Use Cases and Deployment of ML in IC Physical Design

Amur Ghose*, Andrew B. Kahng*†, Sayak Kundu†, Yiting Liu*, Bodhisatta Pramanik†,
Zhiang Wang† and Dooseok Yoon†
UC San Diego Departments of CSE* and ECE†
{aghose,abk,sakundu,yil375,bopramanik,zhw033,d3yoon}@ucsd.edu

ABSTRACT

ML for IC physical design must be deployed in order to have business impacts. However, deployment in production must navigate many practical considerations, including choice of targets, skillsets and infrastructure, expectations and resources, data, and "MLOps". Furthermore, usage of ML is not the same as IC design practice and capability. In this invited paper, we give perspectives on basic strategies for selecting applications and pursuing deployment for ML in IC physical design. Example aspects include checklists for data and ML models, evaluation of model performance and progress on the path to deployment, the shifting landscape of MLOps, and challenges of "LLM-ability".

CCS CONCEPTS

• **Hardware → Physical design (EDA)**; • **Computing methodologies → Machine learning**.

KEYWORDS

Integrated-circuit Physical Design, Machine Learning, MLOps

ACM Reference Format:
Amur Ghose, Andrew B. Kahng, Sayak Kundu, Yiting Liu, Bodhisatta Pramanik, Zhiang Wang and Dooseok Yoon. 2025. Use Cases and Deployment of ML in IC Physical Design. In *30th Asia and South Pacific Design Automation Conference (ASPDAC '25), January 20–23, 2025, Tokyo,Japan.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3658617.3703143

1 INTRODUCTION

In recent years, many IC design organizations have sought to address the complexity, turnaround time and solution quality challenges of IC physical design (PD) through application of AI and machine learning (ML) techniques [17] [23]. This has been spurred by commercial EDA successes such as hyperparameter autotuning [71] [79] [30], as well as ML EDA research that reports promising results on numerous fronts [31] [43]. Prediction tasks in the literature (see [25]) span routing hotspot prediction [49], doomed run prediction [32], and PPA prediction [12]. ML that produces helpful design suggestions includes prediction of tool settings to improve PPA [16], and creation of routing blockages to enhance routability and PPA [24]. Progress on classical PD optimizations has been reported for partitioning [6], clustering [33], macro placement [35], placement [27], timing optimization [36] and more; see [18] and [19] for recent reviews.

Large language models (LLMs) and other generative AI (GenAI) methods further expand the scope of possibilities [50]. Nascent applications of GenAI models include generating testbenches [41], summarizing and diagnosing large collections of place-and-route runs, and assisting engineers by recalling scripts and commands [45]. These hold promise to transform workflows, automate tedious processes, and otherwise improve efficiency in IC PD.

Despite great interest in "AI-powered design flows", challenges to production deployment and business impact are seen in practice. Cross-functional teams and enterprise-wide initiatives often fall short of desired outcomes, which begs the question: *Why have so many efforts fallen short?* Relevant factors span data, skillsets, targets, resources and more. Resourcing of data and "*MLOps*" is a first-class concern, but is often an afterthought. Strategies such as targeting incremental improvement of existing design processes, and early wins to build organizational confidence, are also essential.

In this paper, we give some perspectives on applications and deployment for ML in IC physical design. Our discussion spans (i) issues surrounding data for ML, (ii) high-level principles for deployment, (iii) basic "checklists" for data, models and use cases, and (iv) the rapidly shifting context for MLOps and LLM-based application development.[1] In the following, Section 2 discusses issues surrounding data in IC design. Section 3 addresses deployment of ML in PD, Section 4 discusses implications of LLM-based approaches and "LLM-ability", and the paper concludes in Section 5.

2 DATA

Data is a first-class, up-front concern for ML success: quality, quantity and relevance of data directly influence the reliability and performance of ML models. We now discuss the data context for ML applications and deployment in IC PD: (i) ML data outside vs. inside IC design; (ii) challenges such as silos and multimodality; and (iii) the spectrum of ongoing efforts to address data challenges.

2.1 Data Outside vs. Inside IC Design

Outside of IC design, Generative AI (GenAI) [14] [15] [48] has attracted enormous attention and investment. GenAI models rely heavily on extensive datasets that support unsupervised learning, with five principal types of data: natural language, code, images, video and audio. Leading models can generate high-quality artifacts

[1]For the most part, we focus on issues and challenges that apply to the ML-boosted use of EDA tools in IC design organizations, not the application of ML methods by EDA researchers and tool vendors. Commercial EDA companies have achieved notable product successes based on AI/ML integration, for tool and flow autotuning (e.g., Synopsys DSO.ai [79] and Cadence Cerebrus [71]), acceleration of physics simulations (e.g., Ansys SimAI [62] and Siemens Solido [77]), layout design optimization (e.g., Cadence Allegro X AI [70]), and other domains. However, a review of such commercial EDA tooling is beyond our present scope.

ASPDAC '25, January 20–23, 2025, Tokyo, Japan
© 2025 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0635-6/25/01
https://doi.org/10.1145/3658617.3703143

across a variety of natural language [10] [48] and image processing [5] applications.

As noted above, IC design and EDA have also enthusiastically embraced GenAI across a huge range of potential contexts, e.g., coding/scripting assistance; knowledge retrieval; verification, testing and documentation tasks; and improvement of tool runscripts [27] [29] [56]. However, data within the IC design universe is "different": it includes formal specifications, hardware description language (HDL) code, graphical representations, hierarchical data structures, tabular data and images. Data in IC design is typically also scarce and proprietary. Unlike more common datasets that have massive redundancy, IC design data typically presents unique instances with minimal repetition.

