Invited Paper: The Inevitability of AI Infusion Into Design Closure and Signoff

Jiang Hu
Texas A&M University
College Station, TX, USA
jianghu@tamu.edu

Andrew B. Kahng
UC San Diego
La Jolla, USA
abk@ucsd.edu

***Abstract*—SoC design teams embrace new technologies and methodologies that bring clear value. Given this, future infusion of AI into design closure and signoff is inevitable. Predictive AI models help focus the application of last-mile incremental optimizations (sizing, placement and routing) to achieve timing and noise closure; successful examples range from routing-free crosstalk prediction to timing/power evaluation in early RTL development. Design closure becomes more efficient when "imperfect but fast" ML inferencing is used to filter out potential violations, which can then be passed to golden analysis tools. Learning methods also improve the design process in many ways, ranging from smarter PVT corner selection to predicting the CPU and memory usage of signoff tools. At a higher level, AI will help design teams learn to avoid design trajectories that lead to time-consuming closure and signoff iterations. This talk will provide a broad overview of directions in which AI will inevitably improve the cost and efficiency of signoff in the coming years.**

I. Introduction

SoC design teams always want get to end goals (i.e., tapeout and volume production) with less resources and schedule. Ideally, this is achieved by a single-pass design flow (with no loops back due to violated specifications or design rules) and minimal design guardbanding. In this talk and invited paper, we motivate why signoff analysis and design closure are *inevitable* targets for AI/ML, and discuss potential benefits and limitations of AI/ML in these contexts.

Signoff analysis refers to the "golden", foundry-qualified verifications of physical/geometric, manufacturability, timing, power and reliability attributes. Signoff is the final gatekeeper at the handoff from design to manufacturing. Importantly, product owners – as well as the fabless-foundry business framework – *require* the use of golden signoff tools at tapeout. In this context, AI/ML seeks to minimize turnaround times and license demand by shifting cost-accuracy tradeoffs, as cartooned in Figure 1. This can be achieved with a variety of mechanisms, such as (i) improved (multi)physics modeling; (ii) learning and extrapolation from data; and (iii) early filtering or triage to distinguish "safe" (not of concern) versus "at-risk" (needing greater attention) elements of the design. We note that multiscale-multiphysics analyses – e.g., thermo-mechanical stress and reliability analysis of a heterogeneously-integrated module – demand nearly 1000000× speedups over current methods. All of these bring opportunities for AI/ML.

Design closure makes the design *safe* for signoff analysis: designers do not want loops in their flow due to failed signoff runs. AI/ML for design closure centers on prediction, especially, of future failure. Here, the core challenge arises from the tension between "range" of lookahead into the future, and accuracy of predictions: What can we predict (e.g., post-route crosstalk noise violations), from what earlier state in the design process (e.g., from the post-physical synthesis netlist), without unacceptable pessimism (which leaves design quality on the table) or optimism (which incurs costly loops in the flow)? A corollary challenge is how to model the behavior of black-box EDA tools that may contain millions of lines of code. Design closure is also about *optimization* of the design with respect to a given PPA (Performance, Power and Area) objective, while satisfying constraints. AI/ML offers the potential to learn more relevant optimization objectives or more effective metaheuristics (cf. "Learn to Optimize" [27]). As noted in [19], the complexity of design closure is continually exploding, with ever-more modes and corners, physical effects, rules and checks. Thus, AI/ML for design closure always becomes more appealing over time.

Perspectives. Before continuing, we offer the following perspectives. First, improving signoff analysis and design closure brings a *virtuous cycle*: more accuracy, fewer violations and loops, more design exploration within schedule, better design outcomes, more investment, etc. This is true with AI-driven improvements as well, where an "AI data flywheel" effect also applies [33]. Second, several *caveats* arise from industry structure and IP considerations. Inability to share data pertaining to technology, design enablement and EDA tool – along with prohibitions of reverse-engineering of (signoff) EDA tools – may slow development of AI/ML for signoff. Third, there are *limits* to infusion of AI/ML in signoff. Juxtaposing "AI/ML" and "signoff" elicits near-reflexive arguments why these two can never meet in practice. (i) An AI/ML model makes errors in inference, while "signoff" must be golden and error-free, as it underpins the fabless-foundry business relationship. (ii) AI/ML predictions and estimations may not be explainable or easily diagnosable, and may be expensive to improve. This can be a non-starter. (iii) AI/ML models may not generalize or easily transfer to new designs or enablements, both of which evolve rapidly. Despite these limitations, AI/ML techniques are likely to provide substantial value in early alignment toward signoff, i.e., during design implementation and closure.

The following sections give a sampling of AI/ML opportunities in various facets of signoff and design closure, roughly ordered from low-level physics to high-level prediction. Section II addresses PVT corner selection and reduction in signoff. Section III discusses prediction challenges associated with wiring: parasitics, delays and slews, and design rule violations (DRVs). Section IV presents routing-free crosstalk prediction, while Section V discusses prediction of timing and power at the register-transfer level. Section VI touches on the higher-level need for predictions of EDA tool resource usage, and we conclude in Section VII.

