Solvers, Engines, Tools and Flows: The Next Wave for AI/ML in Physical Design

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

ABSTRACT

It has been six years since an ISPD-2018 invited talk on "Machine Learning Applications in Physical Design" [25]. Since then, despite considerable activity across both academia and industry, many R&D targets remain open. At the same time, there is now clearer understanding of where AI/ML can and cannot (yet) move the needle in physical design, as well as some of the difficult blockers and technical challenges that lie ahead. Some futures for AI/ML-boosted physical design are visible across solvers, engines, tools and flows – and in contexts that span generative AI, the modeling of "magic" handoffs at flow interstices, academic research infrastructure, and the culture of benchmarking and open-source EDA.

CCS CONCEPTS

• **Hardware** → **Physical design (EDA)**; *Software tools for EDA.*

KEYWORDS

Artificial Intelligence, Machine Learning for EDA

ACM Reference Format:
Andrew B. Kahng. 2024. Solvers, Engines, Tools and Flows: The Next Wave for AI/ML in Physical Design . In *Proceedings of the 2024 International Symposium on Physical Design (ISPD '24), March 12–15, 2024, Taipei, Taiwan.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3626184.3635277

1 INTRODUCTION

The ISPD-2018 invited paper [25] is one in a trajectory of works proposing how AI/ML would change physical design.[1] Six years after [25], we see that (i) more than half of the research papers at leading EDA conferences now involve applications of machine learning; (ii) AI/ML pervades the product offerings of major EDA vendors; and (iii) the latest wave of large language models and generative AI is everywhere. With this backdrop, this paper gives some personal thoughts on near-term futures for AI/ML in physical design through the lens of "solvers, engines, tools and flows".

1.1 Looking Back to 2018.

In 2018, a future ecosystem (Figure 1) was envisioned wherein EDA suppliers would open up their tool knobs, thus enabling new AI/ML models for tools and designs, alongside discovery of new optimization objectives for core EDA engines. Designers would identify pain points and supply training and validation data to researchers, who would develop new design-adaptive tool and flow guidance, along with tool outcome predictors. Improved tools and private-label AI models would then increase the value of commercial design technology to users. Unfortunately, the envisioned ecosystem did not materialize, for reasons that include the lack of (i) accessible EDA tool knobs, and (ii) solutions to the data problem. Hence, many "target list" items from six years ago [24–26] are still open.

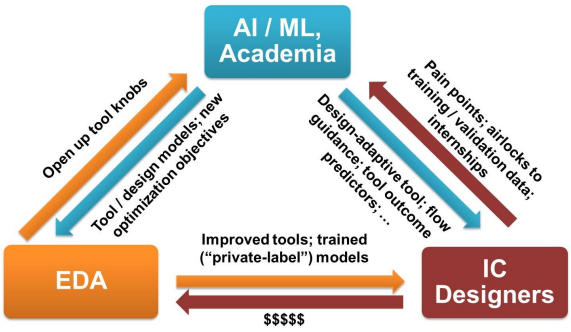

Figure 1: 2018 vision of a future Vendor-Designer-Researcher ecosystem for AI/ML in EDA. From [24].

Also in 2018, Figure 2 from [25] proposed four oncoming stages of ML insertion into PD tooling. ML developments since then reflect this path, and are typically grouped into three main categories: *prediction*, *optimization* and *generation* [52]. *Prediction* applies supervised learning to predict design QoR metrics [32], with many endeavors spanning power prediction [49], timing prediction [16], congestion prediction [54], etc. *Optimization* applies Bayesian optimization (BO) or reinforcement learning (RL) to directly optimize EDA problems. *AutoDMP* [1] wraps BO around DREAMPlace [43] to achieve superior macro placement solutions. The earlier work

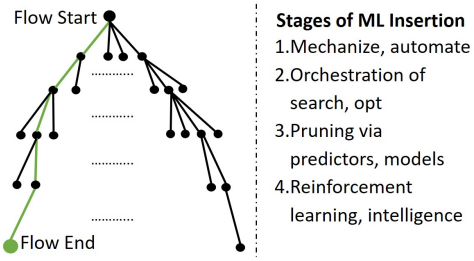

Figure 2: ML insertion for PD optimization. From [25].

[1] [25] is complemented by works such as [24] [26] [27] [28] [29] [30].

ISPD '24, March 12–15, 2024, Taipei, Taiwan

ACM ISBN 979-8-4007-0417-8/24/03
https://doi.org/10.1145/3626184.3635277

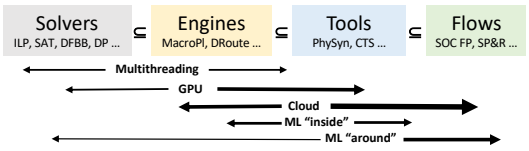

Figure 3: Impact of multithreading and GPU, cloud deployment, and AI/ML across the continuum of solvers, engines, tools and flows.

[50] applies RL to macro placement. *Generation* applies generative models (GANs or transformers) to directly generate solutions to EDA problems, e.g., [57] uses GANs to synthesize realistic layout patterns, and [51] develops a transformer-based gate sizer that optimizes a given placed and unoptimized netlist. Generative approaches struggle with consistent delivery of well-formed solutions, hance are often used to obtain good initial (ballpark, or hint) solutions for traditional methods. At the same time, the rapid advance in areas such as Large Language Models (LLMs) bring hopes of widespread application in EDA and design [17].

Overall, the past six years have brought clearer understanding of where AI/ML can and cannot (yet) move the needle, and of near-term challenges. Some takeaways:

- *Successes* include black-box hyperparameter optimization (e.g., Cadence Cerebrus and Synopsys DSO.ai), i.e., a form of AI/ML "around" the tool. ML has also shown promise to speed up analyses that range from routability and congestion to EM/IR and temperature.
- *Disappointments* include the increasing difficulty of extracting runtime data or applying ML inference results, in the regime of an industry duopoly and closed tool "silos". Prospects for companies sharing data, or public foundation models, are poor. Further, costs of machine learning have turned out to be significant. And, prediction is difficult.
- *Surprises* include the rush to LLMs and generative AI, which aligns with reduced visibility of users into tools.

1.2 A Lens of Solvers, Engines, Tools and Flows.

Figure 3 depicts a hierarchy of design technology elements, along with the applicability of various boosters such as cloud deployment or ML "inside".

