel and inter-layer design rules are properly handled and track assignments are maximized. To date, TritonRoute supports major TSMC16 metal and cut spacing rules, i.e., LEF58_SPACING, LEF58_SPACINGTABLE and LEF58_CUTCLASS. An early evaluation shows approximately 10× reduction of spacing rule violations in a TSMC16 design block. Detailed routing flow with integration and optimization of local net routing is the next step towards a 100%-completion, DRC-clean layout capability.

III. Current Status: Other Elements

Other elements of the OpenROAD project under development include the above-mentioned "base technologies" (machine learning, partitioning, parallel optimization), a design performance analysis backplane (parasitic extraction, static timing analysis, and power/signal integrity), cloud infrastructure for tool/flow deployment and machine learning, the "internal design advisors" task, and corresponding self-driving layout generation capability in the package and PCB domains. This section outlines the status of several of these project elements.

A. Static Timing Analysis

OpenSTA [16] 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 [16] since September 2018. The developer, James Cherry, has added Arnoldi delay calculation, power reporting and other enhancements since the original release. OpenSTA has been confirmed to support multiple advanced foundry nodes, and it supports standard timing report styles. To date, the OpenSTA timer has been integrated into TD-RePlAce (timing-driven enhancement of RePlAce), physical-aware synthesis (Yosys [11]) and a gate-sizing tool (TritonSizer [19]). Figure 5(a) shows a comparison of endpoint timing slacks from OpenSTA and a commercial signoff timer. Figure 5(b) shows the distribution of endpoint slack differences between OpenSTA and the commercial signoff timer.

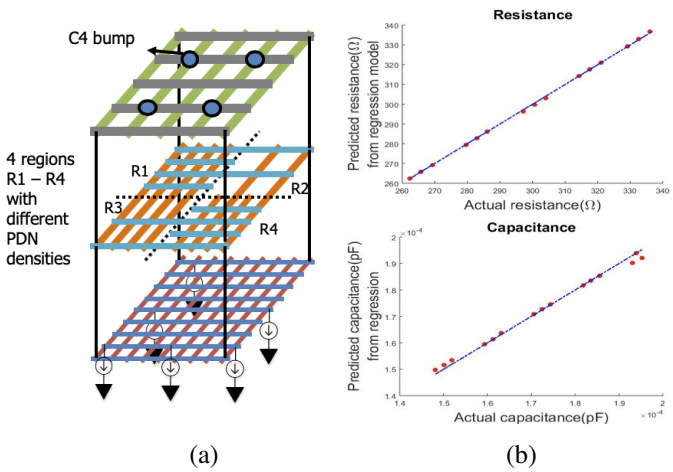

Fig. 6. (a) Example PDN templates; and (b) validation of the regression model for R, C in PEX.

B. Parasitic Extraction

In OpenROAD's approach, 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 another tool in the flow (e.g., CTS, global routing, 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. Figure 6(b) compares the actual and predicted values of the resistance and capacitance obtained from test nets to validate the regression model, and shows a good fit. 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, crosstalk estimation during global routing, etc.

C. Power Integrity

A key goal of our power integrity analysis effort is to enable single-pass, correct-and-safe-by-construction specification of the power delivery network (PDN) layout strategy across the SoC. Our power delivery network (PDN) synthesis tool tiles the chip area into regions, and selects one of a set of available PDN wiring templates (cf. the "config" files noted in the Floorplanning discussion, above) in each region. These templates are stitchable so that they obey all design rules when abutted. The PDN tool takes in a set of predefined templates (Figure 6(a)), 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 ML model then determines a safe template in each region. An early prototype shows that the ML-based approach can successfully deliver a PDN to satisfy a given (e.g., 1mV static) IR drop specification.

D. Cloud Infrastructure

For users to take advantage of OpenROAD tools as well as tools developed by other collaborators, a cloud infrastructure effort aims to provide an end-to-end seamless 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 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, or when a user requests parallel processing capabilities. For instance, the service can elastically deploy multiple machines in order to run a tool (e.g., placer) with multiple random seeds to obtain a better result within a given wall time budget. Or, in conjunction with global design partitioning, the cloud deployment can run each design partition in parallel on a cloud instance, to maximize parallel speedup and minimize design turnaround time.