2.2 Challenges

Varied forms and scales in IC design data present challenges to traditional data handling and analysis methods.

Generalization across modalities. Leading GenAI models [2] [72] have strong ability to understand multiple data modes such as natural language, code and image. However, these models have shown limited transferability when applied to IC data. A primary method for handling structured data is transformation into a common form of structured text, such as code and its derivatives. Thus, constrained output specifications are expressed using formats such as JSON or XML, rather than by enabling specialized formal rules or design representations. As a result, IC-related data – such as specification sheets, technical diagrams and circuit artifacts – remain challenging for GenAI to interpret.[2] Another challenge arises with graph-level analysis, which is ubiquitous in EDA workflows given the underlying hypergraph representation of circuits. Today's leading methods in graph ML, e.g., for drug discovery or theorem proving, typically operate independently of LLMs, and integrating LLM-based methods with graph ML frameworks remains an open challenge.[3]

Scarce and proprietary data. IC design data is costly to produce, and high-quality public data is scarce [22] [39]. This limits opportunities to train (sharable) large-scale ML models. The proprietary nature of IC design-related data further complicates unsupervised learning and data ownership, and brings many legal and ethical complexities. For example, EDA licenses prevent the upload to public LLMs of tool outputs, as well as any sharing of benchmark results for EDA tools. Process design kit (PDK) data, commercial library and soft IP data, and EDA vendor data (tool documentation, logfiles, reports, command syntax, etc.) are all unsharable, per the IP rights assertions of companies that have spent billions developing these technologies. Building a public IC design foundation model that integrates leading commercial EDA tools and workflows is difficult under current constraints, leaving the industry to balance between protecting intellectual property and promoting innovation through shared data and models.

[2]Several startups have emerged to generate generic CAD samples beyond circuits (e.g., for architectural, engineering drawings, etc.) [68] [76]. However, no commercially viable product has been realized as of this writing.

[3]For example, AlphaFold 3 [1] employs a diffusion-based approach rather than relying on LLMs. How LLM and graph ML approaches might be merged in practical implementations is currently unclear even in relatively well-researched domains such as drug design, and more so for EDA applications.

Data quality. Even with large amounts of data, the development of superior ML models is not guaranteed. Several challenges still prevent the effective use of data in training ML. Data can become outdated or "stale", or incomplete in its coverage, given the rapid evolution of IC technology, design enablements, design tools, and product designs themselves. These gaps lead to biases and inability to generalize. Other vulnerabilities such as data poisoning can also degrade the reliability and performance of ML models.

2.3 Ongoing Efforts

Efforts in both academia and industry have been directed towards building ML infrastructure and shared datasets to enhance ML research and applications in IC design. These initiatives have captured significant international attention.

Academic efforts. To overcome data scarcity, academic works such as [47] [26] have proposed artificial netlist generators that match characteristics of real-world designs, to enhance ML modeling for IC PD applications. The open-source OpenROAD toolchain [4] [57] continuously updates its baselines and benchmarks, and has been the basis of works such as [8] [52] [28] as well as multiple academic contests. ML EDA formats, datasets and "proxy" design enablements (e.g., [9]) to unblock ML for PD are also seen in efforts of the IEEE CEDA DATC [7] [55]. The SLICE project [59], an outgrowth of the March 2023 NSF workshop [39], also seeks to develop a sharable and extensible ML infrastructure toward open collaboration and standardized data-sharing practices. Numerous aligned efforts span reproducibility badge initiatives at MLCAD-2024 to a recent "ImageNets" for EDA workshop [38].

Industry efforts. Notable industry-driven projects and collaborations include the recent GT-NVIDIA contest [40], which sought global engagement to enable LLM-assisted design automation. Si2's "AI/ML Schema Open Standards Working Group" (OSWG) is developing a standardized schema specification to support AI/ML methods and advanced data analytics, enabling academia-industry collaboration that avoids disclosure of proprietary data [58]. Google researchers open-sourced an Ariane RISC-V core in protobuf format, as part of "Circuit Training" [35] [53]; this enabled scaled versions in LEF/DEF to be published [7] [60] as new PD benchmarks that reflect sub-10nm (apparently, TSMC 7nm) process technology.

Vision: An AI flywheel. An overarching goal is to realize an "AI Flywheel" for IC design, so as to achieve the explosive innovation that has been created through "frictionless reproducibility" [11], Hugging Face, and other hallmarks of the broader AI/ML domain. In the AI Flywheel, increased participation from users will generate more data, which in turn drives the development of superior ML models. These improved models enable the creation of more efficient tools and high-performance chips. As better chips attract more users, the cycle accelerates, creating a self-reinforcing loop that continuously advances and transforms the field.

Implicit in "frictionless reproducibility", which entails data sharing, code sharing and competitive challenges [11], is benchmarking. We note that development and use of benchmarks for ML models in IC PD will require careful consideration of relevance and scalability. For instance, techniques validated on smaller designs, or on older technology nodes, may not directly transfer to larger designs or more advanced nodes. To this end, benchmarks must be periodically updated and diversified to reflect a broad range of technology nodes, design complexities, and real-world scenarios. Continuous curation is essential to ensure that ML models remain both accurate and robust. Additionally, benchmarking should prioritize representative datasets, consensus evaluation metrics, and standardized testing protocols to facilitate meaningful comparisons and drive progress. Fortunately, there is a groundswell of efforts – including open and proxy design enablements, open-source tools and open-source designs – to enable this future.