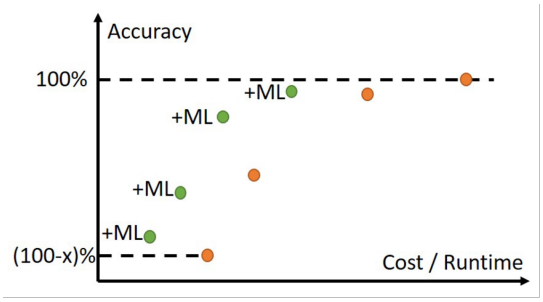

Fig. 1. Accuracy-cost tradeoff in analysis. Figure reproduced from [20].

II. PVT Corner Selection

In today's physical design flow, a significant portion of design turnaround time is spent on timing analysis at various process, voltage and temperature (PVT) *corners*. At final signoff, timing and electrical constraints – as well as power consumption that includes glitch and internal power – must be verified at hundreds of corners, across multiple operational modes. Analyzing more mode-corner combinations within given (compute, license, time) resources is always preferred.[1]

To speed up timing analysis and signoff processes, several machine learning-based approaches have been proposed for *corner selection* in multi-mode multi-corner timing analysis. It is observed in [23] that timing results for a given path at different corners will have strong correlations due to the physics of chained devices and interconnects. The authors propose use of multivariate linear regression to predict timing at *unobserved* corners from analysis results at *observed* corners. The flow in Figure 2 consists of three main steps: (i) *subset selection* uses *greedy deletion* to determine which n corners are most predictive of the remaining N-n corners, for a fixed value of n; (ii) *model training* generates optimal model parameters W^* for each value of n; and (iii) *model inference* predicts timing at unobserved corners from observed corners using the trained model. It affords greater timing accuracy with low overhead, with accumulated value as the design is iterated and reanalyzed throughout the months-long physical implementation process. However, it cannot substitute for the final signoff run that must pass all modes and corners (see Section VI).

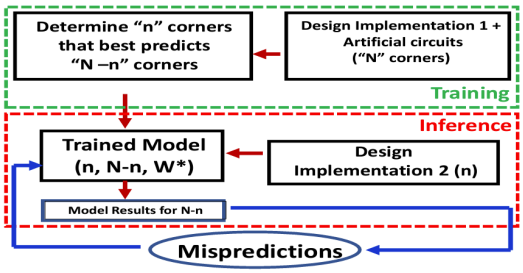

Fig. 2. Modeling flow for "unobserved corner" prediction. Figure reproduced from [23].

More recent works have also addressed the selection and pruning of timing corners. The work of [5] introduces a learning-based framework for predicting circuit path delays at low-voltage corners based on corresponding delays at high voltages. The framework employs a dilated CNN for feature engineering and an ensemble model as the predictor. [39] proposes a dominant corner selection strategy to quickly determine the dominant corner combination, along with machine learning-based models (linear regression, multilayer perceptron network, and random forest) for multi-corner timing prediction. In [7], a learning-based approach is developed to predict path timing for multiple unknown corners at low voltage. It uses long short-term memory (LSTM) to exploit circuit topology correlation with timing and a multigate mixture-of-experts (MMoE) network to capture correlation among all analysis corners.

Two open directions for AI/ML are (i) further improvement of timing corner selection, including in contexts of path-based and SI-aware timing, and "other physics" such as skew corners, temperature inversion, and timing derivatives such as cell internal power; and (ii) generation of a design-specific minimal set of synthetic corners that achieves specified coverage of all corners. The latter can be extended to find synthetic corners that best drive particular SP&R flows to desired outcomes.

III. Wire Parasitic and Delay Prediction

To achieve design closure, timing optimizations are performed throughout the pre-routing stages of logic synthesis, placement, CTS and global routing. During these stages, estimated wire parasitics inform delay and slew estimations, which in turn inform optimization transforms such as remapping, cloning, buffering, sizing and re-placement. For example, logic synthesis may use a wireload model to estimate parasitics, while clock tree synthesis will use Steiner trees and placement will use 3D early global routes.[2] Inevitably, estimated parasitics will differ from the post-detailed routing ground truth values. Such discrepancies cause either pessimistic overdesign (incurring increased power, area, and design cycle time) or optimistic underdesign (incurring failed

[1]Due to practical limitations such as runtime, memory footprint, license resource, or inability of heuristic optimizers to handle too many constraints, only a few mode-corner combinations may be used in early design stages such as logic synthesis or placement. At the same time, the "accuracy" requirement for timing analysis is relaxed in early design stages, e.g., estimating endpoint arrival times to within one percent is usually sufficient for design stages up to early global routing in the placer.

