MLParest: Machine Learning based Parasitic Estimation for Custom Circuit Design

Brett Shook (brett.w.shook@intel.com), Prateek Bhansali (prateek.bhansali@intel.com),
Chandramouli Kashyap (chandramouli.v.kashyap@intel.com), Chirayu Amin (chirayu.s.amin@intel.com),
Siddhartha Joshi (siddhartha.joshi@intel.com)
Intel Corporation, Hillsboro, OR, USA

***Abstract*—A novel machine learning based parasitic estimation (MLParest) method for pre-layout custom circuit design is presented. It reduces the error between pre-layout and post-layout circuit simulation from 37% to 8% on average for different measurements across a variety of analog circuits. MLParest can thus greatly reduce the number of iterations between pre-layout and post-layout design phases. The key contributions of this work are a machine learning based approach to parasitic estimation and a push-button model training framework, scalable across different technology nodes. To the best of our knowledge, a machine learning based framework of parasitic estimation is an industry first.**

Keywords—parasitic estimation, resistance, capacitance, circuit simulation, machine learning, post-layout, analog design

I. Introduction

Traditional analog IP development cycles start with schematic design, which is based on the architecture and specifications expected of an analog block. After initial sizing, analog designers perform performance, reliability and variation analysis using a SPICE-like circuit simulator. If the circuit fails to meet a specification across any process, temperature or voltage corner, analog designers must resize the circuit in the schematic. This phase is known as pre-layout design. Once specifications are met in the schematic, a layout of circuit schematic is drawn. During the layout, a team of mask designers, translates the abstract schematic to a technology specific format which can be fabricated. This process is time consuming, often taking anywhere from days to weeks, and is expected to be even longer in the future as design rules of advanced technology nodes become increasingly complex. Once the layout is ready, circuit designers perform interconnect parasitic (resistance and capacitance) extraction. Next, using the extracted parasitics back-annotated into the schematic design netlist, post-layout SPICE simulations are performed. Only at this stage, if all the specifications of the analog block are met, the design cycle completes. If any specification is not met in the post-layout simulations, the schematic is resized or rearchitected and sent back for layout and extraction causing several iterations between schematic and mask design phases.

Usually, the performance of post-layout circuits is widely different from pre-layout circuits due to the addition of interconnect parasitics and complex device layout effects in post-layout circuits. To illustrate this, we performed simulations on a set of representative analog IP blocks on a 10 nm technology across design metrics such as delay, rise/fall time, duty cycle, frequency, bandwidth, DC gain, leakage current and power. Fig. 1 shows the differences in measurements between pre-layout and post-layout simulations for this set of 10nm analog circuits. This collection of circuits includes a duty-cycle corrector, PLL clock generator, CTLE, Op-Amp, sense-amplifier, TX-driver, level-shifter, ring oscillator, and a PLL delay line. We plot the average and maximum of absolute values of error (%) across a set of measurements for each circuit in order to avoid cancellation of positive and negative errors. From the plot, we can see that there is a huge difference in the performance of the circuits when interconnect parasitics are not included in the simulations. This leads to multiple iterations between the pre-layout and post-layout phases leading to resizing and layout fixes. To circumvent this problem, circuit designers often put excessive margins in specifications as guard bands in pre-layout design phase. This leads to sub-optimal designs in terms of power, performance, and area.

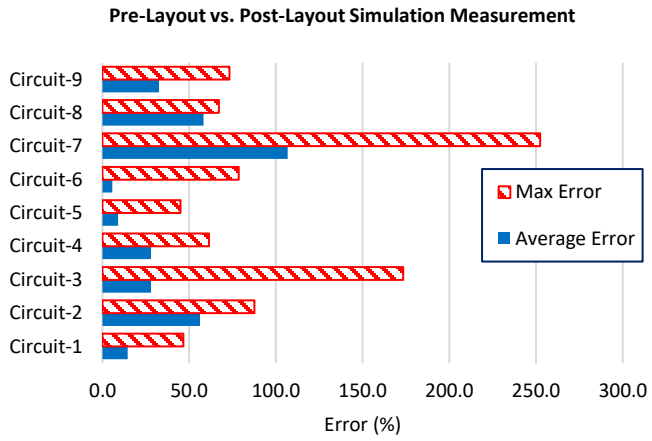

Fig. 1. Bar plot showing the error in pre-layout simulation measurements when compared to the corresponding post-layout simulation measurements.

