Looking Into the Mirror of Open Source

(Invited Paper)

Andrew B. Kahng
CSE and ECE Departments, UC San Diego, La Jolla, CA 92093
abk@ucsd.edu

***Abstract*—The DARPA IDEA program has brought unprecedented resources and attention to development of open-source EDA. As this session convenes, IDEA is well into its second year, with "alpha" releases in the rear-view mirror, and a public v1.0 unveiling just eight months away. This talk will give a perspective on recent open-source development for digital layout generation. First, any technology should have a roadmap of quantifiable metrics, requirements, and potential solutions – projected on the axis of time. What are key aspects of such a roadmap for open-source EDA technology? Second, what does open-source EDA tell us about ourselves? The goal of open-source is a mirror for the ecosystem of academic research, semiconductor design, and commercial EDA. Who is willing to contribute? Who is capable of contributing? What are bars for "moving the needle" and sustainability?**

I. Introduction

The DARPA Intelligent Design of Electronic Assets (IDEA) program [27] has focused new attention on development of open-source EDA technology. The program aims to create "Silicon Compilers 2.0" on a very ambitious timeline: the v1.0 release in July 2020 is targeted to achieve no-human-in-the-loop, 24-hour generation of design rule-clean, manufacturable layouts (in particular, "RTL-to-GDS" in the digital IC domain) for up to 200M-instance systems-on-chip in commercial FinFET node enablements. Within IDEA, the OpenROAD project [31] [1] [2] seeks to realize the Silicon Compilers 2.0 vision in the digital domain, across chips, packages and boards. Further, OpenROAD aims to seed a new ecosystem of *open-source* EDA technology,[1] complementing the rapid growth of open source seen in hardware (DARPA's POSH program, CHIPS Alliance, FOSSI, OpenHW Group, etc.). (The IDEA program's 2017 launch coincides with an uptick in EDA open-sourcing activity, as seen in Figure 1 [35].)

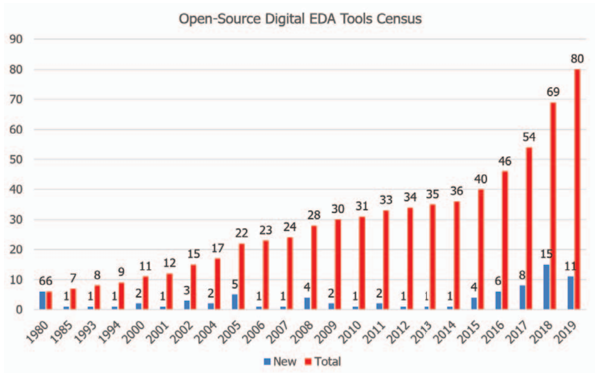

Fig. 1. "Digital" tools inventory collected at [35]. Some listed tools require a download permission step and/or do not have well-formed open-source licenses.

This invited paper gives personal perspectives on the past, present and future of open-source EDA development, with focus on RTL-to-GDS physical implementation for digital ICs. Backdrop for these perspectives spans (i) open-source releases of Capo, MLPart and UCLApack as well as the MARCO GSRC Bookshelf of Fundamental CAD Algorithms [3] [36] initiative; (ii) the METRICS initiative [32] [8] toward "measure to improve" in the IC design process; (iii) the roadmapping of EDA technology within the overall semiconductor industry roadmap; and (iv) initial learnings from OpenROAD's RTL-to-GDS efforts. Open-source EDA lives in an ecosystem of EDA researchers, commercial EDA vendors and semiconductor design. This affords perspectives on culture and potential future paths for EDA in the large. Open-source EDA is also a technology. Ideally, any technology should have a *roadmap* of quantified metrics, and of requirements and potential solutions, projected on the axis of time. This affords perspectives on potential roadmap elements for open-source EDA technology.

In the following, Section II examines open source's "relative lack of success" and impact on EDA in the past, as well as motivations for an open-source EDA culture going forward. Section III reviews several near-term "unblocking" milestones that must be achieved for open source to be viable as a foundation for research and practical impact in the digital IC (RTL-to-GDS) space. Section IV outlines longer-term challenges, and potential aspects of a future roadmap for open-source EDA. Challenges and milestones that must be kept on the research community's radar include gaps to fill, enablement of machine learning "inside and around" design tools and flows, and evolutions of research incentives and culture. Section V concludes the paper.

II. On Open Source in EDA

A. Diagnosing Today's Lack of Open-Source EDA

In the spirit of "those who do not learn from history are doomed to repeat it," it is worth reflecting on why open-source culture has been slow to emerge in our domain. After all, the field of VLSI CAD / EDA is well over a half-century old – and significant roots of commercial EDA lie in early academic tools that were released as permissive open source: SPICE, SUPREM, Magic, Espresso, FASTCAP, MIS/SIS, etc. There is relatively little culture of open source in academic EDA today, even as other fields have visibly flourished (both as academic disciplines and as commercial industries) as a consequence of open-sourcing practices. Root causes of today's non-open source EDA research culture fall into at least two buckets: (i) straight-out blockers of (open source-enabled) EDA research; and (ii) resulting drivers and incentives that are incompatible with open-source EDA. These root causes motivate a number of future goals, such as new mechanisms and community standards to support open-source EDA in the longer term, as discussed in Section IV.

Straight-out blockers. EDA researchers have learned to accept a world where *research progress* is unnecessarily hampered at many turns. Notably, EDA researchers today live in a world where published results are *irreproducible by construction*, due to the end user license agreements (EULAs) that are executed in order to use commercial EDA tools. For example, publication of commercial EDA Tcl commands and runscripts, or excerpts from tool logfiles

[1]Discussion of "What IS open source?" is omitted. See [42] for an excellent, EDA-relevant overview. Importantly, "open source" is accepted to mean "code released under an open source license", where "open source license" complies with the Open Source Definition at opensource.org/osd-annotated. Many "open" or "released" or "free" EDA codes today are actually not "open source".

978-1-7281-2350-9/19/$31.00 ©2019 IEEE

and output reports, is prohibited.[2] The same EULAs also prohibit benchmarking of commercial EDA tools.[3] A consequence is that there has not been any available "open research backplane" within which academic researchers can quickly prototype and assess their point-tool innovations. (On the positive side, the long-standing academic plaint of "we need real testcases", and the irreproducibility of results due to "proprietary testcases that cannot be released", are rapidly becoming moot with the emergence of a vibrant open-source hardware community.)

Beyond current industry-induced blockers to *research progress*, the DAC-2019 invited paper [2] notes additional, higher-level challenges to an *open-source EDA ecosystem*. These include the following. (i) Since universities are not foundry-qualified, (encrypted) design enablements cannot be accessed. As of this writing, there are no public design enablements (PDK, library, IPs, memory compiler, etc.) that can inform tool development for advanced production nodes.[4] (ii) The R&D investment needed to develop golden signoff analyses runs into the billions of dollars and thousands of engineer-years. Verifications are, in general, a difficult challenge for open-source EDA. (iii) Traditional "report a bug along with a testcase" exchanges between EDA supplier and EDA user are painful or blocked when the EDA supplier is a (non-commercial) entity, and cannot execute standard commercial MNDAs (e.g., as a consequence of national origin-blind and other policies, export control exposure, etc.).

