Overview of Research-

The objective of this project is to explore the collaborative landscape of Machine Learning and Formal Verification, both of which are considered indispensable today in almost all application domains. Recent advances in Machine Learning (ML) and Deep Learning (DL), in particular, along with significant advances in hardware architectures, have opened up the possibility of applying learning-based solutions to solve many optimization and classification problems in classical Electronic Design Automation (EDA). Some example application areas include high-level synthesis, electrical sign-off, timing closure, place and route, and design space exploration. In the VLSI design community, Formal Verification (FV) is now an established practice. Formal methods can provide rigorous correctness guarantees on hardware and software systems. Thanks to the availability of mature tools, their use is well established in the industry, and in particular to check safety-critical applications as they undergo a stringent certification process. Though apparently, the topics of ML and FV lie in two different dimensions, there is a magnificent scope to integrate them for solving some intriguing research problems. We primarily emphasize rationalizing two concepts through this work; 1) ML for FV and 2) FV for ML. Based on this, we categorize the research focus of this proposal into four separate directions as stated below.

Automated Specification Mining: In the last decade, most leading chip design companies have been extensively investigating the possibility of integrating Assertion Based Verification (ABV) into their pre-silicon validation flows. ABV allows the system designer to express the key correctness requirements of a system described in terms of formal properties in temporal logic and verify them over a given system implementation. The success of ABV largely depends on the ability to succinctly represent the correctness expectations in an assertion specification language amenable to automatic analysis. A major impediment to the widespread adoption of ABV is the lack of quality assertion suites. As a result, the predominant validation technology even today is based on simulation. A simulation-based validation setup typically consists of a large number of test vectors that intend to simulate various functional behaviors. The objective of this work is to investigate the possibility of mining assertions automatically from execution traces obtained by pre-silicon system simulation. The mined assertions in turn allow the designer to understand better the system under test, verify its correctness, and manage possible evolution changes. Towards this focus, one popular existing assertion miner is GoldMine [15]. GoldMine uses random or user testbench-guided simulation to generate design traces and builds decision trees to uncover and represent potential invariants learned. The use of decision trees limits the scope of the assertions that can be generated. Moreover, GoldMine needs to run a formal verification check to ensure that the generated assertions are indeed true ones. Though it has a sufficient industrial scale usage, it has been observed that since a huge number of properties are being generated by GoldMine, it is becoming very difficult to distinguish between important and useful versus not very interesting assertions. For example, a property like a signal is a clock, may not be interesting since we might already have an attribute on the signal or a naming convention directly inferring this signal.

To alleviate this, we can have a few assertions (invariants) embedded into the design by the designer as part of coverage tracking, then we can prioritize generated assertions based on the uncovered area. Additionally, we can evaluate the quality of the new assertions to increase the coverage. Towards this, we envision that the use of guided semi-formal methods can be helpful. Generally, semi-formal methods [3] extensively use statistical analysis for rare/interesting design regions, in practice instead of performing whole state analysis. We can use partitioned state analysis for seeding the next (semi-)formal analysis. In general, if a rare/interesting state computation heuristic/partition method is found ineffective, it is very difficult to switch from one heuristic to another. However, we try to use invariants to drive more unexplored regions. ML-guided techniques (e.g. clustering and classification) can be used here to create good state partitions, understand good seeds for the next semi-formal step, choose one from a portfolio of heuristics and better characterize the switching conditions. In this work, one of our objectives is to explore this idea in greater depth. Our other focus in this direction is to apply ML methods to extract interesting lighthouse gates that can improve the rare-state guidance or generate abstract states [7] as lighthouses to improve semi-formal search. In this context, we wish to leverage techniques from statistical machine learning [16], importance sampling [18], rare event simulation [17], and concepts from learning theory [19] to explore the task of generating more relevant assertions, focussed, have a high probability to be true, and in turn, lead to improvements in the verification life cycle. We believe that while using machine learning for invariant generation has been somewhat explored in the EDA community, the full

benefits that this synergy can bring in are still unexplored. As part of this project, we wish to take up this task. Moreover, we wish to look at generating non-linear invariants for which we may need sophisticated learning primitives.