3 ML DEPLOYMENT

In this section, we begin by examining high-level strategy elements for ML deployment in an IC design organization. We then discuss two further aspects of ML deployment: (i) key performance indicators (KPIs) and "checklists" that measure progress and systematically assess project feasibility, and (ii) machine learning operations (MLOps) that unite ML with software and data engineering skillsets to support deployment.

For successful deployment of ML in IC design organizations, de-risking and building management confidence are essential. To this end, three basic strategy elements are as follows.

- *Focus on optimizing existing design processes.* Examples could include tuning synthesis knobs, leveraging netlist and flow analytics, or autotuning tool parameters – e.g., to obtain simplified versions of commercial offerings such as Cadence Cerebrus [71] or Synopsys DSO.ai [79]. These offer measurable improvements while building trust in the integration of ML. Early wins or "short ropes" can be pursued by using supervised learning and predictive models in areas where domain experts already have insights into appropriate feature sets and loss functions.
- *Aim for incremental improvements* (as opposed to high-risk, large-scale methodology shifts). "Bolt-on" enhancement of existing methodologies with ML brings less risk, as it minimizes resource costs and allows for simpler rollbacks if needed.
- *Treat data as a first-class, up-front concern* – as robust data management forms the foundation for success of any ML initiative.

Beginning With the End in Mind. IC design organizations must also understand that deploying ML at scale and in production is substantially different from developing research prototypes. An August 2022 Gartner report noted that only 54% of AI projects successfully transit from prototype to production in organizations with some level of AI experience [13]. It has also been estimated that over 80% of AI projects fail, which is twice the failure rate of corporate information technology (IT) projects without AI involvement [21] [44]. In this light, perhaps the most critical guiding principle is the second of Covey's "seven habits": *Begin with the end in mind.*

Successful deployment of ML in IC design requires a clear understanding of the end-goal objectives and the key factors driving its adoption. (i) Misunderstandings of the project's intent and objectives are among the most common reasons for failures. For example, while GenAI tools such as chatbots and copilots have the potential to enhance productivity, their primary role in IC design should be to enhance design methodologies and improve heuristics, rather than attempting to replace existing EDA flows. (ii) Successful projects should focus on the problem to be solved instead of the technology to be used. Since every ML approach has inherent limitations, it is crucial to select the ML approach that best fits the problem's requirements, rather than simply pursuing the latest and most advanced ML innovations. (iii) ML projects need time and patience to achieve success. Keeping the end goals in mind helps balance the development of long-term capabilities and the pursuit of short-term gains. (iv) A clear understanding of the ultimate goals enables efficient allocation of limited resources to foundational infrastructures, such as cloud computing services, accelerated computing platforms, and standardized metrics collection and design process recording [20]. Ultimately, a well-defined vision, coupled with clear strategic objectives and realistic expectations, is the cornerstone for the successful deployment of ML in EDA.

3.1 KPIs and Checklists

Key Performance Indicators. KPIs enable objective assessment and tracking of progress toward expected outcomes, and provide feedback to help adjust ensuing project phases. Commonly used KPIs for ML projects include [3] [81] (i) operational efficiency KPIs (latency, error rate, accuracy rate) that measure how ML improves business processes; (ii) customer or user satisfaction KPIs (system latency, adoption rate, frequency of use, abandonment rate, service quality) that measure how ML tools enhance user engagement and satisfaction; and (iii) revenue growth KPIs (contributions to sales) that measure AI-driven improvements to sales and marketing. KPIs specific to ML deployment in IC design and PD teams might include improvements in license utilization or efficiency of license usage; number of RTL or P&R iterations achieved per week; ratio of (automated vs. human) explorations of floorplan or timing closure recipes; etc.

Checklists. When considering any potential ML project, feasibility must be systematically assessed. Since ML projects are data-driven and different ML techniques have different data requirements, a first checklist applies to data and ML methods.

- *Does the desired output follow from all the input data?* Plans should be checked for implicit assumptions and for human biases or unjustified optimism. It can be a red flag if ML models seem to "magically" solve complex problems (e.g., NP-hard optimizations, partial differential equations, etc.). ML can provide approximate solutions, but absent external function or solver calls is limited in providing exact answers.
- *Does the training data fully capture the functionality?* ML models rely on quality and diversity of training data. Uncovered corner cases, biases in training data, distribution shifts and other factors can degrade model performance.
- *Does a given ML method work well with a given data type?* For example, LLMs handle textual information efficiently but struggle with numerical data and graphs. Similarly, deep learning methods may not do well with large structured and/or multiscale data.
- *Is there enough training data to train a given ML model?* Different ML approaches vary significantly in their data requirements. GenAI and deep RL methods are data-hungry, while gradient-boosted decision trees (e.g., XGBoost) leverage ensemble techniques to perform effectively on relatively small datasets. Bayesian methods are also effective with limited data.
- *Is there too much data for a given ML model?* Overwhelming an ML model with excessive data may reduce efficiency. Techniques

such as Retrieval-Augmented Generation (RAG) or Mixture of Experts can dynamically manage data and improve scalability.
- *Are the model's training speed and cost consistent with updates to data?* ML models should be updated as data changes. In the IC design context, the retraining cost needed for adaptation to new tools, design ECOs, updated library models or technology nodes, etc. must be considered.