E. METRICS 2.0

To enable large-scale applications of machine learning (ML) and ultimately a self-driving OpenROAD flow, we are developing METRICS 2.0 [8], which can serve as a unified, comprehensive design data collection and storage infrastructure (see [13]). A METRICS 2.0 dictionary provides a standardized list of metrics suitable for collection during tool/flow execution, to capture key design parameters as well as outcomes from various tools in the design flow. We also propose a system architecture based on JavaScript Object Notation (JSON) for data logging, and MongoDB database [4] for data storage and retrieval of the metrics. Figure 7 illustrates the METRICS 2.0 system architecture. The proposed architecture eliminates the need to create database schemas and enables seamless data collection. METRICS 2.0 is tightly coupled with machine-learning frameworks such as TensorFlow, which provides easy interfaces to read and write into MongoDB, and enables fast deployment of machine learning algorithms.

F. Early SoC Planning

In light of NHIL and 24-hour turnaround time requirements, it is important to initiate the OpenROAD tool chain with with reliable tentative floorplans as flow starting points, to minimize the likelihood of run failures. This is a key link between the "system-level design" (IDEA TA-2) and "layout generation" (IDEA TA-1, which we address in OpenROAD). 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 that can indicate doomed-to-fail floorplan candidates or suggest design implementation fine-tuning (hard-macro placement, grouping, register slice insertions, etc.) in viable floorplans.

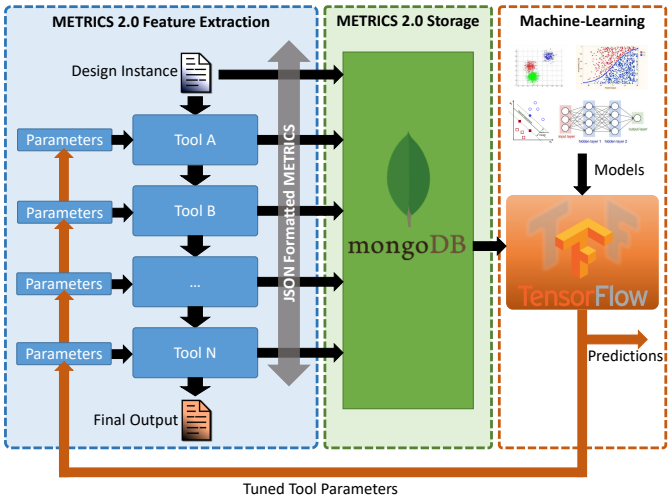

Fig. 7. Overall METRICS 2.0 system architecture.

G. 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 maintained at [17]. To evaluate the flow, non-tool developer entities in our team (i.e., U. Michigan, Qualcomm and Arm) perform fine-grained analyses on our tool outputs and provide target calibration metrics for tool developers. Here, we leverage a testcase suite based around existing designs that have previously been taped out; these designs range across complexity (from small blocks to whole chips) and process (e.g., 16nm and 65nm). Our suite of testcases also includes cutting-edge complex SoCs that are currently in development. A continuous integration test suite validates the tools individually during development and tracks regression metrics and feature impact.

IV. Looking Forward

Our near-term efforts will continue development of the tools and flow described above. More broadly, we will also seek to address various technical, structural and cultural challenges that have become apparent even at project outset.

One key technical challenge is to develop design automation technologies as well as 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; weeks if not months are required to converge across the three designs with manual iterations. To deliver NHIL, 24-hour turnaround time in the PKG and PCB domains, a Unified Planning Tool that seamlessly coordinates among the three databases and enables quick iterations is essential. Figure 8 illustrates our envisioned Unified Planning Tool. The Unified Planning Tool would also include optimization engines, using analytical and ML approaches to evaluate the complex trade-offs across the three design spaces.

Some other technical challenges include the following. (1) The "small and expensive" nature of design process data in IC design – where obtaining a single data point might

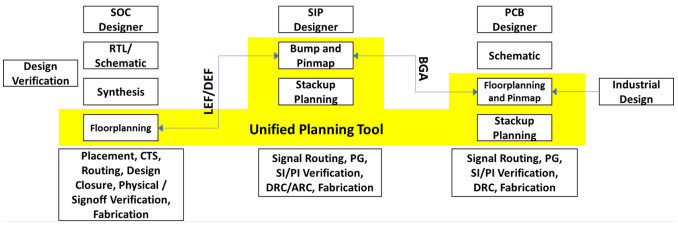

Fig. 8. Illustration of Unified Planning Tool.

require three weeks to run through a tool flow – challenges machine learning and development of "intelligent" tools and flows. (2) The need for new, common standards for measuring and modeling of hardware designs and design tools must be compatible with the IP stances of foundries and commercial EDA; this may shape the future opening of a "Linux of EDA" to broad participation. And, (3) it will be difficult to illuminate the critical junctures where "human intelligence" is now required, yet must be replaced by "machine intelligence", in the hardware design process.