- *Solvers* at the base of the hierarchy typically correspond to combinatorial methods (ILP, SAT, DP), often with performance bounds. An example application is clip-based, optimal place-and-route [8, 20]. Multithreading, GPU and cloud can slightly advance the size of solvable instances, but ML (e.g., RL or L2O) has not had impact in practice.
- *Engines* such as macro placers or global routers typically integrate solvers with metaheuristics (hill-climbing, evolutionary optimization, multi-start). Multithreading, GPU and cloud can all improve engine runtime and scalability.
- *Tools* typically orchestrate multiple engines to perform entire stages (e.g., physical synthesis) of the physical implementation flow, with extensive scripting interfaces. Outcomes can vary substantially with random seeds and initial conditions; cloud deployment enables multi-start and sampling-based approaches (e.g., black-box hyperparameter autotuning [21, 37]) that improve quality and stability of outcomes obtained within schedule limits.
- *Flows* such as SoC floorplanning invoke multiple tools, often with feedback loops, to achieve major design goals. Inherent complexity, long runtimes and chaotic behaviors make flows resistant to techniques such as reinforcement learning. On the other hand, improved results can be obtained via sampling and autotuning with cloud resources.

It is important to consider *where* ML can be applied – "inside" or "around" [27]. Figure 3 indicates that ML "around" has applications across solvers through flows, e.g., by autotuning of hyperparameters, commands and options. ML "inside" has been less tractable, absent infrastructure that can save and leverage data from past design experiences or tool/flow runs. Such infrastructure has been suggested for decades [12, 21, 31], but is blocked by a missing consensus on open standards for data models, naming and APIs.

It is also crucial to comprehend the breadth of challenges and limits seen today for ML in PD: optimization quality of results; data; scalability; generalization; validations; and cost. Three of these stand out: (1) *Optimization baselines are strong*. ML has struggled to show benefit for problems that have well-studied formalisms and combinatorial methods. Metaheuristics such as annealing or genetic algorithms also give strong baselines. And, humans still come up with the most useful cost functions. (2) *ML has not handled scale well.* For example, RL and L2O have proved too data-heavy relative to design complexities and resource limits. This heightens interest in problem decomposition and discovery of useful hierarchy, especially at the front end of chip implementation. (3) *Chaos is still with us.* E.g., perturbing target clock period by 0.1ps may change post-P&R netlist area by 10%. Thus, sampling remains a workhorse for improved solution quality and stability, with little involvement of ML. (4) *Accuracy bars are high* for ML-boosted PD to have value in production.

2 PD CHALLENGES AND LEVERS

This section briefly reviews selected challenges and key levers for PD (see also [25, 28]).

2.1 PD Challenges.

Floorplanning and placement have become severe pain points in PD, due to rapid growth of SoC design complexity in concert with the move to advanced nodes. Some emerging challenges are as follows.[2]

Design partitioning and block shaping. Growing design complexity has collided head-on with superlinear tool runtimes and tighter design schedules. Designers must partition huge designs into small (e.g., 1-2 million instances) blocks. The coevolution of a P&R tool with its typically small-sized R&D regression and customer benchmark testcases means that the tool ends up being highly tuned for small blocks. On the other hand, having more small blocks in the SoC floorplan creates an explosion of complex rectilinear block shaping, pin/port placement, macro placement (or partitioning), channel definition, and floorplan brittleness. In the end, whether to split a design into a smaller number of large blocks with long runtimes, or into a larger number of small blocks with short runtimes, depends on the P&R tool as well as the design. Corollary challenges

[2]These challenges involve "co-optimizations" and/or "coevolutions" at *interstices* of the PD flow (see Section 3.2), and bring high-value applications for AI/ML in PD.

include budgeting, assembly, signoff, and even the movement to 2.5D/3D.

Placement-aware hierarchical floorplanning. Placement-aware (or macro placement-aware) hierarchical floorplanning must also make decisions based on some underlying model of the placement tool's behavior. The placer is exquisitely sensitive to macro locations, block shaping and boundary terminals, and floorplan constraints (fences, regions, guides, (hard/partial/FF) blockages, etc.). Applying the right floorplan constraints can condition the placement canvas and steer the tool toward dramatic improvements of solution quality metrics such as timing or congestion. However, robust methods to synthesize such "magic" floorplans and screens remain elusive [25].

Datapath-aware floorplanning. In many application markets, high-performance datapath modules are essential to competitive success, and layout solutions may be replicated thousands of times in tiled architectures. As had preceded a 1990s wave of datapath-focused EDA offerings [28], today's tools often produce unacceptable misalignment and tangling of datapaths in P&R. Datapath-aware floorplanning entails intelligent pre-placement and fixing of PPA-critical flip-flops and standard cells – after synthesis and before placement – again, to properly condition the placement canvas and steer the tool to good results. How to select and place the fixed instances in the datapath-aware floorplan has also been elusive.

Drive for area reduction. With the slowdown of scaling and increasing wafer cost per transistor, die utilization is now a permanent first-class concern. Cost reductions are sought by squeezing whitespace out of the SoC floorplan, but this is challenging since (i) block shapes interact in the SoC floorplan, and (ii) chaos in the implementation flow generally worsens with more aggressive PPA targets (cf. "Sampling for Stability" in the next subsection). Corollaries of area squeezing include adoption of hybrid cell row architectures at the foundry 3nm node [28, 67] to enable flexible mixing of drive strengths (i.e., fin counts). Use of hybrid row heights (e.g., 117, 169, 286nm in a single P&R block) severely challenges the entire PD flow, from physical synthesis that must predict and manage density distributions of cell heights, to sizing and ECO optimizations whose cost landscapes become less smooth.

2.2 Levers to Advance PD.

Four high-impact levers to advance PD are (i) GPU-based speedups; (ii) use of cloud resources; (iii) use of sampling to gain stability; and (iv) a "multi-spectral" mindset to obtain more data for ML in a given time.

GPU-based speedups. Advanced SoCs can contain up to billions of instances and nets. However, the runtime for P&R and optimization can be several days even for a small 2 million-instance block. Design schedule constraints therefore induce complex design partitioning and block shaping challenges. In this context, massively parallel modern GPUs provide a natural computational substrate for acceleration of physical design [19, 42, 56]. Table 1 lists efforts in this direction that span much of the PD flow.

A key goal for GPU-based acceleration of optimization and ML is to shorten the time to useful PPA feedback, thus enabling faster and more accurate exploration of architecture and chip floorplan

Table 1: Examples of GPU-accelerated physical design.