[2]The problem of early parasitics, delay and slew estimation has been well-studied, with early works including [22] [16].

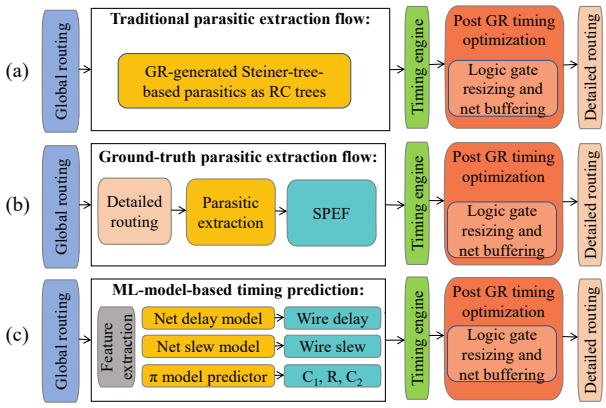

Fig. 3. Three flows that use different parasitic estimates for post-GR timing optimizations: (a) traditional flow (Steiner tree-based RC estimates), (b) ground-truth (DR followed by parasitic extraction to determine post-DR parasitics), and (c) [10] flow (fast ML engine for post-DR parasitic and timing). Figure reproduced from [10].

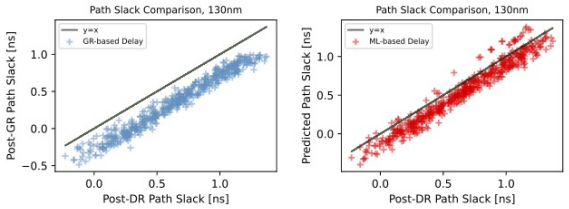

Fig. 4. Path slacks comparison for JPEG 130nm. Figure reproduced from [10].

signoff and loops in the flow). This motivates new methods to reduce parasitic estimation errors relative to post-detailed route ground truth.

Recent works of Chhabria et al. [10], [11] have addressed this challenge by introducing a machine learning-based framework that predicts post-detailed route wire delay, slew, and capacitance from a design that has only undergone global routing. Additionally, the predicted delay and parasitics are used for subsequent timing optimization. Figure 3 illustrates the traditional parasitic extraction flow, the ground-truth parasitic extraction approach, and the ML model-based timing prediction method of [10], [11]. The model uses features such as post-placement HPWL (Half Perimeter Wire Length), number of sinks, slew at driving point, congestion estimates, rise and fall transitions, and source-sink distance. It also considers source-sink R, C values and uses an XGBoost model to predict parasitics and delay. Figure 4 shows the improvement of delay prediction accuracy achieved by the trained ML model versus post-GR estimations, for the JPEG design in a 130nm technology node.

Two open directions for AI/ML in this context are (i) to incorporate predictors of congestion and routability, which strongly determine wiring detours (hence, parasitics, timing and crosstalk) as well as design rule-correctness of final routing; and (ii) addressing the closed loop of estimation and optimization, whereby the iteration between predictors and optimizers must *together* achieve improved design outcomes.

IV. Routing-Free Crosstalk Prediction

Due to the coupling capacitance between two adjacent wires of two different nets, the signal switching of one net causes crosstalk noise to the other net. In addition, coupling capacitance induces extra signal propagation delay depending on switching activities of two neighboring nets. Starting from the deep submicron technologies, the crosstalk effect became very significant and must be considered for signal integrity and timing signoff. However, it is very difficult to tell if an early design step may result in significant crosstalk problems until chip routing, where wire adjacency is finally decided. Due to its importance, crosstalk has been considered in cell placement [30] and even in technology mapping [14]. However, all early crosstalk estimation techniques rely on trial routing, which is time consuming. By machine learning, truly routing-free crosstalk prediction can be achieved [28].

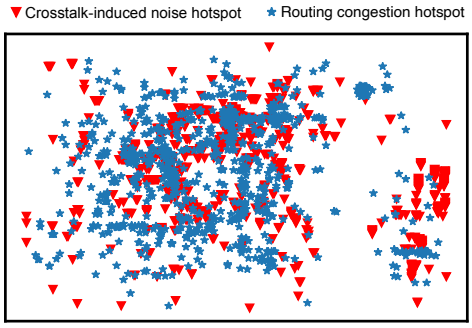

Fig. 5. Congestion hotspots vs. crosstalk hotspots [28].

Although crosstalk and routing congestion are correlated, crosstalk prediction cannot be simply replaced by congestion prediction. Figure 5 illustrates that crosstalk hotspots can be significantly different from routing congestion hotspots. Crosstalk can be evaluated by coupling capacitance, crosstalk noise and crosstalk induced delay, which are correlated but not the same. The Venn diagram in Figure 6 demonstrates that nets with large coupling capacitance, crosstalk noise and crosstalk-induced delay have overlap but are significantly different.