Other checklist items apply to the outputs from ML models.
- *What are the comparisons to existing baselines?* Making a fair comparison between ML methods and robust baselines is essential to accurately measure improvement. A well-noted pitfall is the lack of strong baselines and public, reproducible benchmarking.[4]
- *How do output errors scale with size?* For instance, an ML model might perform well on simple designs but fail to handle designs with millions of instances, hundreds of macros, or tight utilization and timing requirements.
- *Must output errors be found?* ML models deployed at final, signoff stages of physical design must detect all errors accurately to avoid compromising the tapeout. By contrast, ML models deployed during early design space exploration and implementation flow steps might tolerate some inaccuracy.
- *How are output errors tolerated?* Different applications require different error-handling strategies: "verify and fail," "verify and retry," or hot-patching mechanisms such as hallucination correction and statistical adjustment.
- *Will model outputs be consumed by people?* Outputs for human consumption should be concise, clearly confirmable as well-formed (and correct), easy to fix, and timing-saving for human engineers.
- *Will model outputs be consumed by tools?* For outputs that will be fed directly to tools, questions such as the following arise: (i) are there too many errors that will stop tools in their tracks – especially in large designs?; (ii) are there efficient checkers and correctors that handle model output errors automatically?; and (iii) does the ML model output ultimately improve quality of results from the design process?

By systematically checking such aspects, the deployment and evaluation of ML models in EDA can achieve greater robustness and reliability, paving the way for transformative improvements in design processes.

3.2 Machine Learning Operations

Machine Learning Operations (MLOps) integrate core ML activities such as model development, testing, and deployment with best practices from software engineering (DevOps) and data engineering [83]. Historically, large enterprises (Databricks, Snowflake, Amazon etc.) offered end-to-end MLOps platforms that encompassed continuous integration/delivery, advanced observability, and infrastructure management. Early tools such as Weights & Biases (WandB) [82] and TensorBoard [64] played a foundational role in establishing observability and monitoring within MLOps pipelines that catered to the individual software engineer, beginning a trend towards lowering the barriers of entry and subsequent commoditization. Today, the growing influence of Large Language Models (LLMs) is driving a further shift towards commoditization, enabling organizations of various sizes to adopt MLOps more readily. With increasingly specialized domains such as chip design likely to incorporate ML-based optimizations in the future, organizations should consider these operational principles proactively. To guide such integration, the following checklist highlights key considerations for introducing MLOps practices into chip design environments.
- *How will data be archived from runs?* Determine whether data should be archived per-run (streaming) or in larger aggregated batches.
- *What elements of run data?* Identify which logs, scripts, reports, and collaterals are essential to retain.
- *How to manage the lifecycle of design data, or the model store?* Establish policies for creating, updating, and deprecating stored models and relevant datasets.
- *Who creates VectorDBs (for RAG) and fine-tuned models – and when?* Clearly define team responsibilities and timelines for creating and maintaining these resources.
- *Where is the compute?* Consider data volume, privacy, latency, and overall complexity when selecting on-premise or cloud resources.

How do LLMs aid commoditization? As LLM-based services come to the fore, a new paradigm emerges – LLMOps [42]. Fundamentally, deploying a LLM via API is a much more lightweight process than training, fine-tuning, validating and deploying a classical ML model, requiring much less Ops-level expertise. What remains is observability, middleware integration, error tracking and so forth, all of which is now being commoditized under LLMOps. Service providers such as OpenAI handle routing, batching, and load balancing, while newer tools such as LangSmith [63] streamline the creation and maintenance of LLM-centric workflows. The combined effect is that formerly enterprise-only capabilities, ranging from continuous integration/delivery to comprehensive observability, have become more widely accessible. By offloading substantial operational burdens to external platforms, organizations can reduce infrastructure requirements and accelerate ML adoption without incurring the overheads once needed to stand up full-scale MLOps environments. This commoditization is heightened by plugins such as LiteLLM [65] that build atop existing pipelines, e.g., those from OpenAI or Databricks. The relatively low prices of LLMOps have a knock-on effect on pricing of other MLOps tools.

The commoditization of MLOps has fostered a more level competitive landscape. Platforms such as DataDog [74] and enterprise-grade frameworks such as MLflow (Databricks) and SageMaker (Amazon) have begun to incorporate advanced LLM features. As a result, smaller organizations that could not previously afford the complexity and cost of robust ML infrastructures can now piece together leaner, modular solutions from both newer entrants (e.g., LangSmith, DataDog) and older established solutions (e.g., Amazon). ML-driven innovation and experimentation are thus more accessible to smaller enterprises, including both MLOps vendors and customers. Now, a broader range of entities – including those in the IC design ecosystem – can leverage state-of-the-art ML methodologies for improved latency, robustness and observability at lower operational thresholds.

[4]This lack can lead to overestimated performance and controversy (cf. [35] [34]). The crucial role of "code with data" (i.e., reproducibility) and benchmarking as foundations of technical progress is highlighted in Google's research philosophy [46] and such influential publications as [11] [37].

Despite the overall positive economic effects of LLM development and LLMOps on the MLOps sector, it is not straightforward to merge LLMOps with IC design and EDA. LLMOps tools are ultimately specialized for a particular ML paradigm – that of LLMs. As we discuss next, it is still not clear how to integrate the most cutting edge LLM agents with EDA codebases and create cross-domain deployments that bridge IC design and LLMs.