Physical Design Flow	Related works
RTL Simulation	RTLFlow [39]
Logic Synthesis	CULS [48]
Macro Placer	AutoDMP [1]
Global Placement	DREAMPlace [43], Xplace [45]
Detailed Placement	ABCDPlace [44]
Global Routing	GAMER [40], FastGR [47], GGR [41]
DRC Checker	OpenDRC [18]
STA	[14], [15]

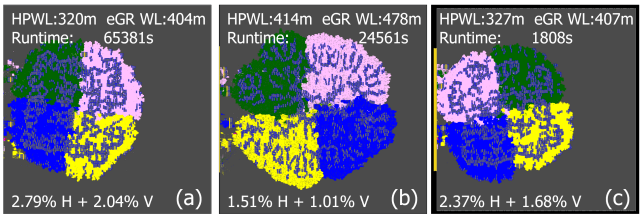

Figure 4: Placements from (a) RePlAce, (b) Commercial placer, and (c) a new academic GPU-accelerated placer.

options. At the same time, classical electrostatics-based global placement engines such as OpenROAD's RePlAce [9, 63] can miss design-level perspective. Figures 4(a), (b), and (c) respectively show global placement results of RePlAce, a commercial placement tool, and a new GPU-accelerated academic tool for the MemPool Cluster RISC-V design [5], which has 9.5M instances and nearly 1300 macros in NanGate45 technology. The new placer substantially reduces runtime while maintaining solution quality in terms of post-placement half-perimeter wirelengths, along with early global route (eGR) wirelength and congestion metrics reported by Cadence Innovus 21.1. Advances in speed, quality and scalability from GPU-based acceleration can potentially revolutionize architecture, RTL and floorplan exploration – orthogonally to improvements in ML and search. Of course, development must focus on where slowdowns arise in real end-to-end flows, e.g., detailed route search and repair, DRC fixing, and MCMM design closure.

Cloud. Another lever for PD is the mainstreaming of EDA in the cloud. Current tools and algorithms follow software patterns optimized for single-GPU engineering workstations, leaving huge opportunities to scale efficiency and capacity. Figure 5(a) shows how in OpenROAD, distributed incremental detailed routing is sped up by $\sim 100\times$ in the cloud, using 20 16-core workers. Overheads of serialization and task distribution limit the achievable cloud-based speedup for this code, as shown in the figure. Figure 5(b) shows how pin access analysis, which is performed at the start of detailed routing, is sped up by $30\times$ using cloud instances as opposed to a single machine. In general, having more machines available to AI orchestration will enable richer patterns of exploration and exploitation. Potentially, some optimization threads can explore the basic flow even as others explore floorplans, cell/corner options, and other high-impact decision points.

Sampling for Stability. Outcomes of physical design are chaotic, especially when tools and flows are driven to "try harder" (e.g., to achieve higher die utilization in costly advanced nodes). The chaotic behavior is due to the complex interactions of many internal metaheuristics [7, 28]. Figure 6(a) shows noise in the output

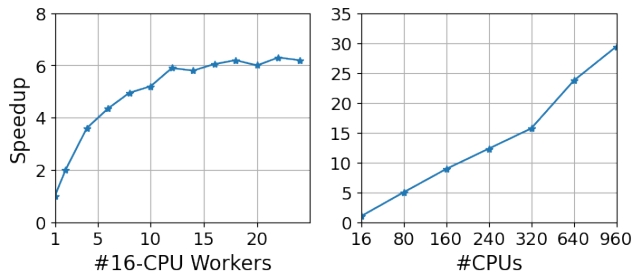

Figure 5: (a) Speedup of incremental detailed routing in OpenROAD by 100× with 20 16-core workers. (b) Pin access analysis can be completed 30× faster using cloud instances as opposed to a single server [38].

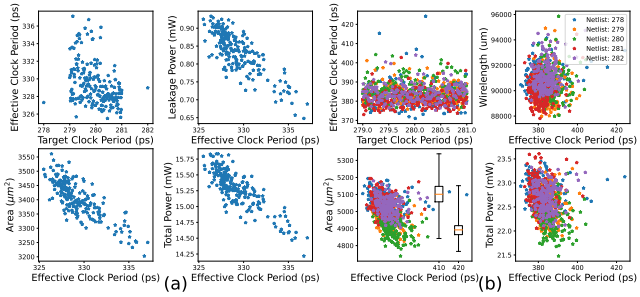

Figure 6: (a) Variation of metrics across 201 output netlists from logic synthesis, with target clock periods evenly spaced in a 2ps interval. (b) Variation of metrics across 1005 final P&R results. Testcase: AES, GF12LP.

netlist metrics of 201 runs of a commercial logic synthesis tool, for the AES design in a GLOBALFOUNDRIES 12nm (triple-VT) enablement. Figure 6(b) shows results from 201 P&R runs for each of five input netlists (i.e., a total of 1005 P&R runs).[3] Two box and whisker diagrams show area distributions for two additional sets of 1005 P&R runs; these sets respectively comprise 201 P&R runs starting from each of the five lowest-area (respectively, highest-area) post-synthesis netlists. This confirms large tool noise as well as relatively poor correlation between the synthesis and P&R flow steps. To mitigate this and improve stability of PD outcomes, cloud deployment of optimizers seems inevitable.

Multiple Views in Unit Time: Tomography. In a number of domains, terms such as "multi-spectral imaging" or "sensor fusion" refer to the use of multiple views of an object to enhance identification or analysis. An analogous term, *tomography*, involves creating multiple images to provide detailed insights into the human body or various solid objects. At UCSD, the idea of tomography in place-and-route has led to techniques that generate *many* views of a given design within the same walltime that is needed for *one* view – along with use of these many views to achieve better optimization or ML modeling.

[3]For (a), the target clock period (TCP) constraint in *logic synthesis* is 280ps; it is stepped by 0.01ps for 100 steps (positive and negative directions), yielding 201 distinct synthesis runs that vary only in the SDC constraint, all within a 2ps interval ([279ps,281ps]). For (b), five input netlists are synthesized with SDC clock periods of {278, 279, 280, 281, 282}ps. For each netlist, 201 P&R runs are made with TCP stepped by 0.01ps in the interval [279ps,281ps], yielding 1005 total P&R results.

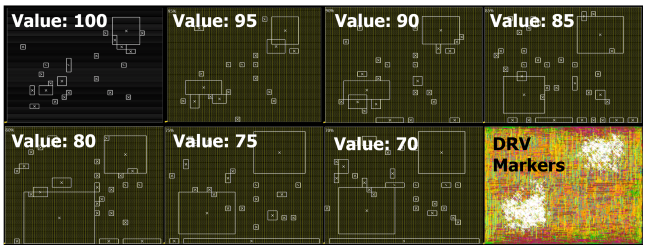

Figure 7: Congestion report for routing blockages with varying partial densities and post-route DRC markers.