Resulting drivers and incentives that run counter to open source. At least in the digital IC implementation domain discussed here, the above blockers have shaped how EDA research is practiced.

(1) Ripple effects of irreproducibility. Given the "irreproducible by construction" aspects of EDA research, it is no surprise that EDA conference and journal publication processes do not require results to be reproducible. In EDA, there are no venues in which independent confirmations of previously-published results can be published, as is common in the life science and physical science literatures. At the same time, the field of design technology itself is quite mature and competitive (often, in a zero-sum sense), with many workers focused on incrementally advancing the state of art for well-studied problem formulations. And even as academia worldwide is rapidly shifting its structures and financial models, variants of "publish or perish" remain inevitable for individual researchers. Given this "perfect storm" of context, there are many reasons to *not* open-source one's research software. These reasons include (i) maintaining a competitive advantage over other researchers in the field, (ii) concern over academic "bean-counting" metrics, (iii) overheads (bandwidth, risks from transparency, and "industrial" skill set) of open-sourcing that are inconsistent with Ph.D. students' goals and the latest project delivery demands, (iv) sponsors' restrictions (example: projects sponsored by the Semiconductor Research Corporation in the U.S.) or co-ownership of IP, (v) entrepreneurial ambitions, and (vi) moral objections to "giving away" the fruits of hard work. [5] Each of these reasons is perfectly rational and sensible. Indeed, 20+ years of experience with open-sourcing (UCLApack, Capo, MLPart, Bookshelf, ..., RePlAce, TritonRoute, etc.) as an aspirational (i.e., ideally but not always followed) research practice have made it clear that whether to open-source is always a 100% personal decision. (But, the cumulative impact of these personal decisions affects an entire research community.)

(2) Ripple effects of unavailability of a research 'backplane'. Over the years, a number of influential academic contests out of necessity have used testcases that embody strong simplifications to data models and/or industry formats. Perpetuating these simplifications (e.g., translating a Bookshelf variant to LEF/DEF), along with a focus on comparisons with previous works using (old) contest benchmarks, has kept academic research in a kind of parallel universe of "translated", as opposed to "standard", industry formats and testcases. Tools developed in this parallel universe are often unable to accept real-world designs and enablements. This hinders technology transfer to, and interest from, the commercial EDA and designer worlds.[6] Furthermore, without a *full-flow* backplane, heuristic innovations cannot be assessed with respect to overall design flow outcomes. Thus, to use the example of standard-cell placement, research can remain bound to a placement metric such as "half-perimeter wirelength" (HPWL), even though this may not capture 21st-century concerns such as routability or timing.

B. Benefits of Open Source

At least several main benefits of an open-source EDA research culture seem obvious. (1) *Improved science and relevance.* For academic researchers, open-source tools afford clarity and transparency, along with better science: every result is reproducible. With open source, the standard reviewer request of "please compare with the previous work of ..." in a benchmark-centric literature becomes much easier to satisfy. Implementations of older methods can be more easily migrated to modern contexts, and would no longer remain anchored in the benchmark suites and problem formulations of the past. An industry-compatible open-source research infrastructure also serves as a two-way channel in which problem formulations and real (or, more realistic) testcases flow to academic researchers, while low-overhead technology transfers and assessments flow to industry.[7] (2) *True CAD-IP reuse.* As pointed out in [3] [36], (CAD

[2]Even if the authors of Paper_A and Paper_B both have proper access to the same foundry process design kits, the same standard-cell libraries and IPs, the same RTL, and the same licensed versions of commercial EDA tools, direct A-to-B communication of the EDA tool-related information needed for one group to reproduce the results of the other is prohibited. Note too that standard terms and names that are familiar to users of commercial EDA tools will be largely off-limits to creators of open-source EDA tools.

[3]Since commercial EDA tools are often required somewhere in the context for a research paper, various contortions (no-benchmarking-intended disclaimers, anonymized tool and vendor names, "extra" columns of data to prevent 1-1 mappings, etc.) are seen in published academic papers.

[4]A rumored open-sourcing of a foundry 130nm (even, 90nm) PDK in late 2019 will be an obvious watershed event for the open-source EDA ecosystem. The timeline for PDKs at other 'key' nodes (65nm, 28nm, 16/14nm, 7nm) will affect the rate of progress for digital IC tool research.

[5]Note 1: Here, the phrase "*not* open-source" encompasses "*not* sharing code or executable with other researchers, even privately, even for the limited purpose of comparison". Note 2: U.S. universities and federal research sponsors almost universally permit investigators to permissively open-source (per the accepted meaning of this phrase) their research products, entirely at the investigator's discretion. Note 3: In this list of reasons, (i) can be closely linked to (ii), i.e., the equivalencing of academic accomplishment with "numbers": number of publications in 'archival', peer-reviewed journals and conferences; number of citations as reported by Google Scholar; number of prizes at CAD contests; etc. In the past, the EDA research community has proactively reached out to academic administrators regarding credit assignment for conference papers versus journal papers. To mitigate (ii) and (i), it may be useful for the research community to propose credit assignment methodologies for open-source contributions.

[6]A given contest will focus on a specific, isolated core optimization and evaluation metric. However, this is orthogonal to the perpetuation of data models, formats and testcases that induce research codes which are unusable in the real world.

[7]Even when "industry-standard formats" are used, unrealistic settings (e.g., Liberty delay and power models) for contests risk driving academic researchers to conclusions that do not match reality. See, e.g., [7]. This highlights the value of having a robust two-way channel between academic research and industry practice.

software) IP reuse offers enormous productivity improvements in code development and the research process. Beyond being a "pay now or pay later" issue [15], the present lack of reusable CAD IP raises unnecessary barriers that force reinventing of wheels and make the field unattractive to new researchers. (3) *Removal of structural "irreproducibility" challenges.* The above-noted "irreproducibility by construction" barriers to research progress would not exist in an open-source EDA environment. (4) *Improved maturity and attractiveness of the field.* In recent years, open-source culture has been strongly associated with the rapid growth and success of other fields, e.g., AI / machine learning and computer vision. Arguably, the ease with which new researchers and practitioners can access leading-edge implementations has made these fields more attractive to new students and entrepreneurs alike. Established companies benefit from the rapid advance of core technologies and the larger pool of potential hires already trained up by an open-source ecosystem. Indeed, in these other fields the ethos of giving back (and/or, paying it forward) is palpable.[8]