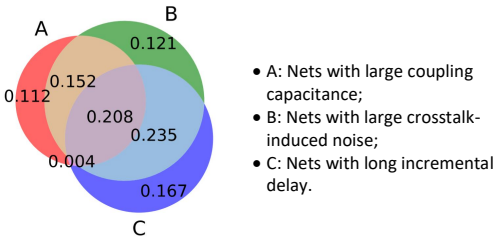

Fig. 6. Venn diagram of nets with large coupling capacitance, crosstalk-induced noise and crosstalk-induced delay [28].

In [28], machine learning models are built to take placement solutions as input and predict post-routing nets with large coupling capacitance, crosstalk noise and crosstalk-induced delay. The model features include logic information, such as gate

type, timing information obtained from post-placement timing reports, such as signal slew rate, and placement information, including net HPWL, source-sink distance and RUDY [36], which is a routing congestion indicator. Several machine learning engines are studied and the best results are produced by XGBoost. The data labels are obtained by performing a commercial signal integrity analysis tool on IWLS05 benchmarks [3] with ASAP7 cell library [12]. The results show that the XGBoost-based prediction achieves AUROC (Area Under Receiver Operating Characteristic curve) 0.99 for all of coupling capacitance, crosstalk noise and crosstalk-induced delay hotspot identification. Feature importance analysis based on XGBoost reveals that the most important features are products between HPWL and RUDY of large nets.

V. RTL Timing and Power Prediction

RTL (Register Transfer Level) design is a common entry for many digital IC design flows and is usually described by HDL (Hardware Description Language) such as Verilog and VHDL. Decisions in RTL include design choices, e.g., FIFO depth, and description styles, e.g., blocking assignments vs. non-blocking assignments. These decisions may cause a large impact on timing and power signoff. Figure 7 shows that different Verilog descriptions for the same design lead to different timing slack and power after placement. One issue is that the impact is usually not clear until late design stages, e.g., cell placement. Moreover, the impact depends on EDA tool parameter settings. In Figure 7, one dot corresponds to one logic synthesis parameter setting and the best setting is not known *a priori*. In order to assess such impacts, one needs to not only perform a flow through late design stages but also run it many times with various parameter settings. Since a design flow through placement for a sizeable circuit can easily take hours of CPU time, the overall assessment with hundreds of parameter settings may take 1-2 weeks. In [38], it is reported that the logic synthesis of a 32×32 systolic array with floating point operations takes a day.

Recently, an industrial tool [34] reduces layout-aware RTL PPA assessment time to several hours by parallel runs of quick design flows. Machine learning-based PPA prediction techniques are investigated in [13]. These techniques take architectural level parameters, such as L1 cache configuration, and logic synthesis parameters as input, and predict post-synthesis PPA results, where the performance is indicated by critical path delays. Multiple ML engines have been investigated and the best results are obtained from neural networks and XGBoost. The dataset are variants of Rocket chip designs [4] using commercial 25nm technology with consideration of different process corners. To effectively train models with a limited number of training data samples, the samples are generated according to Latin Hypercube sampling. Models trained with 150 data samples achieve accuracy of 98% for designs seen in the training dataset. Incremental training with 15 additional data samples provides accuracy of 95% on unseen designs.

Another RTL PPA prediction work is [38], which takes HDL code as input and predicts post-synthesis PPA. The HDL code is first parsed into graph intermediate representation (IR) using Yosys [37]. Model features are collected from paths in the graph. Since the number of paths of a graph is exponential with respect to the number of nodes, the work of [38] samples a subset of paths. The features of each path is fed to a lightweight transformer. The results from all transformers corresponding to the sampled paths are assembled to an MLP (Multi-Layer Perceptron), which produces the overall PPA predictions. This work employs a diversified set of data, including RISC-V cores, CNN cores, FFT, AES, etc. In addition, GAN is used to enrich the dataset. The synthesis in data generation is performed using a commercial tool with the 15nm FreePDK library [31]. The proposed models are hundreds times faster than performing logic synthesis with root relative square errors of around 0.5. For a BOOM core design [8], they reduce design space exploration time from 45 days to 2.1 hours. The path-based approach has an advantage that it can identify timing critical path besides the overall PPA assessment.

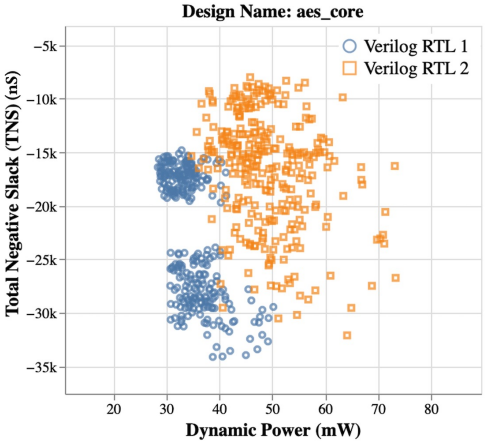

Fig. 7. Impact on post-placement timing and power from different Verilog descriptions for an AES core design with different synthesis parameter settings [35].