A specific example of tomography in P&R uses the Cadence Innovus early global router (eGR) to generate congestion reports under various routing blockage conditions. eGR is much faster than traditional detailed routing (e.g., runtime of < 1 second, versus 1.5 hours), which enables generation of hundreds of congestion reports for different routing blockages near-instantaneously when run in parallel. Figure 7 shows seven congestion reports generated using eGR, each with a different partial density value for the routing blockage covering the design.[4] The congestion report has closest alignment to actual DRC markers when the partial density value is 75. But, using *all* of the images in this placement tomography improves ML model accuracy in congestion or DRC hotspot prediction.

3 ELEMENTS OF A NEXT WAVE

This section gives several elements of a "next wave" for AI/ML in IC physical design. These include the onrush of generative AI, continuing to seek "magic" conditioning of problem instances at flow interstices, infrastructure for ML, and overdue culture changes.[5]

3.1 Generative AI.

Across the technology world, generative AI has "taken all of the oxygen out of the room", and EDA and IC design are no exception. Generative AI does not afford solvers or optimizers in the usual sense: failure rates can be high, so answers require checking and correction. Nevertheless, GenAI has found use cases where guesses can be discarded (if wrong) or else improved. It is very likely generative AI methods – spanning from language to graphs – will soon be pervasive in PD. Applications include interactive tool help, design goal-specific tool and flow recipes, script and HDL code debugging (e.g., [53]) and copiloting, virtual teaching assistants, and more. ([17] provides a more far-ranging perspective than is possible here.)

In the near term, EDA and IC design companies will leverage LLMs along three main vectors. (i) LLM-based design assistants

[4]The semantics in the P&R tool: A partial density of 70 indicates that only 70% of the total routing resource is available during global routing.

[5]A few important elements are not elaborated here. (i) Partitioning engines have rapidly advanced in the past two years, bringing generalized eigenvalue formulations and embeddings, cut-overlay clustering, incorporation of multi-dimensional design and analysis data, and other innovations [4] [3]. Layering of ML on top of these new engines, starting with autotuning, will likely provide even better and more stable methods. (ii) Clustering and sparsification reduce problem size and "prevent mistakes", potentially leading to simultaneous speedup and solution quality improvement [13]. ML may lead to more useful clustering objectives and methods in contexts such as floorplanning, placement and clock tree synthesis.

will provide a chat interface to answer various types of designer queries about the tool, design, or technology. (ii) Prompt-based tool flows will provide live feedback between the tool and the user using prompts, leveraging a combination of LLMs and deep learning models (predictors of downstream flow outcomes, recommenders for flow recipes, etc.). (iii) Graph generative models will be the basis for fast, incremental physical synthesis, place-and-route and optimization that works on the cones of logic affected by incremental changes to RTL, netlist, constraints or floorplan contexts. The second and third vectors, if successful, can potentially revolutionize designer productivity – but perhaps not end quality of outcomes – in physical design.

Broadly speaking, generative methods (and, AI in general) are likely to see adoption in contexts where humans are not in competition with the AI, and are happy to be relieved of mundane, tedious tasks. What humans dislike creating (e.g., documentation, verifications, and tests) will provide the initial applications for generative methods. ChipNeMo [46] applies domain adaptation methods to optimize LLMs for three such tasks – chatbot assistant, tool script generation, and bug summarization.

Whether GenAI can overcome well-recognized blockers remains to be seen. Data for model training must rely on a mix of real and synthetic data; it is unclear whether synthetic data that safeguards IP rights can be produced using real samples as seeds, and then released into the wild. Generalization will depend on data quality and whether general methods – as opposed to models specialized to a few canonical types of IC designs and design styles – are needed. Cost will presumably be attacked using pre-trained models plus fine-tuning, as well as model compression and sparsification. Debugging and diagnosing the inevitable incorrect predictions and hallucinations will require task-specific checkers and debuggers. Other application domains will also develop methods to detect and prevent hallucinations.

Notwithstanding the above, other factors may lead to less rosy futures for generative AI in PD and chip implementation. Three examples: (i) insufficient training data for the ML models that underlie EDA flow control; (ii) data poisoning and improper use of copyrighted materials that increase cost and risk of GenAI models; and (iii) LLM successes seen only in applications for which large amounts of public data are available for training (e.g., small PyTorch scripts, Copilot for Verilog coding).

3.2 ML at Interstices.

[25] and contemporaneous works set out a number of R&D targets for ML and physical design. In the arena of predictive modeling of tools and designs, these were often referred to as "magic" – netlists, floorplans, corners and constraints, etc. For example, it was observed that a single netlist is handed off from logic synthesis to place-and-route, and that a challenge for AI/ML was to generate "magic" recipes to produce a best-possible netlist at this handoff point.

The left side of Figure 8 lists a number of coevolutions and co-optimizations that correspond to interstices between flow steps. There is opportunity for "magic" at these interstices because the tools and corresponding flow steps are typically developed and executed "at arm's length".

Co-optimizations
- Netlist – Backend
- Hierarchy – Floorplan
- Floorplan – SP&R
- Synthesis – P&R
- Place – Route
- GRoute – DRoute

"Magic"
- Netlist
- Netlist Partitioning
- Block shaping *+ boundaries*
- Placement screens
- Route screens
- Route guides
- Corners *+ endpoint SDCs*
- Constraints
- Tool/engine recipes
- ...

Figure 8: Coevolution of optimizers and opportunities for "magic" at interstices.

Recent years have made it clear that "prediction is difficult": the field has not achieved usable predictors of subflow outcomes, such as post-route design rule violations from a global placement, given the high cost of prediction errors. Work at UCSD instead seeks methods to *condition* the problem instance that is passed forward at a given interstice between flow steps. As noted in the discussion of placement-aware hierarchical floorplanning, a placement problem instance can be conditioned by adjusting target density and cell padding, and/or by adding placement and routing blockages. Relatively simple adjustments of target density at the start of placement can yield substantial improvements of post-route-opt wirelength, total power, WNS and TNS metrics. Or, for a fixed placement, proper conditioning of the routing problem with routing blockages (per-region and per-layer; recall Figure 7) can dramatically reduce post-route DRCs. We refer to such routing and placement blockages as "magic screens". An important goal is to develop ML-based generation of magic screens that will reduce design iterations while improving PPA.