To address this issue, we present our machine learning based parasitic estimation framework called MLParest in this paper. MLParest provides an accurate estimate of expected post-layout interconnect parasitics in the pre-layout design phase. Note that both the interconnect resistance and capacitance can be estimated using MLParest to aid circuit designers. This allows designers to incorporate the interconnect effects on performance of the circuit during the schematic design stage. This reduces the gap between post-layout and pre-layout simulation results which helps reduce design iterations between these two phases. MLParest can thus lead to a much faster turn-around time for analog designs.

The rest of the paper is organized as follows. Section II discusses background and related work. In Section III, we present our machine learning based parasitic estimation flow. Section IV summarizes the results. Section V concludes this paper.

II. Related Work

There are no known tools currently in the Electronic Design Automation (EDA) industry which predict interconnect parasitic RC for pre-layout circuits based on just the information contained in the schematics. Prior work [1], [2] has proposed the use of floor-planning, placement, and routing information in addition to schematic information to estimate interconnect parasitics. However, this information is not easily available to analog designers in the early stages of design, which limits its usage. There is a growing body of research on automatic generation of analog layouts [3], [4]. However, these tools are still not fully automated and mature. The current industry standard practice is to either move to the initial layout phase quickly or use designer experience to manually guesstimate the parasitic values for particular nets in a circuit. If the schematic designers don't have an accurate estimate of the interconnect parasitics, it can take multiple cycles of re-design at both the schematic and layout design stages which increases the total design time.

Although interconnect capacitance affects the performance of the circuit by directly changing its power and delay characteristics, estimating this capacitance alone is not enough. Interconnect resistance estimation is increasingly important as process technology scales downward [5] and is especially critical for analog circuits. Fig. 2. shows the large impact of interconnect resistance on simulation accuracy by plotting the difference between post-layout designs with and without interconnect resistance included. Hence, it is important to correctly estimate both R and C during the pre-layout design phase to avoid surprises in the post-layout phase, where it is costlier to rework the design to meet expected power and performance targets.

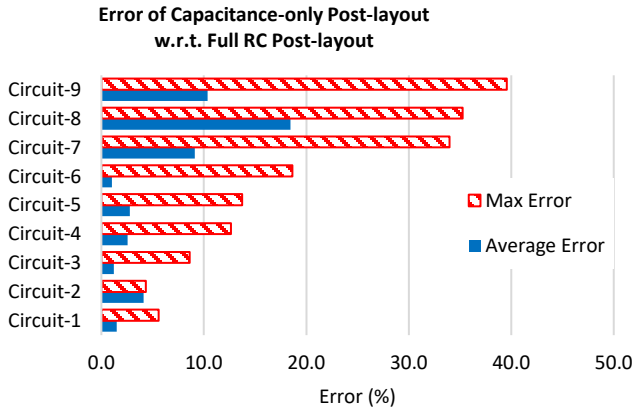

Fig. 2. Bar plot showing the aggregate measurement error between simulations where interconnect resistance is removed with respect to full post-layout interconnect resistance and capacitance.

III. Machine Learning Based Parasitic Estimation

To address the issues outlined in the previous section, a parasitic estimation engine, MLParest, has been developed. We leverage machine learning techniques with existing post-layout extraction data to train estimation models and predict interconnect parasitics. Machine learning has been proposed for pre-routing timing prediction in the digital domain[6]. In the analog mixed signal domain, it is worthwhile to emphasize that using a machine learning paradigm in the context of parasitic estimation is an industry first.