4 CHALLENGES FOR LLM DEPLOYMENT

Over the past two years, approaches based on LLMs have exerted a substantial influence on software engineering workflows, yielding a diversity of automated tools that assist human developers. Code completion, type checking, and debugging represent common use cases. GitHub Copilot [85], along with recently emerged startups such as Magic [80], Anysphere [67], and Cognition [84] (all attaining valuations exceeding one billion dollars), has introduced LLM-based solutions that operate at varying levels of granularity. For instance, Cognition's agent "Devin" can generate comprehensive pull requests from scratch, request clarification from the developer, execute intermediate analyses through shell or browser-based runs for debugging, and devise longer-term development plans. In contrast, tools such as Copilot and Cursor [73] (from Anysphere) focus primarily on smaller, more precise modifications (e.g., debugging, completion of isolated code segments). Intermediate-scale players such as Supermaven [86] (now integrated into Anysphere) and Aider [61] occupy positions between these two extremes. However, the deployment of LLMs in EDA software is difficult. We now expound on the various obstacles, arising from codebase-level architectural differences, that give rise to these issues and make LLMs less than ideal for EDA at the moment.

4.1 LLM $\times$ EDA: Software Engineering Issues

LLM-driven models exhibit their strongest performance in widely prevalent programming languages (e.g., Python, React) and in contexts where structure is standardized and broadly observed across codebases. They rely heavily on canonicalized structural representations, such as JSON-mode or XML-based outputs. Deviations from such established formats introduce rapid error propagation. For example, even rendering LaTeX, with its more stringent formal syntax and less JSON-like structure, presents increased difficulty for LLMs. Similarly, one may define the "depth" of a JSON object as the maximum hierarchical level at which a leaf node resides. LLMs characteristically favor "shallower" structures, and when operating at scale—spanning large codebases—they generally fail to maintain reliable performance. Current mainstream LLM agents perform best on loosely typed, medium-scale monorepositories comprising at most two widely adopted languages in a common stack (e.g., a MERN stack), and they particularly struggle when asked to handle extended reasoning chains across an entire, more complex codebase.

This scenario, when applied to EDA codebases, effectively forms a near-complete "anti-endorsement" for the use of LLMs in that domain. State-of-the-art EDA tools, whether from Cadence or Synopsys, remain closed-source. The LLM cannot inspect underlying source code to diagnose errors, undermining the established LLM-based debugging paradigm that relies on source-level observability. The notion of a "hybrid" action-observation workflow – iteratively modifying source code and then observing the resulting behavior in a fully transparent repository – is thus obstructed. Modeling proprietary EDA tools as black-box APIs might appear viable, but in practice these tools differ significantly from standard web-based APIs in their payload structures, response latencies, and computational overheads.

Considering an open-source alternative such as OpenROAD does not fully resolve these issues. OpenROAD is predominantly written in C++ with TCL and some Python as extension (scripting) languages, for a Verilog-to-GDSII (or, -routed DEF) use domain. Although the TCL scripts constitute a small fraction of the codebase's lines of code, they exert substantial influence on the workflow's execution. This structural configuration is atypical and differs markedly from the software environments upon which LLMs are predominantly trained. Neither Verilog nor TCL shares the same order-of-magnitude popularity as more conventional languages found in widely available training corpora.[5] Errors in the OpenROAD flow (ORFS) may originate from virtually any point in the repository.

Besides, in contrast to a webserver's APIs, which are often neatly modularized, EDA toolflows are complex and highly interdependent. Disrupting any stage of the RTL-to-GDSII flow – ranging from synthesis through detailed routing – may cause failures. Unlike in a setting such as React, where incremental changes yield immediate feedback and can be traced to a specific offending line using built-in browser tooling, OpenROAD errors often surface as terminal messages devoid of clear causality. Worse, the flow can fail silently, producing a final circuit that meets completion criteria but exhibits suboptimal Power, Performance, and Area (PPA) metrics. Such silent failures are rare in typical software development and, when present, are generally easier to diagnose. (We note that making any analogous commentary on commercial EDA tools is prohibited by terms of their license agreements.)

4.2 Challenges from EDA Flows

The current EDA flows present several obstacles to the adoption of LLMs at scale. First, the complexity of EDA output formats poses great challenges for LLMs. The structural requirements inherent in EDA workflows greatly surpass the relatively simple adherence to JSON-based formats. LLMs, which already struggle with deeply nested JSON outputs, currently lack foundation models for more complex data forms such as hypergraphs (i.e., circuit netlists). Such representational gaps severely limit the applicability of LLM-based methods in IC design and EDA contexts.

Second, tool latencies and iterations seen in EDA flows substantially diminish the practicality of LLM-driven copilot techniques. Consider, for example, Cognition's Devin operating on a front-end repository using React: it might request clarification from the human expert every five minutes, with each cycle yielding meaningful progress. An LLM-based EDA agent, by contrast, may need to wait anywhere from minutes to days for observable results of its source code-level changes – and then decide whether the results

[5] While RTL copilots are being developed (e.g., Silimate [78]), achieving robust and generalizable performance may be challenging.

require clarification. Such prolonged and variable iteration cycles are frustrating impediments to development.