3.3 Infrastructure for ML.

OpenROAD and CircuitOps. While ML – including various forms of generative AI – has been a rapidly growing field of EDA research, this area still lacks dedicated research infrastructure: (i) Python APIs in EDA tools for faster data generation, (ii) a standard format for data representation and data sharing, and (iii) a Python interface with EDA tools to enable a feedback path from ML algorithms back into the EDA platform. As shown in Figure 9 the OpenROAD project [2] and NVIDIA's CircuitOps [36, 61] have recently made progress together in addressing these challenges.[6] This has enabled a new, open-source "playground" for ML-EDA researchers. Specifically, (i) Python APIs in OpenROAD wrap the underlying C++ APIs of its EDA engines, which enables faster data generation compared to using commercial tools' TCL interfaces. (ii) NVIDIA's CircuitOps leverages OpenROAD to model chip data as labeled property graphs, and uses pandas data frames to store graph properties as features. (iii) CircuitOps further leverages recently-added Python APIs in OpenROAD as a feedback loop to interact with the ML algorithms. The recent ASP-DAC 2024 tutorial [59] demonstrates a reinforcement learning-based gate sizer which uses this feedback loop in OpenROAD to perform gate sizing.[7]

[6] Figure provided by Professor Vidya Chhabria, with portions from [6, 36].
[7] Several closed efforts also seek to enable standard APIs and data/ML platforms for EDA. Sources span consortia (Si2 SPEED API), EDA companies (Cadence JedAI and Synopsys DesignDash), and academia (Drexel University EDA-Schema).

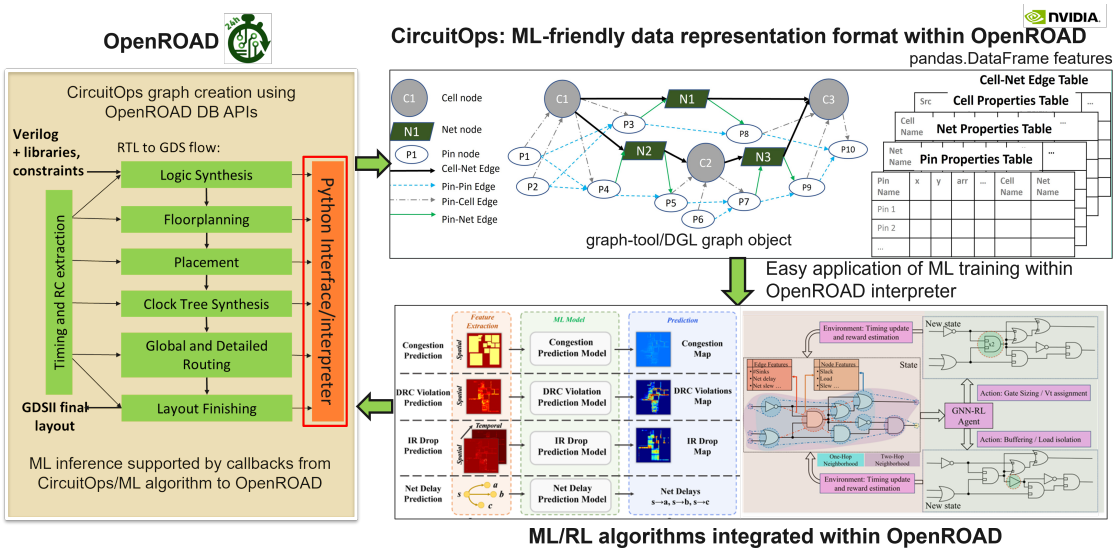

Figure 9: OpenROAD as a new EDA playground for ML researchers.

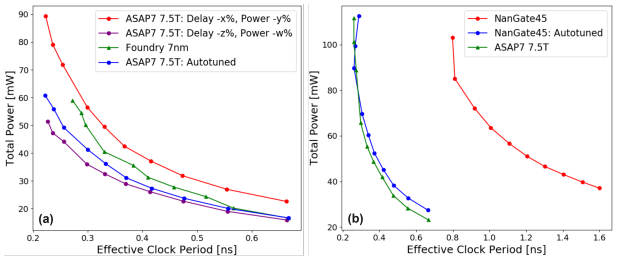

Figure 10: (a) Autotuning of ASAP7 enablement to match the PPA hockey stick of foundry 7nm enablement. (b) Autotuning of NanGate45 enablement to match ASAP7 enablement.

Proxies for Open Data. Progress of ML for EDA continues to be hindered by the lack of open data for research and education. Today, the data produced by commercial tools in academic contexts is blocked from dissemination, sharing, or use in ML model (e.g., LLM) training. This is a consequence of tool documentation, Tcl command names and reports, and tool outputs all being proprietary. Efforts to develop open *proxies* (PDKs, design enablements, tools and designs) seek to remove this obstacle [22].

For example, when creating a proxy design enablement the design- and research-relevant characteristics of a *source* technology should match the corresponding characteristics of a *target* PDK. In Figure 10(a), *source* is the ASAP7 academic research PDK, while *target* is a commercial 7nm foundry technology. The plot shows the Power versus Effective Clock Period "hockey stick" for ten target clock periods. Simple scaling of ASAP7 delay and power can bound (red, purple) outcomes from the foundry 7nm enablement (green). Autotuning with Ray/Tune [37, 65], using simple tuning parameters (delay, pin capacitance, internal/switching power, setup/hold times) and a loss function of Mean Absolute Percentage Error across the 10 target clock periods, yields the blue hockey stick with loss function value of 11%. Figure 10(b) demonstrates how autotuning can bridge technologies (from NanGate45 to ASAP7).

Open design testcases, along with infrastructure for proxy cell library creation and design-technology co-optimization [11, 64] and artificial-but-realistic netlist generation [33, 58], are rapidly coming online. There is also opportunity for ML to take us deeper in the context of proxies and matching. For example, sufficient analysis data might enable deeper ML-based approximation of the design enablement – down to BSIM models, RC extraction techfiles, and even etch and CMP models. Indeed, this may be inevitable, as there is an "analog hole" for design analysis information, and because "it's just physics" after all.

3.4 Overdue Culture Changes.

Among the hallmarks of a mature technical field are that (i) reported results are reproducible, and (ii) progress can be clearly measured – because we "Measure, to Improve". To enable reproducibility of their reported results, PD researchers can now post Tcl scripts for research purposes, with proper acknowledgment of the interests of the tool suppliers. This has been an amazing breakthrough, and kudos are due to the EDA vendors for making this policy change [23]. However, as a *community*, we do not yet require papers with code or award badges for excellent open-source collaterals. And, benchmarking of commercial EDA remains forbidden.