C. Guidance from the Community

Recent forums such as the DAC-2018 and DAC-2019 "birds of a feather" meetings on Open-Source Academic EDA Software [34], and at workshops such as the Workshop on Open-Source EDA Technology (WOSET) [43] have elicited important messages and guidance from the broader community of open-source advocates, potential open-source EDA tool users, academic EDA researchers, and potential open-source donors/contributors. A starting goal of the OpenROAD project, namely, "critical mass and critical quality to seed a FOSS EDA ecosystem", was clearly reinforced. Toward such critical mass and critical quality, table stakes include (i) a full (RTL-to-GDS) tool flow with industry-standard inputs and outputs to match commercial practice; (ii) proper tool integration onto an incremental subtrate that includes an industry-compatible database (and, underlying data model) [24]; and (iii) a level of software engineering and EDA development infrastructure (building from source, unit tests, QA regressions, continuous integration / development framework, scripting extension language, etc.) that enables *and motivates* others to use and/or contribute. Also mandatory: (iv) proper open-source licensing [42] that enables tools to be freely used in both research and commercial settings. Last, great attention must be paid to (v) continual and positive engagement with numerous, fluid communities of stakeholders that span professional societies (ACM, IEEE), open software and hardware foundations (CHIPS Alliance, FOSSI, OpenHW Group), academic conference and contest organizers (DAC, ICCAD, ISPD, TAU, IWLS), funding sources (government, consortia, companies), etc.[9] The following section describes several ongoing or near-term steps that respond to this guidance and/or mitigate the root causes noted above.

[8]By contrast, the "no benchmarking" constraint in standard EDA industry EULAs has been well-noted for many years as a sign of an "immature" industry. Critics of the "no benchmarking" constraint ask how other product domains (automobiles, mobile phones, PCs, etc.) would look today if competitors were forbidden to buy and tear down each others' products. Defenders of the constraint usually note that "benchmarks can be skewed in unfair ways". At the same time, non-EDA industries appear to flourish in conjunction with public benchmarking via SPEC, AnTuTu, Fortnite, etc.

[9]Different geographies such as EMEA, East Asia and North America can have parallel efforts. As recounted in Section V of [18], past "research gap" analyses assume a 3× redundancy of effort due to regional effects. An open question is whether this sort of "parallel" redundancy is expected, or viable, in a future open-source EDA ecosystem and culture. (This said, "only" 3× loss of efficiency would arguably be a great improvement over the status quo.)

III. Unblocking: Near-Term Milestones

Four near-term, "unblocking" milestones for open-source EDA are (i) unified flow, (ii) shared netlist architecture, (iii) continuous build and integration, and (iv) generic node enablement.

A. Unified Flow

The IEEE CEDA Design Automation Technical Committee (DATC) [28] has over the past several years maintained a *Robust Design Flow* (RDF) based on winning tools from academic research contests. The RDF provides (i) an academic reference flow from logic synthesis to detailed routing based on existing contest results; (ii) a database of design benchmarks and point tool libraries; and (iii) a bridge between academic research and industry practice and designs via the use of industry-standard design input/output formats. At the same time, several opportunities for added research impact and industry engagement remained out of reach due to factors such as licensing, the "parallel universe" issue noted above, etc. As detailed in [4], tools from the OpenROAD project and other researchers have been added into the latest RDF-2019 release of the Robust Design Flow (see Figure 2).

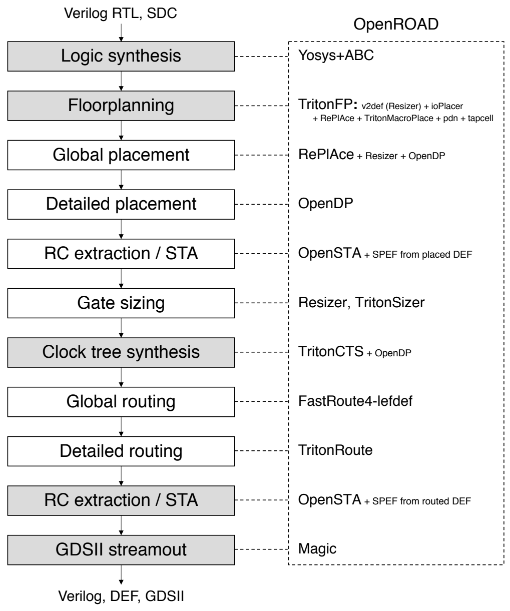

Fig. 2. The RDF-2019 flow, reproduced from [4].

Relative to the previous RDF release, horizontal extensions include adding numerous alternatives to existing tools, and vertical extensions include adding previously-missing steps such as floorplanning, I/O placement, power planning and clock tree synthesis. The result is a full academic flow from behavioral (RTL) Verilog to final detail-routed DEF. RDF-2019 provides many paths from Verilog to routed DEF, including paths that are entirely open-source and capable of delivering DRC-clean layout in a commercial (65nm) foundry enablement.

B. Shared Netlist Architecture

Figure 3, adapted from [24], illustrates the architecture as well as calls in the "common, incremental substrate" to which commercial EDA has converged in the RTL-to-GDS domain. The thin blue arrows illustrate the function or API calls and the thick blue arrow illustrates the overall flow from synthesis to routing.

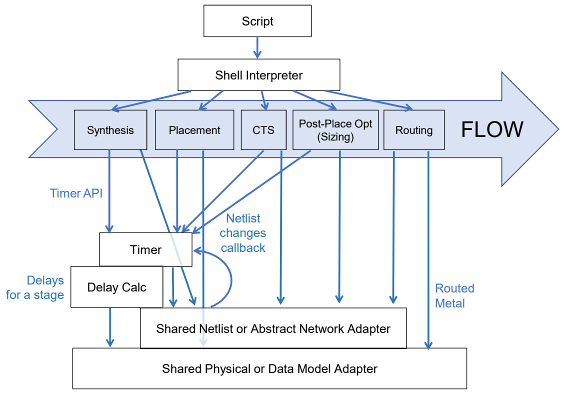

Fig. 3. Incremental shared netlist architecture.

By calling the Timer API, all of Synthesis, Placement, CTS, and Post-placement Optimization can be timing-driven. The **Shared Physical** or **Data Model** Adapter, which contains physical information such as placement location or routed metal shapes, can communicate with Synthesis, Placement, Routing, and the Timer. The **Shared Netlist** or **Abstract Network** Adapter, which enables incremental netlist modification, can communicate with Synthesis, Placement, CTS, Post-Place Opt, Routing, and the Timer. In commercial EDA tools, up to 10K incremental optimizations iterations per second can be supported by this architecture.[10]

C. Continuous Build and Integration

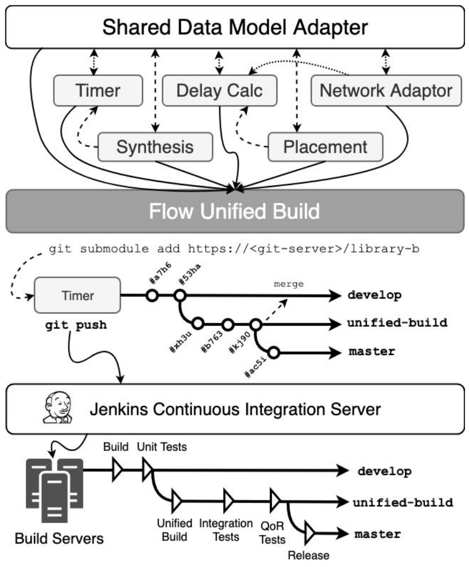

Fig. 4. A multi-module continuous integration pipeline for EDA flow.