Enhancing FV with ML: In this task, our objective is to explore the uses of ML in different verification tasks, that are acknowledged to be hard for the model checker, and any aid or guidance may have good enrichment in performance. The primary motivation is to explore how to achieve faster closure to justify formal properties. As the design complexity increases, a significant majority of the properties remain unjustified or lack a proof or counterexample due to resource constraints. A careful examination has often revealed more intelligent navigation of the cone of influence of a property could have steered the tool towards the conclusion, however, in the absence of such useful hints, a conclusion could not be achieved. We focus on some key elements in the discussion below.

In recent times, there is an increased importance of theorem provers in the formal verification context, particularly because of their ability for automated deduction and reasoning. Automated first-order theorem provers can aid in the formalization and verification of mathematical theorems and play a crucial role in program analysis, theory reasoning, security, interpolation, and system verification. Here, we suggest ML-based guidance in the proof search of the theorem prover. The problem with theorem provers today is that they still require a large amount of human intervention to successfully guide the proof. Our proposal is to explore data mining techniques to mine proof tactics from successful and unsuccessful proofs with the aim that they can be more widely applied.

Further, we intend to use ML in identifying 1) the best algorithm sequence, 2) possible options for faster proof/disproof of properties, and 3) the best heuristic selection for proof search. In the formal verification of hardware, there are many algorithms for redundancy identification (generally referred to as transformation algorithms) along with actual solving algorithms. Examples of transformation algorithms include SAT-sweeping [20], Logic refactor & rewriting, re-timing, speculative redundancy removal [21], etc. On the other hand, solving algorithms include different SAT solvers or different CNFization techniques, Bounded Model Checking (BMC), Property Directed Reachability (PDR), localization-based PDR, invariants on internal bits [1], semiformal [3], simulation, etc. In practice similar to super_prove [2], there are techniques like Expert-System Guided transformation-based verification [4], a rule-based methodology improving over a manually written rule set to decide if a particular formal verification algorithm can be effective during a verification step. However, using all these techniques in parallel and deciding on an effective engine can be very resource intensive and impractical. In this direction, we aim to investigate the following.

1. When a particular algorithm is being run, we try to find if it can generate some hints/features on how useful this algorithm has been and an indication of the next effective algorithm based on the extracted features. Also, we try to decide the important features that could be extracted from a particular FV algorithm.
2. When we have a huge number of verification properties embedded in a design, if an algorithm is effective on one property in the group [8][9], it could also be effective in the group. Hence, we will explore how semantic grouping of properties with respect to possible solver algorithms, or semantic grouping/clustering of the properties can be done based on an early quick simulation (like bit-parallel simulation) or other design feature extraction techniques.
3. The motivation for looking at heuristic selection is that for the type of theorem prover selected, the main barrier to effective use by non-experts is the need to select a good heuristic. Towards this, we aim to apply ML techniques for heuristic classification. Each heuristic can be separated by a subset of features for which it is well matched, hence, we can investigate the possibility of using separate feature sets for each heuristic classifier. To investigate this properly requires coverage of a very large combined search space. Individual heuristic classifiers cannot be treated in isolation because the best choice of features depends on the performance measure used. The feature subset investigation, based on overall heuristic selection performance, gave different results to the individual heuristic classifiers in terms of which features are of most significance.

In this task, our motivation is to explore modern feature extraction, engineering, representation, and inference techniques from state-of-the-art ML literature to generate insights to guide the different engines inside a formal tool. We believe that as part of this project we will be able to identify and enrich some of the algorithms for the

so-called “hard problems” for formal and make them more pervasive in the verification life cycle. While on one side our objective would be to attack the classical state explosion problem, on the other side, our outcomes may lead to faster convergence of formal tools.

Guided Debugging and Localization: In this direction, our primary objective is to use ML techniques for trace minimization and clustering, and prioritization of failures. In a large failure trace, localizing the bug to a specific area [5][6] length (counter-example length) and breadth (number of signals that can affect the bug) is quite a difficult task. Further, evaluating the areas in code that will be affected when the bug is fixed for further analysis of newly introduced logic is a difficult task. It is possible that a fix can introduce several other unexpected bugs. Usually, a simulation trace can be very deep, and very dense with a lot of uninteresting signal details. Our objective is to explore ML techniques to minimize the uninteresting information and group the traces for effective clustering. It would be interesting to use ML methods to reduce the similarity based on the uninteresting features present in the trace which do not affect the final bug evaluation. Similar to techniques for property partitioning, we also target to extract some features which can be reused across the failure clustering and property partitioning. A futuristic direction using ML to minimize the effort required for Post-Si failures is another goal.