Can we accelerate culture changes that move the research leading edge toward benchmarking and open source? There is no question that benchmarking and benchmarks will accelerate the (transparent, reproducible) improvement of optimizers, baselines, and in-context assessments. Similarly, there is no question that open infrastructure, open proxies, and open-source EDA will accelerate data generation that feeds the learning of objectives and the discovery of more impactful AI/ML methods (cf. the "AI flywheel"). This would be true for AI/ML both "inside" EDA and "around" EDA, with the

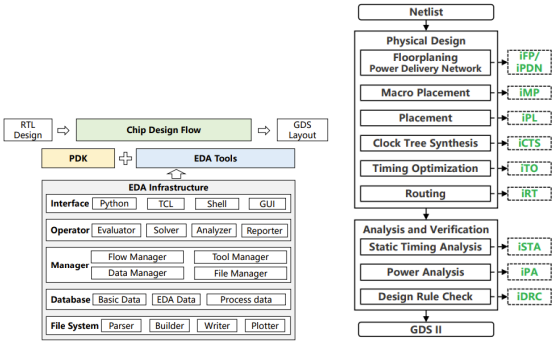

Figure 11: (a) The iEDA infrastructure [34]; (b) physical design steps and coverage by iPD tools [35].

former potentially being infeasible for researchers *unless* they work with open-source codes.[8]

Toward these ends, added impetus is likely to come from the recent *iEDA* [34] and *iPD* [35] efforts. iEDA ("intelligent EDA", Figure 11(a)) is rapidly growing open-source EDA infrastructure that seeks to facilitate development of more valuable EDA tools and algorithms, while attracting R&D contributors from diverse academic disciplines and closing gaps between industry and academia. iPD ("intelligent Physical Design", Figure 11(b)) is an open-source physical design toolchain built on iEDA infrastructure; it has already supported SoC tapeouts in commercial nodes down to 28nm. Notably, in contrast to OpenROAD, the iPD toolchain has adopted a more decoupled software architecture, aiming for faster development of more extensive and varied EDA research tools and algorithms.

A final, cautionary note: Transparency and reproducibility are not a substitute for research integrity, and can still be swamped by marketing hype and other "noise". Rigorous empirical evaluation – using public data – of claimed improvements to solvers, engines, tools and flows is increasingly important to preserve a healthy research ecosystem in PD and EDA [10]. Fortunately, such empirical evaluation is embarrassingly parallel and amenable to automation.

4 CONCLUSIONS

Over the six years since ISPD-2018 and [25], researchers have incorporated AI/ML methods into nearly every core area of physical design. EDA vendors have achieved commercial successes, notably with hyperparameter optimization (autotuning) around SP&R, but also in verification, PCB and advanced packaging layout, and other areas as well. Interestingly, the "half a node's progress" [60] from wrapping an autotuner around digital SP&R lends support to long-standing claims that commercial EDA had left over a node of optimization QoR scaling on the table. AI/ML will play an important part in clawing back semiconductor value via "the last scaling levers" of quality, cost and schedule.

[8]One might ask: Will ISPD-2030 have a papers-with-code requirement, and award a prize for best-curated open-source repository? By 2030, will there be a journal of open-source design and design automation – or, of confirmation studies? And, will there be a neutral organization that measures and reports on the progress of EDA technology?

The adoption of AI/ML in PD has generally followed the taxonomy and sequencing proposed in [24–26] (Figures 1, 2). However, blockers have emerged from the closed nature of commercial tool silos, and from the lack of open infrastructure, data and sharing mechanisms that would enable an "AI flywheel" for the PD research community [62]. For target list items proposed in 2018, comparatively more progress has been made on estimations of physics analyses (delay, crosstalk, power, IR, thermal), while less progress has been made on predictors of syntheses and optimizations (particularly, extremal criteria such as worst path timing or overcongestion in routing hotspots). Many of the target list items remain open today.

The context for research going forward includes basic realizations: (i) optimization baselines are hard to beat (and accuracy bars are high), so ML will take aim at new, less-formal and/or mechanical challenges; (ii) ML has not scaled well to large instances; and (iii) best-performing metaheuristics are chaotic at their QoR limits. Some challenges, such as predicting outcomes of long multi-stage subflows, or reusing training data and models across designs, tools and foundry nodes, will likely prove quite stubborn. Following are just a few out of many threads to watch for in the coming years.

- ML-boosted automations in production for lower-hanging fruits: floorplan squeezing, datapath-aware floorplanning, and small-scale end-case optimizations (e.g., detailed placement for routability, post-route-opt DRC fix, gate sizing).
- Production-worthy ML methods to *condition* problem instances with "magic" screens, corners, and/or constraints at one or more interstices of the flow (Figure 8).
- Critical mass, usability *and* sustaining mechanism for an *open* ML EDA infrastructure that provides researchers with a spinning AI flywheel (i.e., running "on the exhaust of data") and reference ops pipelines.
- Advancement of open-source EDA and proxy research enablements through efforts such as OpenROAD [2], iEDA [34], CircuitOps [61] and the IEEE CEDA DATC [65].
- Commercial success – based on dominating QoR, cost, and runtime – of generative AI methods for prompt-based flow control and for *incremental* PD optimizations.
- Industry acknowledgment of the value of sampling and cloud scaling – and of the much greater resources that are needed to unlock AI/ML benefits with these levers.
- Widespread adoption of generative AI-infused training modules across levels (high school through re-skilling and up-skilling) for IC physical design and design automation.
- Advancement of culture in a healthy PD research community, to seek and reward transparency, reproducibility, and benchmarking to measure progress.

5 ACKNOWLEDGMENTS

Many thanks to Sayak Kundu, Bodhisatta Pramanik, Zhiang Wang and Dooseok Yoon for their help with the figures and text in this paper. Discussions with Siddhartha Nath, Igor Markov, Chuck Alpert and Ilgweon Kang are also gratefully acknowledged. Research at UCSD is partially supported by DARPA, Samsung, the C-DEN center, and gifts from Google, Intel and others.

REFERENCES

[1] A. Agnesina, P. Rajvanshi, T. Yang, G. Pradipta, A. Jiao et al., "AutoDMP: Automated DREAMPlace-based Macro Placement", *Proc. ISPD*, 2023, pp. 149-157.

[2] T. Ajayi, V. A. Chhabria, M. Fogaça et al., "Toward an Open-Source Digital Flow: First Learnings from the OpenROAD Project", *Proc. ACM/IEEE DAC*, 2019, pp. 76:1-76:4.

[3] I. Bustany, G. Gasparyan, A. B. Kahng, Y. Koutis, B. Pramanik and Z. Wang, "An Open-Source Constraints-Driven General Partitioning Multi-Tool for VLSI Physical Design", *Proc. ICCAD*, 2023, pp. 1-9.