Maintaining the integrity of an open source-based EDA flow requires a *continuous integration* (CI) pipeline that includes automated testing and packaging as part of the development life cycle. The CI pipeline is necessary for speed of development in addition to the stability of the code base. To reach this goal, automation is a key component in the CI infrastructure. When implemented using modular and independent open-source tools, the incremental architecture presented in Figure 3 can be achieved by adopting the approach that is illustrated in Figure 4 (see also [31]). Key aspects are as follows.

Flow Components. Each module resides in its own git-based repository and all modules can be built from sources. One module can include another using the *submodule* feature of the git protocol. The git submodule mechanism enables a tool to reference a specific version of another tool or a module without breaking the *flow unified build* if it uses another version. Here, the *flow unified build* is a separate repository unto itself that has the recipe for building the flow from the independent tools.

Git Branches. Adopting a branching scheme is important in maintaining the code base of all the tools, and subsequently the flow unified build. A variation of Git flow [38], a well-known scheme, is shown in the figure where all contributions are committed to the dev (development) branch. Changes that should be tested against the flow unified build can be checked out to an intermediate branch that represents the unified build. The master branch should always contain a clean and stable state of the code base. Hence, the master branch is merged to only when the unified build has been tested against the new changes.

Automation Server. Although there are many cloud-based services that can reduce the time to implement a CI pipeline, testing of EDA tools and flows sometimes requires private environments to run specific tests using private technology libraries. Therefore, at least within the OpenROAD project, employing a self-hosted Jenkins [39] installation is deemed essential. The Jenkins server monitors repositories for changes committed to any of the code base, and runs predefined steps in the automation pipeline. The pipeline builds the code and runs unit tests on every commit pushed to the dev branch. When commits are merged to the unified-build branch, flow integration tests and Quality of Results (QoR) tests (see Section IV-C below) are executed automatically and a full report is generated. If this stage succeeds, it automatically merges changes to the master branch. *Build servers* are where the actual execution of the pipeline steps occurs. Build servers might also hold private designs and technology libraries.

Community Contribution. The above-described branching scheme allows code base maintainers as well as outside contributors to build and contribute to the development in the same way – i.e., by pushing (or issuing pull requests) to the develop branch only. Direct commits to the unified-build or master branches are not allowed. In this way, the presented approach welcomes outside researchers to contribute directly to tools in the exact way that core maintainers do, thus leading to faster research-to-production turnaround times.

D. Generic Node Enablement

Achieving advanced-node, no-human-in-the-loop, design rule-clean layout is exceptionally challenging. This might be attributed to the past decades of “coevolution” of commercial EDA tools, cell libraries, IC design methodologies, and tool users’ expectations.[11]

[10]An open-source instantiation of this architecture, including an open-source physical design database and many aspects of a bona fide “EDA framework”, is anticipated in OpenROAD [31] by the time of this paper’s publication.

[11]Commercial EDA and silicon vendors enable ultimate product PPA to be achieved by large, expert design organizations over very long timelines for design enablement and physical implementation. Design rule manuals have thousands of rules; SP&R tools have thousands of command-option knobs; and last-mile, manual fixing of several hundreds or thousands of design rule violations just before tapeout is expected. Notably, today’s foundry PDKs and third-party IPs (standard-cell libraries, etc.) do not need to be consistent with “no-humans” EDA technology that guarantees design rule-clean layout.

Today, a given EDA provider spends enormous engineering effort to qualify its tools with respect to each individual commercial design enablement, without any of the "straight-out blockers" noted above. An open-source EDA ecosystem has neither the R&D resources, nor the access to proprietary foundry and third-party IP, to match this, and the "coevolved" context is not likely to change. On the other hand, projects such as OpenROAD aim for ultimate ease of use, not ultimate PPA; furthermore, care-abouts of software in academic research (e.g., transparency and modularity) are very different from those for commercial software (e.g., ultimate "fusions" and scalability in service of hyperoptimization). A crucial milestone for open-source EDA is to achieve a foundation of infrastructure and design enablement methodology that – as far as possible – "will work for any node". Such "generic" node enablement methodology is required given the R&D resource and IP access limits cited above.

To generically support even advanced technology nodes, an approach taken by the OpenROAD project is to impose simplified routing rules that serve to reduce "degree of difficulty" for academic open-source detailed routers. Under the proposed simplifications, the main requirements for the routing tool developer are (i) to enable strict unidirectional routing, and (ii) to enable on-grid and in-pin pin access. First, unidirectional routing makes design rules much simpler. Especially for double-patterning technology routing layers, judicious choices for only the end-to-end and side-to-side spacing rules (see Figure 5 (bottom)) can cover most of the foundry design rule set for routing layers. Furthermore, one of the most difficult challenges with advanced nodes is for tools to correctly understand advanced design rule syntax.[12] The goal of generic node enablement implies that we must avoid or otherwise subsume complex rules with LEF57 and LEF58 prefix seen in advanced-node technology LEF. Second, on-grid and in-pin access can simplify design rules related to pin access. Since this avoids extra shapes on pin layers (e.g., M1), it enables routing of designs without consideration for any rules between pin shapes and routing.

Initial assessments of a commercial 14nm enablement from a leading IP vendor show that existing commercial cell libraries and memory generators are likely to be usable by open-source place-and-route tools with only relatively minor restrictions. Such restrictions include, for example, setting "don't use" for cells with high pin density. Depending on the sophistication of the open-source routing tool, other help may be required, such as wrapping a "LEF-prime" veneer area around certain cells and macros to ensure straightforward on-grid pin access. Figure 5 (top) shows an example of such a veneer around an SRAM instance to ensure on-grid pins.

IV. Futures and Longer-Term Considerations

Open source must be welcoming and open, but cannot flourish as a chaotic "tower of Babel". Threading a path between openness and chaos is helped by existence of a strong "backplane" of industry-compatible database and underlying data model, along with well-engineered, well-architected and maintainable tools. Three of the near-term needs described in the previous section contribute to this "backplane". However, beyond near-term steps, many longer-term considerations must be kept in mind. This section offers comments on several of these.