In order to introduce our approach to parasitic estimation, let us consider an example of a pre-layout schematic net and its post-layout counterpart shown in Fig. 3. Here a pre-layout net transforms into a multi-port RC circuit after interconnect parasitics are extracted. This transformation happens for every net of a circuit, which then leads to significant differences in pre-layout and post-layout circuit simulation. With MLParest we aim to estimate this RC-network for each pre-layout net and then use it as a proxy for the post-layout network in pre-layout simulations.

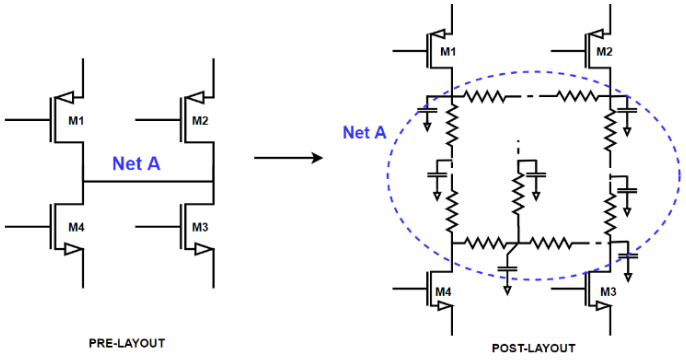

Fig. 3. Pre-layout and post-layout representation of a net.

A. Modeling

Extracted nets can be of an arbitrary structure, typically a tree, with several resistors and capacitors. Each net can be different from others in terms of its topology and routing. In order to make this problem tractable with machine learning, we approximate each post-layout net using two scalars – effective capacitance and effective resistance. Effective capacitance, C_{eff}, is the sum of total capacitance incident on a net – including grounding and cross-capacitance to other nets. To compute effective resistance, R_{eff}, we use linear time-invariant (LTI) circuit theory. Here we define the concept of an effective time constant, τ_{eff}, which accounts for the time constants of the original circuit system. For a system with N poles, $p_1, p_2, \ldots, p_N$, an effective time constant is defined as below:

$$\tau_{\text{eff}} = \sqrt{\frac{1}{{p_1}^2} + \cdots + \frac{1}{{p_i}^2} + \cdots + \frac{1}{{p_N}^2}} \quad (1)$$

This allows us to compute effective resistance as follows:

$$R_{eff} = \frac{\tau_{eff}}{C_{eff}} \quad (2)$$

Having computed effective resistance and capacitance, we can synthesize a post-layout net using a simple star topology shown in Fig. 5, where $R_{branch} = \frac{R_{eff}}{M}$ where M is the number of transistor connections to the net. The star topology has a time constant equal to the effective time constant of the post-layout net. We selected a star topology, instead of a dense multi-port admittance matrix, for two reasons. The first being that a dense multi-port matrix for each net is prohibitively expensive to compute. Secondly, based on our extensive testing on industry circuits we found that the star topology suffices for our parasitic estimation application.

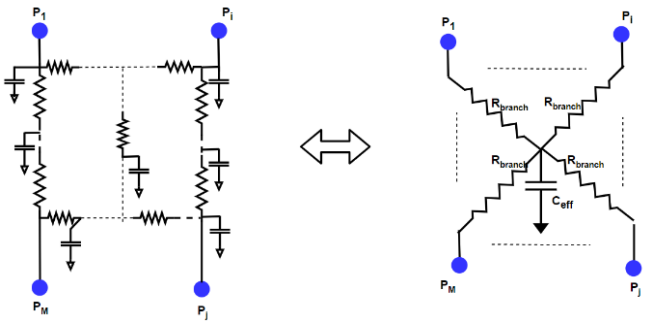

Fig. 4. Modeling of a multi-port net using the star topology.