[4] I. Bustany, A. B. Kahng, Y. Koutis, B. Pramanik and Z. Wang, "K-SpecPart: Supervised Embedding Algorithms and Cut Overlay for Improved Hypergraph Partitioning", *arXiv.2305.06167*, 2023.

[5] M. Cavalcante, S. Riedel, A. Pullini and L. Benini, "MemPool: A Shared-L1 Memory Many-Core Cluster with a Low-Latency Interconnect", https://arxiv.org/pdf/2012.02973.pdf

[6] Z. Chai, Y. Zhao, W. Liu, Y. Lin, R. Wang and R. Huang, "CircuitNet: An Open-Source Dataset for Machine Learning in VLSI CAD Applications With Improved Domain-Specific Evaluation Metric and Learning Strategies", *IEEE Trans. on CAD* 42(12) (2023), pp. 5034-5047.

[7] T.-B. Chan, A. B. Kahng and M. Woo, "Revisiting Inherent Noise Floors for Interconnect Prediction", *Proc. SLIPP*, 2020, pp. 1-7.

[8] C.-K. Cheng, A. B. Kahng, I. Kang, M. Kim, D. Lee, B. Lin, D. Park and M. Woo, "CoRe-ECO: Concurrent Refinement of Detailed Place-and-Route for an Efficient ECO Automation", *Proc. ICCD*, 2021, pp. 366-373.

[9] C.-K. Cheng, A. B. Kahng, I. Kang and L. Wang, "RePlAce: Advancing Solution Quality and Routability Validation in Global Placement", *IEEE Trans. CAD* 38(9) (2019), pp. 1717-1730.

[10] C.-K. Cheng, A. B. Kahng, S. Kundu, Y. Wang and Z. Wang, "Assessment of Reinforcement Learning for Macro Placement", *Proc. ISPD*, 2023, pp. 158-166.

[11] S. Choi, J. Jung, A. B. Kahng, M. Kim, C.-H. Park, B. Pramanik and D. Yoon, "PROBE3.0: A Systematic Framework for Design-Technology Pathfinding with Improved Design Enablement", *IEEE Trans. on CAD*, to appear (2023).

[12] 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.

[13] M. Fogaça, A. B. Kahng, E. Monteiro, R. Reis, L. Wang and M. Woo, "On the Superiority of Modularity-Based Clustering for Determining Placement-Relevant Clusters", *Integration: The VLSI Journal* 74 (2020), pp. 32-44.

[14] Z. Guo, T. -W. Huang and Y. Lin, "Accelerating Static Timing Analysis Using CPU–GPU Heterogeneous Parallelism", *IEEE Trans. on CAD* 42(12) (2023), pp. 4973-4984.

[15] G. Guo, T. -W. Huang, Y. Lin, Z. Guo, S. Yellapragada et al., "A GPU-Accelerated Framework for Path-Based Timing Analysis", *IEEE Trans. on CAD* 42(11) (2023), pp. 4219-4232.

[16] X. He, Z. Fu, Y. Wang, C. Liu and Y. Guo, "Accurate Timing Prediction at Placement Stage with Look-Ahead RC Network", *Proc. DAC*, 2022, pp. 1213–1218.

[17] Z. He and B. Yu, "Large Language Models for EDA: Future or Mirage?", to appear in *Proc. ISPD*, 2024.

[18] Z. He, Y. Zuo, J. Jiang, H. Zheng, Y. Ma et al., "OpenDRC: An Efficient Open-Source Design Rule Checking Engine with Hierarchical GPU Acceleration", *Proc. DAC*, 2023, pp. 1-6.

[19] T. -W. Huang, "Machine Learning System-Enabled GPU Acceleration for EDA", *Proc. VLSI-DAT*, 2021, pp. 1-1.

[20] J.-W. Jeon, A. B. Kahng, J.-H. Kang, J. Kim and M. Woo, "SLO-ECO: Single-Line-Open Aware ECO Detailed Placement and Detailed Routing Co-Optimization", to appear in *Proc. ISQED*, 2024.

[21] J. Jung, A. B. Kahng, S. Kim and R. Varadarajan, "METRICS2.1 and Flow Tuning in the IEEE CEDA Robust Design Flow and OpenROAD", *Proc. ICCAD*, 2021, pp. 1-9.

[22] J. Jung, A. B. Kahng, S. Kundu, Z. Wang and D. Yoon, "IEEE CEDA DATC Emerging Foundations in IC Physical Design and MLCAD Research", *Proc. ICCAD*, 2023.

[23] D. Junkin, "Supporting the Scientific Method for the Next Generation of Innovators", *presentation at DAC Open-Source EDA Birds-of-a-Feather Meeting*, 2022. https://open-source-eda-birds-of-a-feather.github.io/doc/slides/BOAF-Junkin-DAC-Presentation.pdf

[24] A. B. Kahng, "New Directions for Learning-Based IC Design Tools and Methodologies", *Proc. ASP-DAC*, 2018, pp. 405-410.

[25] A. B. Kahng, "Machine Learning Applications in Physical Design: Recent Results and Directions", *Proc. ISPD*, 2018, pp. 68-73.

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

[27] A. B. Kahng, "Machine Learning in EDA", *IBM eDAW keynote talk*, April 2020. https://vlsicad.ucsd.edu/NEWS20/IBM-ABK-200430-v4a-ACTUAL.pptx

[28] A. B. Kahng, "Advancing Placement", *Proc. ISPD*, 2021, pp. 15-22.

[29] A. B. Kahng, "Leveling Up: A Trajectory of OpenROAD, TILOS and Beyond", *Proc. ISPD*, 2022, 73-79.

[30] A. B. Kahng, "AI/ML and EDA: Current Status and Perspectives on the Future", *ASP-DAC 2024 keynote talk*. https://vlsicad.ucsd.edu/NEWS24/ASP-DAC-2024-Keynote-Kahng-v1f.pptx

[31] A. B. Kahng and S. Mantik, "A System for Automatic Recording and Prediction of Design Quality Metrics", *Proc. ISQED*, 2001, pp. 81-86.

[32] A. B. Kahng, and Z. Wang, "ML for Design QoR Prediction", *Machine Learning Applications in Electronic Design Automation*, Cham, Springer, 2022.

[33] D. Kim, S. Y. Lee, K. Min and S. Kang, "Construction of Realistic Place-and-Route Benchmarks for Machine Learning Applications", *IEEE Transaction on Computer-Aided Design of Integrated Circuits and Systems* 42(6) (2022), pp. 2030–2042.