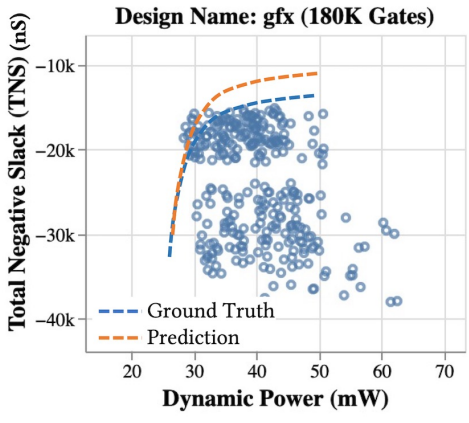

Fig. 8. Prediction of timing-power tradeoff curve [35].

Method	Model Features	Model Predictions	ML Engines	Handling of Tool Parameters
[13]	Architecture options, tool parameters	Post-synthesis PPA	XGBoost, NN	Different settings are considered individually
[38]	RTL paths	Post-synthesis PPA	Transformer	A fixed tool parameter setting
[35]	AST-based graph	Post-placement timing-power tradeoff	XGBoost	Concurrent consideration of multiple settings
[29]	AST-based graph	Post-synthesis delay, slew & AT	GNN	A fixed tool parameter setting

TABLE I
COMPARISON OF DIFFERENT RTL TIMING/POWER/AREA PREDICTION TECHNIQUES.

Machine learning-based prediction of RTL timing, power and their tradeoffs is studied in [35]. This framework also starts with HDL descriptions. It first parses an HDL code into an AST (Abstract Syntanx Tree) using an off-the-shelf tool, and then converts the AST into a graph representation similar to data and control flow graphs, where machine learning model features are extracted. Unlike other techniques, which predict post-synthesis results, it predicts post-placement timing and power, which are generally more accurate than synthesis results as layout information is additionally considered. Another difference is that its timing metric is Total Negative Slacks (TNS), which is more commonly used for signoff than path delays. This work includes two prediction problem formulations. In the first formulation, synthesis and placement parameters are taken as features besides the graph features and predicts the TNS and power for a specific tool parameter setting corresponding to a single dot in Figure 7. The second formulation predicts the overall TNS-power tradeoff for different parameter settings without including tool parameters in the model features. The tradeoff is described by a regression function empirically chosen to be

$$y = \alpha \ln(x - \beta) + \gamma \tag{1}$$

where x denotes dynamic power and y represents TNS. The values of coefficients α, β and γ are the machine learning model outputs. Figure 8 shows an example of such prediction for a case with 180K gates. Please note the tradeoff curve is to indicate a general trend instead of an accurate estimation and such indications are usually good enough for RTL design assessments. Multiple ML engines are evaluated in this work and the best results are based on XGBoost. For IWLS 2005 benchmarks [3] with 45nm technology, the XGBoost-based technique achieves 0.95 correlation and 0.0098 root mean square error compared to post-placement timing and power analysis by a commercial tool. It is eight orders of magnitude faster than running a commercial synthesis and placement flow due to the fact that the overall circuit timing and power can be captured without trying different tool parameters. One application scenario of this technique is quick feedback to HDL coding styles. For example, in Figure 9 left, the two dashed curves show the different timing-power tradeoffs of two different coding styles. While the blue one emphasizes more on low power, the orange one allows higher performance. The prediction shown on right correctly reflects this trend and can provide the feedback to designers almost instantaneously.

A recent work of RTL timing prediction is [29]. It starts from HDL code, extracts AST, which is converted to graph-based intermediate representations. It predicts post-synthesis delay and slew of major components, such as adders, as well as the arrival time at flip-flops. The consideration of signal slew rate is a main difference from other RTL timing prediction techniques. Another difference is its focus on a graph neural network (GNN) approach, although GNN-based techniques have been studied in [38], [35] as well. Different from the other techniques, where data are obtained using commercial tools, the data of this work are all generated by open-source tools, such as Yosys [37] and OpenSTA [2]. Compared to post-synthesis timing analysis results, it obtains 0.82 correlation with three orders of magnitude speedup. A comparison among the four RTL prediction methods is provided in Table I.

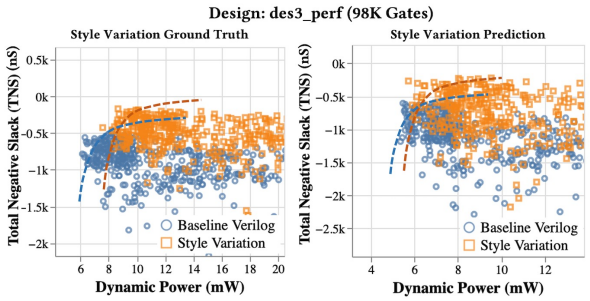

Fig. 9. Prediction of Verilog coding style difference [35].