Several structural challenges stem from our status as academic tool developers of a tool chain that must produce tapeout-ready GDSII. (1) OpenROAD tools will likely not be foundry-qualified, which implies that OpenROAD tools and tool developers will not be able to read encrypted advanced-node PDKs. To achieve safety and correctness by construction of the tapeout database, OpenROAD tools require *config* files and one-time generation of "OpenROAD kit" elements, for each foundry enablement. (2) OpenROAD's analyses and estimators for timing, parasitics and power/signal integrity are not "signoff" verifiers. Thus, additional performance guard-bands are required throughout the layout generation flow. And, post-OpenROAD verifications may be performed by designers and/or foundries. (3) OpenROAD tools are developed and released by non-commercial entities. Commercial EDA vendors receive bug/enhancement requests accompanied by a testcase that exhibits the bug or behavior at issue. By contrast, bug reports that we receive are unlikely to be accompanied by testcases due to blocking NDA / IP restrictions. This complicates the bug-fixing and enhancement process.

Finally, our outreach efforts seek culture change and engagement across the community of potential developers and tool users. For example, in the academic research world, a lab's code is its competitive advantage (or, potential startup), and liberal open-sourcing is still rare (cf. [12]). We hope that OpenROAD and the IDEA/POSH programs help drive culture change in this regard. With regard to tool users, we observe that commercial EDA tools are invariably driven to production-worthiness by "power users" – i.e., paying customers who have deep vested interests in the capability and maturation of a given tool. Traditionally, power users expose a new tool to leading-edge challenges and actively drive tool improvement. For OpenROAD, finding our "power users" is a critical need, especially since they would be able to improve tools and flows at the source-code level.

V. Conclusion

In this paper, we have reviewed the scope and status of OpenROAD, a DARPA IDEA project that aims to develop a self-driving, open-source digital layout implementation tool chain. The above is only a snapshot, taken six months into a four-year project. We welcome feedback, participation and contributions.

Acknowledgments

We thank Payal Agarwal, Chia-Tung Ho, Hsin-Yu Liu and Siddharth Singh for their contributions to the OpenROAD project. Mateus Fogaça (permanent affiliation: UFRGS, Porto Alegre, Brazil) developed TritonCTS as a visiting Ph.D. student at UCSD. The OpenROAD project is supported by DARPA (HR0011-18-2-0032).

References

[1] 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. ACM/IEEE Intl. Symp. on Physical Design*, 2018, pp. 10-17.

[2] 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. IEEE Intl. Conf. on Computer Design*, 2014, pp. 153-160.

[3] 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

[4] K. Chodorow, *MongoDB: The Definitive Guide: Powerful and Scalable Data Storage*, O'Reilly Media, Inc., 2013.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] A. B. Kahng, L. Wang and B. Xu, "TritonRoute: An Initial Detailed Router for Advanced VLSI Technologies", *Proc. ACM/IEEE Intl. Conf. on Computer-Aided Design*, 2018, pp. 81:1-81:8.

[10] L. Kou, G. Markowsky and L. Berman, "A Fast Algorithm for Steiner Trees", *Acta Informatica* 15(2) (1981), pp. 141-145.

[11] C. Wolf, J. Glaser and J. Kepler, "Yosys – A Free Verilog Synthesis Suite", *Proc. Austrian Workshop on Microelectronics*, 2013.

[12] VLSI CAD Bookshelf 2, MARCO/DARPA Gigascale Silicon Research Center, http://vlsicad.eecs.umich.edu/BK/

[13] The METRICS Initiative, MARCO/DARPA Gigascale Silicon Research Center, https://vlsicad.ucsd.edu/GSRC/metrics/

[14] ISPD-2018 Initial Detailed Routing Contest, http://www.ispd.cc/contests/18/index.html

[15] LEF/DEF reference 5.8, http://www.si2.org/openeda.si2.org/projects/lefdefnew

[16] OpenSTA, https://github.com/abk-openroad/OpenSTA

[17] The OpenROAD Project, https://theopenroadproject.org

[18] TritonCTS, https://github.com/abk-openroad/TritonCTS

[19] TritonSizer, https://github.com/abk-openroad/TritonSizer

[20] RePlAce, https://github.com/abk-openroad/RePlAce