[34] X. Li, Z. Huang, S. Tao, Z. Huang, C. Zhuang et al., "iEDA: An Open-source infrastructure of EDA", *Proc. ASP-DAC*, 2024. https://www.cse.cuhk.edu.hk/~byu/papers/C196-ASPDAC2024-iEDA-slides.pdf

[35] X. Li, S. Tao, S. Chen, Z. Zeng, Z. Huang et al., "iPD: An Open-source intelligent Physical Design Tool Chain", *Proc. ASP-DAC*, 2024. https://www.cse.cuhk.edu.hk/~byu/papers/C195-ASPDAC2024-iPD-slides.pdf

[36] R. Liang, A. Agnesina, G. Pradipta, V. A. Chhabria and H. Ren, "CircuitOps: An ML Infrastructure Enabling Generative AI for VLSI Circuit Optimization", *Proc. ICCAD*, 2023.

[37] R. Liaw, E. Liang, R. Nishihara, P. Moritz, J. E. Gonzalez and I. Stoica, "Tune: A Research Platform for Distributed Model Selection and Training", *arXiv preprint 1807.05118*, 2018.

[38] M. Liberty, *personal communication*, October 2023.

[39] D.-L. Lin, H. Ren, Y. Zhang, B. Khailany and T.-W. Huang. "From RTL to CUDA: A GPU Acceleration Flow for RTL Simulation with Batch Stimulus", *Proc. ICPP*, 2022, pp. 88:1-88:12.

[40] S. Lin, J. Liu and M. D. F. Wong, "GAMER: GPU Accelerated Maze Routing", *Proc. ICCAD*, 2021, pp. 1-8.

[41] S. Lin and M. D. F. Wong, "Superfast Full-Scale GPU-Accelerated Global Routing", *Proc. ICCAD*, 2022, pp. 1-8.

[42] Y. Lin, "GPU Acceleration in VLSI Back-end Design: Overview and Case Studies", *Proc. ICCAD*, 2020, pp. 1-4.

[43] Y. Lin, Z. Jiang, J. Gu, W. Li, S. Dhar et al., "DREAMPlace: Deep Learning Toolkit-Enabled GPU Acceleration for Modern VLSI Placement", *IEEE Trans. on CAD* 40(4) (2021), pp. 748-761.

[44] Y. Lin, W. Li, J. Gu, H. Ren, B. Khailany et al., "ABCDPlace: Accelerated Batch-Based Concurrent Detailed Placement on Multithreaded CPUs and GPUs", *IEEE Trans. on CAD* 39(12) (2020), pp. 5083-5096.

[45] L. Liu, B. Fu, M. D. F. Wong and E. F. Y. Young, "Xplace: an Extremely Fast and Extensible Global Placement Framework", *Proc. DAC*, 2022, pp. 1309-1314.

[46] M. Liu, T.-D. Ene, R. Kirby et al., "ChipNeMo: Domain-Adapted LLMs for Chip Design", *arXiv preprint 2311.00176*, 2023. https://arxiv.org/pdf/2311.00176.pdf

[47] S. Liu, P. Liao, R. Zhang, Z. Chen, W. Lv et al., "FastGR: Global Routing on CPU-GPU with Heterogeneous Task Graph Scheduler", *Proc. DATE*, 2022, pp. 760-765.

[48] T. Liu and E. F. Y. Young, "Rethinking AIG Resynthesis in Parallel", *Proc. DAC*, 2023, pp. 1-6.

[49] Y.-C. Lu, S. Nath, S. Pentapati and S. K. Lim, "ECO-GNN: Signoff Power Prediction Using Graph Neural Networks with Subgraph Approximation", *ACM Trans. Des. Autom. Electron. Syst.* 28(4) (2023), pp. 1-12.

[50] A. Mirhoseini, A. Goldie, M. Yazgan, J. Jiang, E. Songhori et al., "A Graph Placement Methodology for Fast Chip Design", *Nature* 594(7862) (2021), pp. 207-212.

[51] S. Nath, G. Pradipta, C. Hu, T. Yang, B. Khailany et al., "TransSizer: A Novel Transformer-Based Fast Gate Sizer", *Proc. ICCAD*, 2022, pp. 1-9.

[52] H. Ren, "Embracing Machine Learning in EDA", *Proc. ISPD*, 2022, pp. 55-56.

[53] Y. Tsai, M. Liu and H. Ren, "RTLFixer: Automatically Fixing RTL Syntax Errors with Large Language Models", *arXiv preprint 2311.16543*, 2023.

[54] B. Wang, G. Shen, D. Li, J. Hao, W. Liu et al., "LHNN: Lattice Hypergraph Neural Network for VLSI Congestion Prediction", *Proc. DAC*, 2022, pp. 1297–1302.

[55] Z. Xie, Y.-H. Huang, G.-Q. Fang, H. Ren et al., "RouteNet: Routability Prediction for Mixed-Size Designs Using Convolutional Neural Network", *Proc. ICCAD*, 2018.

[56] E. F.Y. Young, "GPU Acceleration in Physical Synthesis", *Proc. ISPD*, 2023, pp. 167-167.

[57] X. Zhang, J. Shiely and E. F.Y. Young, "Layout Pattern Generation and Legalization with Generative Learning Models", *Proc. ICCAD*, 2020, pp. 1-9.

[58] Artificial Netlist Generator. https://github.com/daeyeon22/artificial_netlist_generator

[59] ASP-DAC 2024 Tutorial. https://github.com/ASU-VDA-Lab/ASP-DAC24-Tutorial

[60] "The Power of Computational Software – From Revolutionizing Chips to Cancer Research", *Cadence Blog*, August 23, 2023. https://community.cadence.com/cadence_blogs_8/b/corporate/posts/the-power-of-computational-software-from-revolutionizing-chips-to-cancer-research

[61] CircuitOps Repository. https://github.com/NVlabs/CircuitOps

[62] NSF Workshop on Shared Infrastructure for Machine Learning EDA, March 2023. https://sites.google.com/view/ml4eda/home

[63] OpenROAD RePlAce. https://github.com/The-OpenROAD-Project/OpenROAD/tree/master/src/gpl

[64] The PROBE3.0 Framework. https://github.com/ABKGroup/PROBE3.0

[65] IEEE CEDA DATC RDF 2023. https://github.com/ieee-ceda-datc/RDF-2023

[66] SkyWater Open Source PDK. https://github.com/google/skywater-pdk

[67] TSMC FinFlex. https://www.tsmc.com/english/news-events/blog-article-20220616