VI. EDA TOOL RESOURCE USAGE PREDICTION

Recall from Section I that design closure and signoff will improve if they can use less resources and schedule. A surprisingly valuable lever is the modeling and prediction of EDA tool resource usage: cores, cache, main memory, bandwidth to filers, etc. Accurate modeling enables jobs to be launched on the right-sized hardware, conserving resources without sacrificing project schedule or risk. Such modeling is non-trivial: (i) there is a diversity of compute and storage requirements across EDA tools (e.g., contrast front-end functional simulation and formal verification, or back-end P&R, physical verification and timing signoff) [6], and (ii) the existence of other running jobs, along with data and scripting dependencies, can greatly affect runtimes. On the other hand, if armed with resource usage models, design organizations can optimize project schedules and provisioning of engineering IT and EDA licenses, with potential savings in the millions of dollars [1].

In the late 1990s, METRICS [15] attempted to address the above challenges with a unified approach to collection, storage and communication of design process data and tool param-

eters. The goal was to record all tool activities and design attributes, so as to enable data mining and prediction of tool outcomes and tool-specific "sweet spots" (i.e., fields of use). Follow-on work [24] developed a data mining-based predictor of placer runtime that achieved a correlation coefficient of 0.82. METRICS 2.0 [17] updated the METRICS scope (e.g., to handle the multiplicity of PVT corners, clocks, and threshold voltages seen in modern designs) and system architecture (e.g., to leverage a MongoDB database [40]) while retaining the focus on tool and design-specific metrics.

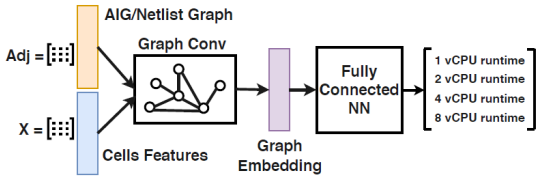

Fig. 10. The GNN-based runtime prediction model of [18]. Figure reproduced from [18].

More recently, [18] has proposed a graph convolution network (GCN) based model to predict EDA tool runtimes. The authors of [18] also outline an approach for deploying multiple EDA jobs on the cloud to minimize deployment costs while meeting deadline constraints. Figure 10 shows the architecture of the proposed model. Depending on the task, the model accepts either an RTL netlist for place-and-route runtime prediction, or an And-Inverter Graph (AIG) for synthesis runtime prediction. The model generates embeddings through two GCN layers and transforms these embeddings into predictions using a fully-connected neural layer [18]. Leveraging these predicted runtimes, speedup gains are derived based on the utilization of varying numbers of virtual CPUs (vCPUs). A multi-choice knapsack problem is solved to select an optimal number of vCPUs to run the EDA job, such that deployment cost is minimized and the deadline is met. The authors of [18] show that their model can achieve a prediction accuracy of 87% and a deployment cost reduction of 35.29%.

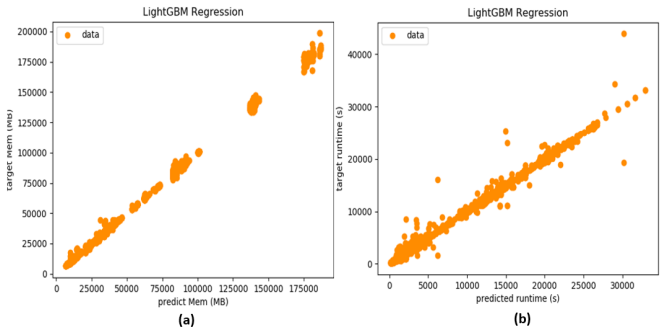

Fig. 11. Regression results for memory usage and runtime prediction of an industrial (signoff) static timing analysis tool.

Figure 11 presents sample results of a machine learning regression analysis using LightGBM, aimed at predicting memory usage and runtime for a signoff static timing analysis tool. The motivating context is final timing signoff, where hundreds of mode-corner combinations must be run using available servers and licenses, within as short a period as possible since the signoff directly gates the tapeout.[3] The example shown leverages data from an industrial collaborator. For predicting memory usage, the features taken into account include: (i) specifications of the machine on which the run was launched, (ii) operational corners belonging to a given mode of operation, and (iii) #hierarchical and #leaf cells in the design being timed. For runtime prediction, additional features are used, including (i) number of cores requested by the user (e.g., in an LSF queue) for running the tool, (ii) CPU clock frequency and architecture specifications, and (iii) total memory used by the runs.

VII. Conclusions

AI/ML techniques, in spite of their approximate nature, can provide substantial values to design signoff through design predictions for early alignment with design specifications. Such design predictions can avoid both excessive design iterations and overly pessimistic design margins. This paper showcases several AI/ML applications in this regard, including PVT corner selection, wire parasitic and delay prediction, routing-free crosstalk prediction, and RTL timing/power prediction. Besides design predictions, AI/ML helps efficient management of EDA tool resource usage, which is a key factor in design signoff but which has long been manually handled.