[12]Today, design rules for a given advanced foundry node are encoded in completely different ways (technology LEF, SVRF, TF) by major EDA place-and-route tools.

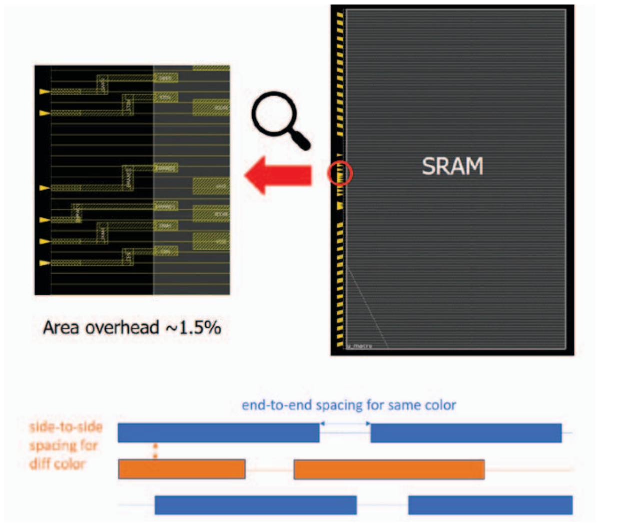
SRAM
Area overhead ~1.5%
end-to-end spacing for same color
side-to-side spacing for diff color

Fig. 5. Top: Illustration of SRAM "veneer" to achieve on-grid pins. Bottom: Simplified design rule set made possible by strict unidirectional routing for DPT layers.

A. Community and Ethos

An open-source software project critically depends on its *community* of developers and users.[13] The rise of internet-based software development tools have enabled highly distributed collaborations with others. To make it inviting to contribute to the project, the project must provide *value* to the individual contributor. A first major source of value is the *infrastructure* of the project. With viable EDA open source, the developer can start with an existing and tested data model, parsers and testcases which together provide much of the overhead that would be needed to start a project from scratch. This value is enhanced by the regression testing process, which protects the integrity of the code base. A second major source of value is that the open-source project provides a platform for publishing contributions. A basic tenet is that making a direct and tangible contribution to the project community leads to meaningful recognition. It is therefore important for the open-source EDA community to provide proper credit for contributions where due – and to help ensure that the credit is properly comprehended by academic promotion committees, R&D managers, funding agencies, etc.

Working in an open-source project requires adjustments from developers as well. To be valued contributors, developers will need to pay attention to the interests of the community. While it is certainly possible to make a copy of open source and "do your own thing with it" for one's own purposes, in the long term this is not a winning strategy [42]. Increasing divergence over time makes it more difficult to profit from contributions of others. Contributions will need to fit within the existing project and be a positive contribution, or at least not be disruptive.

B. Outreach and Alignments

Open-source tools, software infrastructure (build/CI, test/QA), and clear value delivered to open-source developers and users are at the start of a long journey. To achieve a vibrant, *self-sustaining* open-source EDA ecosystem, issues of growth, outreach and alignment must be strategically considered – sooner rather than later. A key issue is the support of stable, expert development resources; here, the

[13] [42] [9] provide additional commentary on open-source culture and behaviors.

substantial support that GNU, LLVM and other projects draw from user companies can be instructive. Should open-source EDA achieve greater maturity and capability, community needs could bifurcate into those of academic researchers and those of large IC design organizations; if this occurs, careful attention to and management of interests will be needed.

Within the traditional EDA community – which has little experience with open-source culture – recent outreach and alignment efforts have included the following. (1) Significant vertical and horizontal extensions of the IEEE CEDA DATC academic Robust Design Flow have been made, including the incorporation of several open-source tools from the OpenROAD project. This provides a skeletal RTL-to-GDS "full-flow" academic research backplane that can access industry designs and foundry enablement. As a result, RDF-2019 brings academic EDA research and industry design practice significantly closer together [4]. (2) Meetings such as WOSET [43] and DAC Birds-of-a-Feather [34] meetings have been organized. Such face-to-face events will likely be complemented by freely-accessible online seminars (e.g., [26] or those organized by [41]) and illustrative donated source code bases. (3) The ICCAD-2019 global routing contest [6] aims to address real-world challenges and, for the first time, directly encourage codes to be released under a *permissive* open-source license. Besides the core global routing algorithm, the contest is unique in setting "resource abstraction" in the to-do list for contestants, rather than providing pre-digested capacity and edge adjustments. This captures the need for a modern global router to achieve accurate resource modeling, with consideration of pin accesses, existing power delivery networks (PDNs), blockages and design rules – all from the raw inputs. Another unique aspect of the contest is its evaluation of entries using an academic detailed router, rather than comparing abstracted overflow metrics. A good global router must correlate well with the detailed router, since only metrics at the "achieved" end of the flow are meaningful in a real-world sense. (4) Another recent mechanism directly encourages the *community* itself to recognize meaningful contributions to open-source EDA. The Open-Source Community Contribution Awards [30], initiated in mid-2019, have a simple and open nomination mechanism and are overseen by an industry committee; the first award was made in August 2019 to the three undergraduates at UFRGS in Brazil who developed the FastRoute4-lefdef global router. Many more creative ideas and mechanisms toward "outreach" must emerge.

There is a clear prevailing zeitgeist of open-sourcing in systems, hardware, software and design tooling. The DARPA ERI, and its IDEA/POSH programs, are part of a wave that includes active and well-supported entities such as RISC-V, CHIPS Alliance, FOSSI, and OpenHW Group. These entities have brought reflected attention, if not a spotlight, to open-source EDA. The EDA community would do well to pay attention to the needs of these entities, as their memberships include many leading semiconductor and system companies who know EDA well and have been strong supporters of production-quality open-source development.

C. Challenges of EDA Regression Testing

Section III-C above reviewed the architecture of CI/CD (continuous integration / continuous delivery) [23] as implemented in the OpenROAD project. This provides developers with a stable code base to develop new features from, and provides users with stable code to use. However, "stable" is not the same as "bug-free". Bug-free code is an ideal that may be approximated but can never be reached. Stable code, on the other hand, is good enough so that people can work on real issues and improvements, rather than deal with fixing things that used to work. In this context, it is clear that *regression testing* plays a crucial role in the EDA development and integration process.[14]

While not specific to open-source EDA, it is worth noting the special concern of *instability* that arises in regression testing for EDA *optimization* (i.e., 'synthesis' or 'creation') heuristics. Trevillyan [25] refers to such heuristic optimization methods as "automated design algorithms"; such an algorithm is termed *unstable* when a small change in the input can cause a large change in the output. (This phenomenon has been referred to as "noise" or "chaos" in other works [21] [10] [19].) For decades, EDA technology in well-studied problem domains such as logic optimization or place-and-route has been dominated by heuristics that exhibit chaotic behaviors.[15]

Chaotic tool behavior poses special challenges for regression testing. While in most software projects a regression test checks for correct versus incorrect behavior, in automated design algorithms a wide variety of solutions can be considered correct. For example, any routing solution that connects all of wires correctly without overlap can be considered a correct solution to the routing problem. Various different solutions can be evaluated by quality metrics – e.g., different routing solutions can be evaluated by total wirelength. So, a regression test for the router which routes an input design would set an upper bound for the acceptable wirelength. When the wirelength exceeds the upper bound, the regression test fails.