Third, Devin and other LLM-based agents will also struggle when confronted with code dispersed across multiple repositories or when incorporating unknown black-box tools – but such scenarios are pervasive in IC design and EDA. The introduction of a new Process Design Kit (PDK) or design database can be as disruptive as adding a new framework, a scenario for which current LLM agents are ill-prepared. Finally, the ecosystem of EDA and IC design relies heavily on manuals and siloed documentation formats, rather than the docstrings and inline comments that are standard in modern software engineering practices.

State of Play. As of late 2024, integrating LLMs with EDA remains challenging both technically and culturally. Technical barriers include incompatible data formats, lack of standard code structures, and the monolithic nature of EDA repositories, which differ greatly from the microservices-driven architecture of modern software engineering. The absence of modularity and composable APIs (cf. the "Bezos API mandate") further complicates LLM applications in EDA.

In software engineering, modular APIs and microservices enable developers to quickly integrate and scale systems. Frameworks such as React and Python packages simplify development, while backend tasks can be outsourced to services such as Firebase [75]. Projects often start with minimal capital, sometimes as open-source hobby efforts, and grow through subscriptions. Generous startup credits provide millions in free resources, reinforcing a flywheel of open-source contributions and economic incentives. This ecosystem attracts a diverse range of participants, including outsiders such as finance professionals turned entrepreneurs. EDA, in contrast, operates with monolithic, tightly coupled frameworks that lack modularity or open standards. It demands high capital investment, functions in oligopolistic markets, and provides few opportunities for entry. This cultural and economic disparity creates significant hurdles, resulting in technical complexity and "culture shock" for ML engineers accustomed to the agility of web-oriented development environments.

5 CONCLUSIONS

In this paper, we have presented several perspectives on application selection and deployment of ML in IC physical design. Despite some progress in recent years, the adoption of AI/ML in production remains limited. We outline the challenges inherent in IC design data, along with ongoing efforts to mitigate these challenges. Our discussion includes high-level precepts for successful deployment of AI/ML in IC design, e.g., "begin with the end in mind"; the use of well-chosen KPIs to assess and track progress; and basic "checklists" for effective ML deployment. We furthermore discuss potential challenges and opportunities for LLM-enhanced EDA and IC design methods, where integrating software and data engineering practices such as MLOps and LLMOps can automate and standardize processes across the ML lifecycle.

Through replication of the "AI flywheel" and culture of frictionless reproducibility seen in the broad AI/ML community [11], the IC design and EDA ecosystem has the potential to break free from its constraints, enabling seamless ML integration. Unified frameworks, analogous to PyTorch or TensorFlow, could promote modular, reusable workflows and move away from monolithic codebases. Standardized benchmarks, such as the recent "ImageNets" for EDA [38] effort, would provide better progress metrics and encourage rapid experimentation. Open-source datasets and models, coupled with user-friendly tools, could lower entry barriers and potentially reduce reliance on proprietary data. Beyond technical innovations, cultural shifts toward openness and collaboration will bridge gaps between EDA and ML experts, driving the field forward. While an "AI singularity" for EDA is likely years away, advances that the community can anticipate include (i) AI agents boosting engineering productivity, and (ii) AI agents learning from human-driven innovations to optimize design processes.

ACKNOWLEDGMENTS

This paper draws from the "Deployment" section of an IEEE Learning Network webinar, "Artificial Intelligence and Machine Learning in Chip Design", presented by the second-listed author (ABK) in August 2024. For that webinar, many colleagues (notably, Igor Markov and Thomas Andersen at Synopsys (data and checklists), Scot Weber at AMD (MLOps), Rod Thorne and Steve Brown at Cadence, Siddhartha Nath at Intel, and Tuck-Boon Chan at Qualcomm, among others) generously provided discussion and inputs. We thank Professor Youngsoo Shin of KAIST and the ASP-DAC 2025 committee for organizing this special session and inviting this paper.

REFERENCES

[1] J. Abramson, J. Adler, J. Dunger et al., "Accurate Structure Prediction of Biomolecular Interactions with AlphaFold 3", *Nature*, 2024, pp. 493-–500.
[2] OpenAI, J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. Leoni Aleman, D. Almeida et al., "GPT-4 Technical Report", *arXiv preprint 2303.08774*, 2023.
[3] N. Aggarwal and A. Liu, "KPIs for Gen AI: Why Measuring Your New AI Is Essential to Its success", 2023. https://cloud.google.com/transform/kpis-for-gen-ai-why-measuring-your-new-ai-is-essential-to-its-success
[4] T. Ajayi, V. A. Chhabria, M. Fogaça et al., "Toward an Open-Source Digital Flow: First Learnings from the OpenROAD Project", *Proc. DAC*, 2019, pp. 1–4.
[5] M. Chen, S. Mei, J. Fan, and M. Wang, "An Overview of Diffusion Models: Applications, Guided Generation, Statistical Rates and Optimization", *arXiv preprint 2404.07771*. 2024.
[6] M. Chen and T.-C. Wang, "A Hypergraph Partitioner Utilizing a Novel Graph Generative Model", *Proc. ICCAD*, 2024.
[7] V. A. Chhabria, V. Gopalakrishnan, A. B. Kahng, S. Kundu, Z. Wang, B.-Y. Wu and D. Yoon, "Strengthening the Foundations of IC Physical Design and ML EDA Research", *Proc. ICCAD*, 2024.
[8] V. A. Chhabria, W. Jiang, A. B. Kahng, R. Liang et al., "OpenROAD and CircuitOps: Infrastructure for ML EDA Research and Education", *Proc. VTS*, 2024.
[9] 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. CAD* 43(4) (2024), pp. 1218–1231.
[10] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding", *Proc. ACL*, 2019.
[11] D. Donoho, "Data Science at the Singularity", *Harvard Data Science Review*, 2024. https://doi.org/10.1162/99608f92.b91339ef
[12] H. Esmaeilzadeh, S. Ghodrati, A. B. Kahng, J. K. Kim et al., "An Open-Source ML-Based Full-Stack Optimization Framework for Machine Learning Accelerators", *TODAES* 29(4) (2024), pp. 68:1–68:33.
[13] "Gartner Survey Reveals 80% of Executives Think Automation Can Be Applied to Any Business Decision", Gartner, August 2022.