While significant progress has been achieved in leveraging AI/ML techniques for expediting design closure, there remains ample room for further improvement. Most AI/ML-based design prediction techniques have demonstrated considerable speedup compared to performing actual design flows. However, the integration of these techniques with design flows has not yet been well-studied. How should an optimization tool incorporate design predictions within an improved, iterative process? Will the incorporation lead to higher-quality correct-by-construction designs – or, alternatively, can restrictions of the design space (cf. the much simpler nature of signoff in FPGA or regular fabrics [32]) make it easier for AI/ML to improve signoff and design closure? How much will the incorporation of AI/ML-based design predictions within a design flow change the nature of the flow that is being predicted [21]? These are a few samples of important problems that deserve careful studies. Above all, the impact of AI/ML techniques on overall design closure instead of individual steps needs to be systematically evaluated.

VIII. Acknowledgments

We thank Sayak Kundu, Bodhisatta Pramanik, Zhiang Wang and Dooseok Yoon for their efforts in implementing and compiling material covered in this paper. Partial support from DARPA HR0011-18-2-0032, NSF CCF-2112665 and CCF-2106725, and SRC GRC-CADT 3103.001 is gratefully acknowledged.

[3]This becomes a problem of (i) finding a robust packing of jobs into servers based on runtime and memory predictions, while (ii) ordering jobs based on likelihood that they will expose violations that requires design iteration.

REFERENCES

[1] P. Agrawal, M. Broxterman, B. Chatterjee, P. Cuevas, K. H. Hayashi, A. B. Kahng, P. K. Myana and S. Nath, "Optimal Scheduling and Allocation for IC Design Management and Cost Reduction", *ACM TODAES* 22(4) (2017), pp. 60:1-60:30.

[2] T. Ajayi and D. Blaauw, "OpenROAD: Toward a Self-Driving, Open-Source Digital Layout Implementation Tool Chain", *Proc. of Government Microcircuit Applications and Critical Technology Conference*, 2019.

[3] C. Albrecht, "IWLS 2005 Benchmarks", *Proc. IWLS*, 2005.

[4] K. Asanovic, R. Avizienis, J. Bachrach, S. Beamer, D. Briancolin, C. Celio, H. Cook, D. Dabbelt, J. Hauser and A. Izraelevitz, "The Rocket Chip Generator", EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2016-17.

[5] W. Bao, P. Cao, H. Cai and A. Bu, "A Learning-Based Timing Prediction Framework for Wide Supply Voltage Design", *Proc. GLSVLSI*, 2020, pp. 309-314.

[6] E. Brown and A. Carter, "Predict the Cost of Electronic Design Automation on AWS Using Simulation", *AWS for Industries*, June 29, 2022. https://aws.amazon.com/blogs/industries/predict-the-cost-on-of-electronic-design-automation-on-aws-using-simulation/

[7] P. Cao, T. Yang, K. Wang, W. Bao and H. Yan, "Topology-Aided Multicorner Timing Predictor for Wide Voltage Design", *IEEE Design & Test* 40(1) (2023), pp. 62-69.

[8] C. Celio, D. A. Patterson and K. Asanovic, "The Berkeley Out-of-Order Machine (BOOM): an Industry-Competitive, Synthesizable, Parameterized RISC-V Processor", EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2015-16.

[9] J. Chen, I. H.-R. Jiang, J. Jung, A. B. Kahng, V. N. Kravets, Y.-L. Li, S.-T. Lin and M. Woo, "DATC RDF-2019: Towards a Complete Academic Reference Design Flow", *Proc. ICCAD*, 2019, pp. 1-6.

[10] V. A. Chhabria, W. Jiang, A. B. Kahng and S. S. Sapatnekar, "From Global Route to Detailed Route: ML for Fast and Accurate Wire Parasitics and Timing Prediction", *Proc. ACM/IEEE Workshop on Machine Learning for CAD*, 2022, pp. 7-14.

[11] V. A. Chhabria, W. Jiang, A. B. Kahng, S. S. Sapatnekar, "A Machine Learning Approach to Improving Timing Consistency between Global Route and Detailed Route", *ArXiv*, arxiv.org/pdf/2305.06917.pdf, 2023.

[12] L. T. Clark, V. Vashishtha, L. Shifren, A. Gujja, S. Sinha, B. Cline, C. Ramamurthy and G. Yeric, "ASAP7: a 7-nm finFET Predictive Process Design Kit", *Microelectronic Journal*, 2016, Vol. 53, pp. 105-115.

[13] W. R. Davis, P. Franzon, L. Francisco, B. Huggins III and R. Jain, "Fast and Accurate PPA Modeling with Transfer Learning", *Proc. ICCAD*, 2021, pp. 1-8.