While a regression test can work around the problem of algorithmic instability by allowing a range of solutions, such an approach is not problem-free. If the upper bound in the regression test is set tightly, the regression test may fail occasionally due to "noise". Mere random fluctuations in the wirelength quality metric may cause the test to fail even when no real bug or error exists in the code. Thus, debugging the failed regression may be difficult, as it may be difficult to determine whether a bug is actually present. On the other hand, setting a loose upper bound minimizes spurious regression failures but suffers a different problem: small changes in the optimization quality may not be detected, and the optimization heuristic may slowly deteriorate over time without triggering a regression failure. When finally detected, several problems may have crept in over time without being detected, and the latest change may not be the actual source of the regression failure. These challenges of QoR testing at both flow and tool levels will likely require significant management oversight and QA/developer effort in the open-source context.

D. Measuring Progress: Roadmapping of Open-Source EDA

Open-source EDA is synonymous with open-source *design technology*. In this light, the over 25 years of *technology roadmapping* in the semiconductor industry, including over 15 years of the ITRS roadmap [40], can guide the roadmapping of open-source EDA. Among other functions, an impactful roadmap must (i) set out difficult challenges,

[14]Regression testing consists of applying testcases that were known to be successfully handled by the software. Having sufficient coverage by such testcases ensures that most features that work will continue to work. In the context of the release process, the regression testing assures that releases meet a certain quality threshold.

[15]As an example, consider the case of a router. Most routers route wires one by one, starting with an empty routing area and avoiding existing wires as they are added. Even if each wire can be optimally routed, the final result will be dependent on the order of the wires. If one starts with a different wire, the routes of the subsequent wires will be different and the overall solution will be different. [19] and Section 2 of [16] point out that this chaos is especially apparent when complex heuristics are pushed to their solution quality limits. When different types of optimization are executed in sequence in a flow, the problem is compounded. And, unfortunately, not only changes to the input design but also changes to the software or even changes in the runtime environment can cause this type of behavior.

potential solutions, and a mapping between challenges and solutions, at both near- and long-term time horizons; (ii) establish required trajectories of *quantified metrics* that meaningfully capture progress of the field; and (iii) engage and unite with common purpose a broad community of stakeholders (e.g., industrial R&D, academic researchers, government/consortia, users, entrepreneurs and investors).

Illustration of Roadmap Styles for Open-Source Design Technology Metrics

Metric	*2019*	*2022*	*2025*	*2028*	*2031*	*2034*
#Open-source EDA repositories > **threshold** (**relative** growth from 2019)	1.0	1.5	2.0	2.5	3.0	3.0
#Tapeouts/yr (**academic entity**, sub-40nm) w/full open-source EDA flow	0	6	20	🟨 40	🟨 60	🟨 ≥ 100
#Tapeouts/yr (**commercial entity**, sub-40nm) w/full open-source EDA flow	0	🟨 4	🟨 10	🟥 ≥ 20	🟥 ≥ 30	🟥 ≥ 40
#Companies based on open-source EDA (tools + services + IC products) (>3 employees)	4	6	🟨 15	🟨 30	🟨 50	🟨 ≥ 70
%relevant EDA papers citing open-source EDA repositories	8	15	25	30	≥ 35	≥ 35
Lowest achieved **PPA metric ratio** vs. a best known implementation (**N-4** node)	10 : 1	🟨 2 : 1	🟥 1.5 : 1	🟥 1.4 : 1	🟥 1.2 : 1	🟥 1.1 : 1

⬜	Solutions Known; Solved Problem (future trajectory is viable and expected)
🟨	Solutions Known; Refinement Needed (dependence on community, market, etc.)
🟥	No Known Solution (one or more fundamental blockers must be overcome)

Fig. 6. Illustration of roadmap styles for open-source design technology metrics.

It is difficult to precisely define metrics that can be used over a decade or more to measure progress of a given technology. Figure 6 illustrates various flavors of open-source EDA metrics that might exist within the classic 15-year, white-yellow-red roadmap table style seen in the ITRS. The first row illustrates a *normalized* or *relative* metric, for which the definition of *threshold* (e.g., as a function of contributors, commits, forks, pull requests, documented usage, ...) is crucial. The second through fourth rows illustrate *numerically-quantified* metrics, along with the kind of specificity (commercial entity, technology node, market sector, headcount, ...) that enables metric evaluation. These rows also illustrate the yellow-red coloring convention of roadmap targets to indicate, e.g., the dependence of commercial product tapeouts on emergence of open-source industry structures (cf. Section 5 of [2]). The fifth row again highlights the critical nature of a metric's definition - in this case, with respect to the universe of "relevant" papers. The sixth row, "PPA metric ratio", illustrates a metric with *bounded value* – in this case, bounded from below, by 1.

The use of devices such as "$N-4$" (i.e., "four nodes behind the leading-edge node") in the "PPA metric ratio" example helps enable longitudinal assessment of progress without referring to a specific technology node. To avoid references to specific tools, algorithms or testcases, PROBE [17] gives a testcase-insensitive methodology for comparing design enablement elements ranging from placement and routing tools to back-end-of-line (metal layer) stacks. The so-called K_{th} criterion [17] defines a continuum of layout problem difficulty; tools can be compared by, e.g., the K_{th} value at which the number of design rule violations exceeds a given threshold. Figure 7 shows how K_{th} could potentially expose and quantify the difference between OpenROAD and commercial capabilities for placement and for routing.