[14] I. J. Goodfellow, J. P. Abadie, M. Mirza, B. Xu, D. W. Farley, S. Ozair, A. Courville and Y. Bengio, "Generative Adversarial Nets", *NeurIPS* 27 (2014).
[15] J. Ho, A. Jain and P. Abbeel, "Denoising Diffusion Probabilistic Models", *Proc. NeurIPS*, 2020, pp.6840–6851.
[16] H.-H. Hsiao, Y.-C. Lu, P. Vanna-Iampikul and S. K. Lim, "FastTuner: Transferable Physical Design Parameter Optimization using Fast Reinforcement Learning", *Proc. ISPD*, 2024, pp. 93–101.
[17] J. Hu and A. B. Kahng, "The Inevitability of AI Infusion Into Design Closure and Signoff", *Proc. ICCAD*, 2023, pp. 1–7.
[18] J. Hu and H. Ren, eds., *Machine Learning Applications in Electronic Design Automation*, Springer, 2022.
[19] G. Huang, J. Hu, Y. He, J. Liu, M. Ma, Z. Shen et al., "Machine Learning for Electronic Design Automation: A Survey", *TODAES*, 26(5) (2021), pp. 1–46.
[20] 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.
[21] J. Kahn, "Want Your Company's A.I. Project to Succeed? Don't Hand It to the Data Scientists, Says This CEO", 2022. https://fortune.com/2022/07/26/a-i-success-business-sense-aible-sengupta/
[22] A. B. Kahng, "A Mixed Open-Source and Proprietary EDA Commons for Education and Prototyping", *Proc. ICCAD*, 2022, pp. 1–6.
[23] A. B. Kahng, "Solvers, Engines, Tools and Flows: The Next Wave for AI/ML in Physical Design", *Proc. ISPD*, 2024, pp. 117–124.
[24] A. B. Kahng, S. Kundu and D. Yoon, "Placement Tomography-Based Routing Blockage Generation for DRV Hotspot Mitigation", *Proc. ICCAD*, 2024.
[25] A. B. Kahng and Z. Wang, "ML for Design QoR Prediction", in *Machine Learning Applications in Electronic Design Automation*, Springer, 2022.
[26] D. Kim, S. Y. Lee, K. Min and S. Kang, "Construction of Realistic Place-and-Route Benchmarks for Machine Learning Applications", *IEEE Trans. CAD* 42(6) (2022).
[27] V. Lee, C. Deng, L. Elzeiny, P. Abbeel and J. Wawrzynek, "Chip Placement with Diffusion", *arXiv:2407.12282*, 2024.
[28] R. Liang, A. Agnesina et al., "CircuitOps: An ML Infrastructure Enabling Generative AI for VLSI Circuit Optimization", *Proc. ICCAD*, 2023, pp. 1–6.
[29] R. Liang, S. Nath, A. Rajaram, J. Hu and H. Ren, "BufFormer: A Generative ML Framework for Scalable Buffering", *Proc. ASP-DAC*, 2023, pp. 264–270.
[30] R. Liaw, E. Liang, R. Nishihara et al., "Tune: A Research Platform for Distributed Model Selection and Training", *arXiv preprint 1807.05118*, 2018.
[31] D. S. Lopera, L. Servadei, G. N. Kiprit et al., "A Survey of Graph Neural Networks for Electronic Design Automation", *Proc. MLCAD*, 2021, pp. 1–6.
[32] Y.-C. Lu, S. Nath, V. Khandelwal and S. K. Lim, "Doomed Run Prediction in Physical Design by Exploiting Sequential Flow and Graph Learning", *Proc. ICCAD*, 2021, pp. 1–9.
[33] Y.-C. Lu, T. Yang, S. K. Lim and H. Ren, "Placement Optimization Via PPA-Directed Graph Clustering", *Proc. MLCAD*, 2022, pp. 1–6.
[34] I. L. Markov, "Reevaluating Google's Reinforcement Learning for IC Macro Placement", *Communications of the ACM*, 2024.
[35] A. Mirhoseini, A. Goldie, M. Yazgan et al., "A Graph Placement Methodology for Fast Chip Design", *Nature* 594 (2021), pp. 207–212.
[36] S. Nath, G. Pradipta, C. Hu, T. Yang, B. Khailany and H. Ren, "TransSizer: A Novel Transformer-Based Fast Gate Sizer", *Proc. ICCAD*, 2022, pp. 1–9.
[37] National Academies of Sciences Engineering and Medicine, *Reproducibility and Replicability in Science*, Washington, DC, The National Academies Press, 2019. https://www.nap.edu/catalog/25303/reproducibility-and-replicability-inscience
[38] NSF Workshop on "ImageNets" for EDA, November 2024. https://wp.nyu.edu/imagenets_eda/
[39] NSF Workshop on Shared Infrastructure for Machine Learning EDA, March 2023. https://sites.google.com/view/ml4eda/home
[40] Nvidia Corp. and Georgia Tech, "ICCAD Contest on LLM-Assisted Hardware Code Generation", *ICCAD*, 2024. https://nvlabs.github.io/LLM4HWDesign/