Bug-fixing in deeply embedded portions of the logic is typically accompanied by the post-facto addition of new assertions, which cover the bug scenario. Formally verifying the assertions defined over such deeply embedded portions of the logic is challenging because formal methods do not scale to the size of the entire logic. Verifying the assertion on the embedded logic in isolation typically throws up a large number of counterexamples, many of which are spurious because the scenarios they depict are not possible in the entire logic. Using ML techniques, we wish to come up with some method for ranking the counterexamples so that only the most likely counterexamples are presented to the designer. For this ranking step, we can utilize the invariants mined from simulation traces of the design. These invariants typically come with a confidence value from the invariant miner, we can rank counterexamples based on their relationships with these invariants. Further, bug traces produced in simulation serve as the basis for patching the design in order to fix a bug. It is important to prove that the patch covers all instances of the bug scenario; otherwise, the bug may return with a different valuation of the variables involved in the bug scenario. For large circuits, formal methods do not scale well enough to comprehensively eliminate the bug, and achieving adequate coverage in simulation and regression testing becomes expensive. We wish to explore the synergy of formal methods and ML for analyzing the control trace leading to the observed manifestation of the bug and verifying the robustness of the bug fix with respect to that control trace. We can use classification techniques for classifying the bug fix based on the guarantee that our analysis can provide about the quality of the bug fix. Further, we can also recommend and prioritize the types of tests to validate the bug fix on other types of scenarios.

Formally Verified ML in EDA: The primary concern of the direction of work is to investigate correctness proofs of learning-based methods being used at different stages of EDA today, replacing their age-old counterparts. Nowadays, ML for EDA is becoming one of the hot topics. A lot of research efforts have been proposed that use ML techniques to improve the EDA method, almost in every stage of the design flow, starting from design space reduction to logic synthesis to placement/routing to testing verification [11, 12]. While ML holds the promise of delivering valuable insights and knowledge into the EDA flow, broad adoption of ML leads to an arsenal of risks starting from privacy concerns, and black box decision making, to self-improvement, or unexplainable intelligence. For instance, trust in technology is generally based on the understanding of how it works and the assessment of its safety and reliability. To trust a decision made by an ML technique, circuit designers or EDA tool users need to know that it is reliable and will make no mistake, however, given the scale and complexity of today’s system designs and applications, guaranteeing the satisfaction of safety/reliability objectives of these ML-assisted techniques under all possible input scenarios is an invincible challenge, due to factors such as nonlinearity and non-convexity of the model, high dimensional input spaces, real-valued weights, etc. In this work, we intend to systematically address some of these challenges.

ML has significant usage in the Physical Design stage of EDA [13,14]. To enable fast and accurate routability prediction, deep learning is used in [13] where the number and position of design rule violations of routing are

predicted with a given placed design before actually performing the routing. Generally, physical design has many similarities to image classification, hence, the authors exploit the well-studied image classification problem to the 2D placement, which also can be represented as an image. This allows them to use similar models such as CNN. The resulting predictions are used during placement to proactively avoid placements that are difficult to route. In this case, if the underlying ML model misclassifies, then it leads to an erroneous placement and routing. The misclassification that can happen may be because of some specific inputs. Then the pertinent question comes "Can we find such adversarial inputs for a well-trained ML model applied in the physical design stage of EDA?". One of our goals is to find a solution to this research question.

Another great application of ML in EDA is found in estimating the platform and environment properties. Generally, an ML model is trained to estimate the physical properties of the computing platform and the environment in which it operates. The models are used to predict future system changes or to predict the impact of management actions before executing them. The results of this estimation are presumably fed to other subsystems, either automatic or manual, for use in system management. Uncertainty in the inputs of these ML models can generate wrong estimated results about the platform or environment properties. To overcome this, it is of immense importance to bound the uncertainty of the underlying ML models so that we always can get a safe range of estimated results at the output. Though there has been significant research on the output range analysis of ML techniques [22] in the context of autonomous systems, there has been a little effort for the uncertainty-bound analysis of ML models from an EDA point of view.