E. Long-Term Challenges and Potential Solutions

Many long-term challenges, along with their potential solutions, require thoughtful roadmapping over at least a 5-10 year horizon. Examples of such challenges (several touched on above) are as follows. **(1)** With the very notable exception of the commercial static timing analysis engine OpenSTA [37], open-source EDA has not yet made significant progress toward **signoff-quality verifications** such

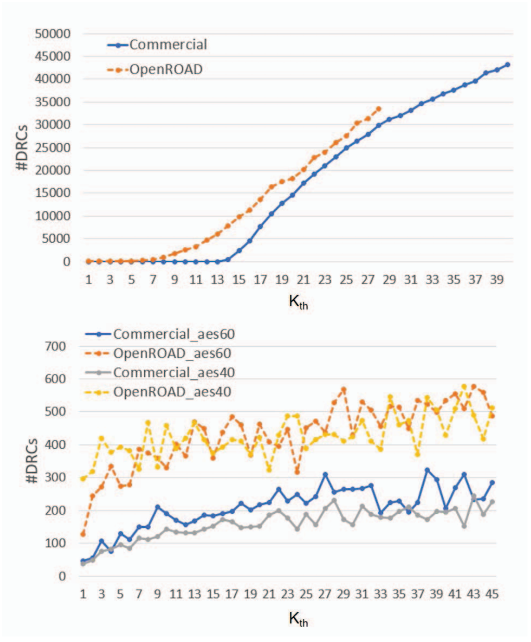

Fig. 7. (a) PROBE-based comparison of OpenROAD placement and a leading commercial placer. The name "aes60" indicates testcase *aes* with 60% initial utilization. (b) OpenROAD's routing tool begins to show significant #DRCs at $K_{th} = 9$, and cannot produce any solution when $K_{th} > 28$.

as timing and power analyses, or DRC/ERC/LVS checkers. Section II-A noted the structural barrier of foundry qualification. More daunting is the sheer difficulty of mastering this EDA technology, particularly for advanced-node design requirements. There is likely no shortcut for well-resourced development by EDA veterans over a multi-year span. **(2)** For current open-source projects, the **non-commercial status of academic research labs** remains a barrier to the traditional bug-fix and support interactions between EDA provider and user. Here, the aegis of open-source foundations, and/or secure cloud-based EDA infrastructures pioneered by commercial EDA in partnership with foundry/IP and customers, and/or emerging IP-preserving cryptographic protocols, may provide solutions. Perhaps more critically, academic researchers often lack training and motivation to produce and support production-quality tools. Section IV-B noted the possibility of a future bifurcation of open-source EDA as academic research needs and commercial IC design needs diverge. **(3)** The IC design ecosystem increasingly looks to bf enablement of machine learning for future cost and schedule savings in IC design [16] [32]. Such savings are among the "last scaling levers" in a late- or post-Moore's-Law scaling era. Since open-source EDA tools are transparent with regard to "what the tool is thinking", machine learning in the context of open-source tools and designs can advance more rapidly than in the typical commercial (closed-source tools, arms-length insight into customer designs) context. Standardized metrics naming, enablement of flow-scale (reinforcement) learning, federated learning and privacy-preserving modeling, model sharing mechanisms, and many other needs are reviewed in various talks linked from https://vlsicad.ucsd.edu/. **(4) Viable business and support models** ("RedHat of FOSS EDA", GPL open-sourcing plus consulting services, funding of development via an open-source foundation, etc.) are needed. As noted in Section IV-B, potential alignments to open hardware organizations may provide a path forward, while also helping with growth and support of a **healthy ecosystem** for open-source EDA. **(5)** Open-source EDA should **accelerate growth paths for commercial EDA**. With critical quality and critical mass including elements described in Section III above, open-source innovation should see faster research-to-production pathways in the existing scope of EDA. Open-source EDA also has great potential to boost the quality of research tools for nascent or niche "XDA" challenges (design automation for X = monolithic 3D VLSI, microfluidic labs-on-chip, next-generation FPGA P&R, etc.) that commercial vendors have not yet addressed. A long-term roadmap of "XDA" targets (cf. [44]) may provide useful direction to the community.

V. Conclusions

Over 40 years ago, permissively open-sourced academic tools helped seed the field of electronic design automation as we know it. Since that time, our field has endured decades of blame for cost, quality and productivity shortfalls in the IC design ecosystem. While research in EDA has always been very challenging, it has also been made unnecessarily difficult by a number of "straight-out blockers", as well as a systemic lack of reusable research artifacts, research infrastructure, and bridges to industrial practice. At the same time, a wave of open-source has swept across software, hardware and systems – every adjacency of electronic design automation – visibly accelerating growth and innovation.

As stated in Section II-A above, "whether to open-source is always a 100% personal decision". Open source affords the EDA community a means of paying it forward, sharing wisdoms accumulated over entire careers, and attracting and teaching a next generation of EDA researchers and technologists. Open source is not a threat, but rather a complement and boost, to commercial EDA. At a personal level, it is liberating. For a community that has the strength of good intentions and a shared vision, it is absolutely doable. Let's not be afraid.

Acknowledgments

The author thanks many past and current students and collaborators, including members of the OpenROAD project, who have helped to develop the works and perspectives given here. Many thanks are due to Lukas van Ginneken, Tom Spyrou and Abdelrahman Hosny for providing substantial inputs, including figures and text, for the discussions of testing, incremental tool architecture, and CI/repository organization. And, many thanks to Pierre-Emmanuel Gaillardon for inviting this contribution. Research at UCSD is supported by Qualcomm, Samsung, NXP Semiconductors, Mentor Graphics, DARPA (HR0011-18-2-0032), NSF (CCF-1564302), and the C-DEN center.

References

[1] T. Ajayi, D. Blaauw, T.-B. Chan, C.-K. Cheng, V. A. Chhabria, D. K. Choo, M. Coltella, S. Dobre, R. Dreslinski, M. Fogaça, S. Hashemi, A. Hosny, . B. Kahng, M. Kim, J. Li, Z. Liang, U. Mallappa, P. Penzes, G. Pradipta, S. Reda, A. Rovinski, K. Samadi, S. S. Sapatnekar, L. Saul, C. Sechen, V. Srinivas, W. Swartz, D. Sylvester, D. Urquhart, L. Wang, M. Woo and B. Xu, "OpenROAD: Toward a Self-Driving, Open-Source Digital Layout Implementation Tool Chain", *Proc. GOMACTECH*, 2019, pp. 1105-1110.

[2] T. Ajayi, V. A. Chhabria, M. Fogaça, S. Hashemi, A. Hosny, A. B. Kahng, M. Kim, J. Lee, U. Mallappa, M. Neseem, G. Pradipta, S. Reda, M. Saligane, S. S. Sapatnekar, C. Sechen, M. Shalan, W. Swartz, L. Wang, Z. Wang, M. Woo and B. Xu, "Toward an Open-Source Digital Flow: First Learnings from the OpenROAD Project", *Proc. DAC*, 2019, pp. 76:1-76:4.

[3] A. E. Caldwell, A. B. Kahng and I. L. Markov, "Toward CAD-IP Reuse: The MARCO GSRC Bookshelf of Fundamental CAD Algorithms", *IEEE Design and Test of Computers* 19(3) (2002), pp. 70-79.

[4] J. Chen, I. H.-R. Jiang, J. Jung, A. B. Kahng, V. N. Kravets, Y.-L. Li, S.-T. Lin and M. Woo, "DATC RDF: Towards a Complete Reference Flow", *Proc. ICCAD*, 2019, to appear.

[5] R. Cox, "Surviving Software Dependencies", *Comm. of the ACM* 62(9) (2019), pp. 36-43.

[6] S. Dolgov, A. Volkov, L. Wang and B. Xu, "2019 CAD Contest: LEF/DEF Based Global Routing", *Proc. ICCAD*, 2019, to appear.