[14] F.-Y. Fan, H.-M. Chen and I. Liu, "Technology Mapping with Crosstalk Noise Avoidance", *Proc. ASPDAC*, 2010, pp. 319-324.

[15] 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.

[16] K. Han, A. B. Kahng, J. Lee, J. Li and S. Nath, "A Global-Local Optimization Framework for Simultaneous Multi-Mode Multi-Corner Skew Variation Reduction", *Proc. ACM/IEEE DAC*, 2015, pp. 1-6.

[17] S. Hashemi, C. -T. Ho, A. B. Kahng, H. -Y. Liu and S. Reda, "METRICS 2.0: A Machine-Learning Based Optimization System for IC Design", *Workshop on Open-Source EDA Technology*, 2018.

[18] A. Hosny and S. Reda, "Characterizing and Optimizing EDA Flows for the Cloud", *IEEE Trans. on CAD* 41(9) (2022), pp. 3040-3051.

[19] A. B. Kahng, "New Game, New Goal Posts: A Recent History of Timing Closure", *Proc. ACM/IEEE DAC*, 2015, pp. 1-6.

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

[21] A. B. Kahng, "Machine Learning for CAD/EDA: The Road Ahead", *IEEE Design & Test*, Special Issue on Machine Learning for CAD/EDA, January-February 2023, pp. 8-16.

[22] A. B. Kahng, S. Kang, H. Lee, S. Nath and J. Wadhwani, "Learning-Based Approximation of Interconnect Delay and Slew in Signoff Timing Tools", *Proc. ACM/IEEE SLIP*, 2013, pp. 1-8.

[23] A. B. Kahng, U. Mallappa, L. Saul and S. Tong, "Unobserved Corner" Prediction: Reducing Timing Analysis Effort for Faster Design Convergence in Advanced-Node Design", *Proc. DATE*, 2019, pp. 168-173.

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

[25] H. Kellerer, U. Pferschy and D. Pisinger, "The Multiple-Choice Knapsack Problem", *Knapsack Problems*, Berlin, Springer, 2004.

[26] T. N. Kipf and M. Welling, "Semi-Supervised Classification with Graph Convolutional Networks", *arXiv 1609.02907*, 2017.

[27] K. Li and J. Malik, "Learning to Optimize", *arXiv 1606.01885*, 2016. https://arxiv.org/abs/1606.01885

[28] R. Liang, Z. Xie, J. Jung, V. Chauhan, Y. Chen, J. Hu, H. Xiang and G.-J. Nam, "Routing-Free Crosstalk Prediction", *Proc. ICCAD*, 2020, pp. 1-6.

[29] D. S. Lopera, I. Subedi and W. Ecker, "Using Graph Neural Networks for Timing Estimations of RTL Intermediate Representations", *Proc. MLCAD*, 2023, pp. 1-6.

[30] J. Lou and W. Chen, "Crosstalk-Aware Placement", *IEEE Design & Test of Computers*, 2004, 21(1), pp. 24-32.

[31] M. Martins, J. M. Matos, R. P. Ribas, A. Reis, G. Schlinker, L. Rech and J. Michelsen, "Open Cell Library in 15nm FreePDK Technology", *Proc. ISPD*, 2015, pp. 171-178.

[32] L. Pileggi, H. Schmit, A.J. Strojwas et al., "Exploring Regular Fabrics to Optimize the Performance-Cost Trade-Off", *Proc. DAC*, 2003, pp. 782-787.

[33] R. Puri, "Engineering the Future of AI for the Software and the Hardware Design Industry", *CASS research seminar*, June 2022. https://www.youtube.com/watch?v=5ukIftE38Vg

[34] J. Schultz, "RTL Architect: Parallel RTL Exploration with Unparalleled Accuracy", *Technical Report*, 2021.

[35] P. Sengupta, A. Tyagi, Y. Chen and J. Hu, "How Good Is Your Verilog RTL Code? A Quick Answer from Machine Learning", *Proc. ICCAD*, 2022, pp. 1-9.

[36] P. Spindler and F. M. Johannes, "Fast and Accurate Routing Demand Estimation for Efficient Routability-Driven Placement", *Proc. DATE*, 2007, pp. 1-6.

[37] C. Wolf, "Yosys Open Synthesis Suite", 2016.

[38] C. Xu, C. Kjellqvist and L. W. Willis, "SNS's not a Synthesizer: a Deep-Learning-Based Synthesis Predictor", *Proc. ISCA*, 2022, pp. 847-859.

[39] Z. Zhao, S. Zhang, G. Liu, C. Feng, T. Yang et al., "Machine-Learning-Based Multi-Corner Timing Prediction for Faster Timing Closure", *Electronics* 11(10) (2022), pp. 1571.

[40] MongoDB. https://www.mongodb.com/.