[1] IC3 with Internal Signals, R Dureja, A Gurfinkel, A Ivrii, Y Vizel in FMCAD 2021
[2] Super prove - https://github.com/berkeley-abc/super_prove
[3] The Art of Semiformal Bug hunting, P. Nalla, R. Gajavelly, J. Baumgartner, H. Mony, R. Kanzelman, A. Ivrii; ICCAD 2016
[4] Scalable Automated Verification via Expert-System Guided Transformation, H. Mony, J. Baumgartner, V. Paruthi, R. Kanzelman and A. Kuehlmann, Conference on Formal Methods in Computer Aided Design, November 2004.
[5] Fault Diagnosis and Logic Debugging Using Boolean Satisfiability, A.Smith, A.Veneris, M.F.Ali, A.Viglas. IEEE Transactions on Computer-Aided Design of Integrated Circuits, Vol. 24, No 10, Oct. 2005
[6] Using unsatisfiable cores to debug multiple design errors, A.Suelflow, G.Fey, R.Bloem, and R.Drechsler. Great Lakes Symp. VLSI, 2008
[7] An effective guidance strategy for abstraction guided simulation, F. M. de Paula, A J. Hu; DAC 2007
[8] Boosting verification scalability via structural grouping and semantic partitioning of properties, R. Dureja, J. Baumgartner, A. Ivrii, R. Kanzelman, and K. Y. Rozier, FMCAD 2019
[9] Accelerating parallel verification via complementary property partitioning and strategy exploration, R. Dureja, J. Baumgartner, R. Kanzelman, M. Williams, and K. Y. Rozier, FMCAD 2020
[10] FRIENDS - Finding Related Interesting Events via Neighbor Detection, Raviv Gal, Haim Kermany, A. Ivrii, Ziv Nevo, A. Ziv in DAC 2020

[11] M. Rapp et al., "MLCAD: A Survey of Research in Machine Learning for CAD Keynote Paper," in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, doi: 10.1109/TCAD.2021.3124762.

[12] Guyue Huang et al.,. "Machine Learning for Electronic Design Automation: A Survey". ACM Trans. Des. Autom. Electron. Syst. 26, 5, Article 40 , September 2021, 46 pages. https://doi.org/10.1145/3451179.

[13] Z. Xie, Y.-H. Huang, G.-Q. Fang, H. Ren, S.-Y. Fang, Y. Chen, and J. Hu, "RouteNet: Routability Prediction for Mixed-Size Designs Using Convolutional Neural Network," in International Conference on Computer-Aided Design (ICCAD). IEEE, 2018, pp. 1–8.

[14] A. F. Tabrizi, N. K. Darav, L. Rakai, I. Bustany, A. Kennings, and L. Behjat, "Eh? Predictor: A Deep Learning Framework to Identify Detailed Routing Short Violations from a Placed Netlist," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD), 2019.

[15] S. Vasudevan, D. Sheridan, S. Patel, D. Tcheng, B. Tuohy and D. Johnson, "GoldMine: Automatic assertion generation using data mining and static analysis," *2010 Design, Automation & Test in Europe Conference & Exhibition (DATE 2010)*, 2010, pp. 626-629, doi: 10.1109/DATE.2010.5457129.

[16] Sugiyama, Masashi. *Introduction to statistical machine learning*. Morgan Kaufmann, 2015.

[17] Juneja, Sandeep, and Perwez Shahabuddin. "Rare-event simulation techniques: an introduction and recent advances." *Handbooks in operations research and management science* 13 (2006): 291-350.

[18] Tokdar, Surya T., and Robert E. Kass. "Importance sampling: a review." *Wiley Interdisciplinary Reviews: Computational Statistics* 2.1 (2010): 54-60.

[19] Bolles, Robert C. "Learning theory." (1975).

[20] Zhu, Qi, et al. "SAT sweeping with local observability don't-cares." *Advanced Techniques in Logic Synthesis, Optimizations and Applications*. Springer, New York, NY, 2011. 129-148.

[21] Case, Michael, et al. "Optimal redundancy removal without fixedpoint computation." *2011 Formal Methods in Computer-Aided Design (FMCAD)*. IEEE, 2011.

[22] Wang, Zhilu, et al. "Bounding perception neural network uncertainty for safe control of autonomous systems." *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2021.