[7] H. Fatemi, A. B. Kahng, H. Lee, J. Li and J. Pineda de Gyvez, "Enhancing Sensitivity-Based Power Reduction for an Industry IC Design Context", *Integration, the VLSI Journal* (2019), doi:10.1016/j.vlsi.2019.01.008

[8] S. Fenstermaker, D. George, A. B. Kahng, S. Mantik and B. Thielges, "METRICS: A System Architecture for Design Process Optimization", *Proc. DAC*, 2000, pp. 705-710.

[9] T.-W. Huang, C.-X. Lin, G. Guo and M. D. F. Wong, "Essential Building Blocks for Creating an Open-source EDA Project", *Proc. DAC*, 2019, pp. 78:1-78:4, https://tsung-wei-huang.github.io/talk/dac19-invited.pdf.

[10] K. Jeong and A. B. Kahng, "Methodology From Chaos in IC Implementation", *Proc. ISQED*, 2010, pp. 885-892.

[11] J. Jung, I. H.-R. Jiang, J. Chen, S.-T. Lin, Y.-L. Li, V. N. Kravets and G.-J. Nam, "DATC RDF: An Academic Flow from Logic Synthesis to Detailed Routing", *Proc. ICCAD*, 2018, pp. 37:1-37:4

[12] J. Jung, I. H.-R. Jiang, J. Chen, S.-T. Lin, Y.-L. Li, V. N. Kravets and G.-J. Nam, "DATC RDF: An Open Design Flow from Logic Synthesis to Detailed Routing", *Proc. WOSET*, 2018, pp. 6:1-6:4.

[13] J. Jung, I. H.-R. Jiang, G.-J. Nam, V. N. Kravets, L. Behjat and Y.-L. Li, "OpenDesign Flow Database: the Infrastructure for VLSI Design and Design Automation Research", *Proc. ICCAD*, 2016, pp. 42:1-42:6

[14] J. Jung, P.-Y. Lee, Y.-S. Wu, N. K. Darav, I. H.-R. Jiang, V. N. Kravets, L. Behjat, Y.-L. Li and G.-J. Nam, "DATC RDF: Robust Design Flow Database", *Proc. ICCAD*, 2017, pp. 872-873.

[15] A. B. Kahng, "CAD Research, Pay Now or Pay Later...", *ICCAD Monday Evening Panel*, 2006. https://vlsicad.ucsd.edu/Presentations/talk/ICCADPanel-abk-v3.ppt

[16] A. B. Kahng, "Reducing Time and Effort in IC Implementation: A Roadmap of Challenges and Solutions", *Proc. DAC*, 2018, pp. 36:1-36:6.

[17] A. Kahng, A. B. Kahng, H. Lee and J. Li, "PROBE: A Placement, Routing, Back-End-of-Line Measurement Utility", *IEEE Trans. on CAD* 37(7) (2018), pp. 1459-1472.

[18] A. B. Kahng and F. Koushanfar, "Evolving EDA Beyond its E-Roots: An Overview", *Proc. ICCAD*, 2015, pp. 247-254.

[19] A. B. Kahng, S. Kumar and T. Shah, "A No-Human-in-the-Loop Methodology Toward Optimal Utilization of EDA Tools and Flows", *unpublished manuscript*, 2017, linked from https://vlsicad.ucsd.edu/MAB/.

[20] A. B. Kahng, M. Luo, G.-J. Nam, S. Nath, D. Z. Pan and G. Robins, "Toward Metrics of Design Automation Research Impact", *Proc. ICCAD*, 2015, pp. 263-270.

[21] A. Kahng and S. Mantik, "Measurement of Inherent Noise in EDA Tools", *Proc. ISQED*, March 2002, pp. 206-211.

[22] H. A. Landman, "Visualizing the Behavior of Logic Synthesis Algorithms", *SNUG San Jose*, 1998.

[23] R. Potvin and J. Levenberg, "Why Google Stores Billions of Lines of Code in a Single Repository", *Comm. of the ACM* 59(7) (2016), pp. 78-87.

[24] T. Spyrou, "Open-Source EDA Challenges and Architecture", opening talk at DAC-2019 Open-Source Academic EDA Software Birds-of-a-Feather meeting.

[25] L. Trevillyan, "An Overview of Logic Synthesis Systems" *Proc. DAC*, 1987, pp. 166-172.

[26] L. van Ginneken, "Regression Testing & Quality Assurance", *online seminar*, August 1, 2019. Video and slides at https://youtu.be/LhssH9Qx9Lc.

[27] DARPA IDEA, https://www.darpa.mil/program/intelligent-design-of-electronic-assets

[28] DATC, https://ieee-ceda.org/node/2591 and https://github.com/ieee-ceda-datc/RDF2019

[29] OpenROAD, https://theopenroadproject.org/

[30] Open-Source Community Contribution Awards Nomination Form, https://theopenroadproject.org/openroad_event/oscca/

[31] OpenROAD GitHub, https://github.com/The-OpenROAD-Project (see also https://github.com/The-OpenROAD-Project/alpha-release).

[32] The METRICS Initiative, https://vlsicad.ucsd.edu/GSRC/metrics/

[33] Yosys Open SYnthesis Suite, http://www.clifford.at/yosys/

[34] DAC 2019 Birds-of-a-Feather Meeting: Open-Source Academic EDA Software Continued, https://github.com/The-OpenROAD-Project/Birds-of-a-Feather-Open-Source-Academic-EDA-Software/wiki/DAC-2019-Birds-of-a-Feather:-Open-Source-Academic-EDA-Software

[35] Census of Open-Source Academic EDA Software, https://github.com/abk-openroad/Academic-Open-Source-Tool-and-Contest-Winners/wiki/Academic-Open-Source-Tool-and-Contest-Winner-Stats

[36] (MARCO GSRC) VLSI CAD Bookshelf. http://vlsicad.eecs.umich.edu/BK/

[37] OpenSTA, https://github.com/The-OpenROAD-Project/OpenSTA/

[38] Git branching model, https://nvie.com/posts/a-successful-git-branching-model/

[39] Jenkins website, https://jenkins.io/

[40] *International Technology Roadmap for Semiconductors* (multiple year reports of Design, System Drivers, and System Integration (ITRS2.0) chapters), http://www.itrs2.net/itrs-reports.html.

[41] VLSI System Design website, https://www.vlsisystemdesign.com/

[42] T. Ansell, "FOSS 101" and "FOSS 102" presentations, July 2019. j.mp/eri19-foss101 and j.mp/eri19-foss102.

[43] Workshop on Open-Source EDA Technology (WOSET), http://scale.engin.brown.edu/woset/.

[44] IEEE CEDA Design Automation Futures Workshop, October 2016. https://ieee-ceda.org/event/design-automation-futures-workshop-2016-dafw