[41] R. Qiu, G. L. Zhang et al., "AutoBench: Automatic Testbench Generation and Evaluation Using LLMs for HDL Design", *Proc. MLCAD*, 2024, pp. 1–10.
[42] A. Quinn, "LLMOps vs. MLOps: Understanding the Differences", 2024. https://www.iguazio.com/blog/llmops-vs-mlops-understanding-the-differences/
[43] M. Rapp, H. Amrouch, Y. Lin, B. Yu, D. Z. Pan, M. Wolf and J. Henkel, "MLCAD: A Survey of Research in Machine Learning for CAD Keynote Paper", *IEEE Trans. on CAD*, 41(10) (2022), pp. 3162–3181.
[44] J. Ryseff, B. F. De Bruhl and S. J. Newberry, "The Root Causes of Failure for Artificial Intelligence Projects and How They Can Succeed", 2024. https://www.rand.org/pubs/research_reports/RRA2680-1.html
[45] U. Sharma, B.-Y. Wu, S. R. D. Kankipati, V. A. Chhabria and A. Rovinski, "OpenROAD-Assistant: An Open-Source Large Language Model for Physical Design Tasks", *Proc. MLCAD*, 2024, pp. 1–7.
[46] A. Spector and P. Norvig, "Google's Hybrid Approach to Research", *Communication of the ACM*, 55(7), (2018), pp. 34–37.
[47] D. Stroobandt, P. Verplaetse and J. V. Campenhout, "Generating Synthetic Benchmark Circuits for Evaluating CAD Tools", *IEEE Trans. CAD* 19(9) (2000).
[48] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N Gomez, L. Kaiser and I. Polosukhin, "Attention is All You Need", *Proc. NeurIPS*, 2017, pp. 6000–6010.
[49] 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.
[50] K. Xu, R. Qiu, Z. Zhao, G. L. Zhang, U. Schlichtmann and B. Li, "LLM-Aided Efficient Hardware Design Automation", *arXiv preprint 2410.18582*, 2024.
[51] ASP-DAC 2024 Tutorial. https://github.com/ASU-VDA-Lab/ASP-DAC24-Tutorial
[52] CircuitOps Repository. https://github.com/NVlabs/CircuitOps
[53] Google Brain Ariane testcase (protobuf). https://github.com/google-research/circuit_training/tree/main/circuit_training/environment/test_data/ariane
[54] IEEE CEDA DATC RDF 2023. https://github.com/ieee-ceda-datc/RDF-2023
[55] IEEE CEDA DATC Robust Design Flow, 2024. https://github.com/ieee-ceda-datc/Robust-Design-Flow
[56] IEEE Intl Workshop on LLM-Aided Design (LAD'24). https://www.islad.org
[57] The OpenROAD Project. https://github.com/The-OpenROAD-Project/
[58] Si2, AI/ML Schema Open Standards Working Group (OSWG). https://si2.org/q4-2024-ceo-message/
[59] SLICE: A Shared Machine Learning Infrastructure for the EDA Community. https://slice-ml-eda.github.io
[60] TILOS AI Institute MacroPlacement GitHub repository. https://github.com/TILOS-AI-Institute/MacroPlacement
[61] Aider. https://aider.chat/
[62] Ansys SimAI. https://www.ansys.com/products/simai
[63] LangSmith by LangChain. https://www.langchain.com/langsmith
[64] TensorBoard by TensorFlow. https://www.tensorflow.org/tensorboard
[65] LiteLLM by BerriAI. https://www.ycombinator.com/companies/berriai
[66] Anyscale. https://www.anyscale.com/
[67] Anysphere Inc. https://anysphere.inc/
[68] ArchiLabs. https://www.archilabs.ai/
[69] Astrus AI. https://www.astrus.ai/
[70] Cadence Allegro X AI. https://www.cadence.com
[71] Cadence Cerebrus. https://www.cadence.com
[72] Claude, 2023. https://www.anthropic.com/
[73] Cursor. https://www.cursor.com/
[74] DataDog LLM analytics. https://www.datadoghq.com/dg/monitor/llm/llm-observability/
[75] Firebase. https://firebase.google.com/
[76] Hestus Inc. https://www.hestus.co/
[77] Siemens Solido. https://eda.sw.siemens.com/en-US/ic/solido/
[78] Silimate: The Copilot for Chip Designers. https://www.silimate.com
[79] Synopsys DSO.ai. https://www.synopsys.com/ai/ai-powered-eda/dso-ai.html
[80] Magic Inc. https://magic.dev/
[81] "Measuring Success: Key Metrics and KPIs for AI Initiatives". https://chooseacacia.com/measuring-success-key-metrics-and-kpis-for-ai-initiatives/
[82] Weights and Biases. https://wandb.ai/
[83] "What is MLOps?". https://aws.amazon.com/what-is/mlops/
[84] Cognition Inc. https://www.cognition.ai/
[85] Github Copilot. https://github.com/features/copilot
[86] Supermaven. https://supermaven.com/