B. Training

As shown in Fig. 5, the starting point of any Machine Learning (ML) based approach is collecting a large amount of training data called the dataset. In Fig. 5, $\vec{x}$ is the vector of input variables and y is the response i.e. $y = f(\vec{x})$. It is this unknown function f that an ML algorithm tries to learn given a dataset. The learning algorithm chooses from a set of candidate functions called the hypothesis set [7]. Based on appropriate error criteria, such as Mean Square Error (MSE) or Mean Absolute Error (MAE), the best function $g(\vec{x})$ from the hypothesis set is selected. $g(\vec{x})$ is an approximation to the actual function $f(\vec{x})$, which is unknown. The learned function $g(\vec{x})$ is then used in applications as a proxy for $f(\vec{x})$. The richness of the hypothesis set provides a trade-off between the accuracy of the learned model and overfitting. At one extreme, if the hypothesis set has only one candidate, it could turn out to be very inaccurate. On the other hand, if the hypothesis set has too many candidates, there could be overfitting since the learning algorithm could in principle pick a candidate that fits the training data exactly. This balancing act, called the bias-variance trade-off, is at the heart of machine learning algorithms including ours.

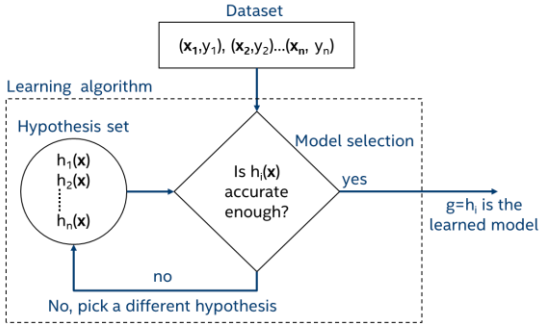

Fig. 5. Flowchart of the general machine learning paradigm.

We now specialize the general ML training algorithm to the problem of parasitic estimation as shown in Fig. 6. We start with a collection of circuits with pre-layout (schematic) and post-layout extracted (RC) data. Next, we extract features of each net for all circuits by traversing their netlists. The $\vec{x}$ variables in the dataset, on a per net basis, are the features from the pre-layout netlist which are: the number of connections on a net, the number of hierarchies the net traverses, the number of the drain, gate and source connections, the total width of all MOS devices, net type (internal or port), and the number of p-type and n-type devices. The y variables are the R_{eff} and C_{eff} values from the past layout circuits. We learn two functions: one for resistance and one for capacitance. Since we use supervised learning, we compute actual C_{eff} and R_{eff} using post-layout nets. As explained previously, C_{eff} is the sum of the total incident capacitance on a net. To compute R_{eff}, we first obtain a set of dominant poles using PRIMA [8] of the multi-port net. These dominant poles are then used to compute τ_{eff}, which then yields R_{eff} based on (2).

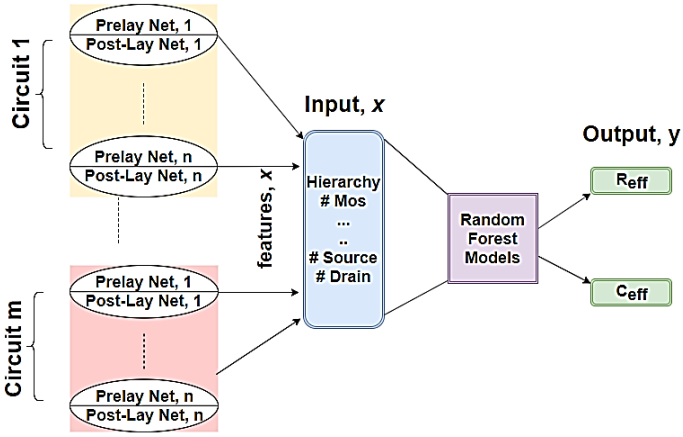

Fig. 6. Model training for parasitic estimation.

After evaluating several machine learning methods like neural-networks, linear regression, etc., we chose the Random Forest model [7], [9] for supervised learning since it offered the best prediction accuracy based on our experiments. From the total collection of nets derived from all the circuits in the dataset, we randomly select 70% for training and 30% for testing. We used an K-fold cross-validation scheme to tune the parameters of the decision trees such as depth, the number of trees in the forest, etc. We also use the RANSAC algorithm to remove outliers [10]. These outliers may be present in the data due to extraction with dirty layouts, inconsistency between schematic and layout, etc.

C. Inference

Once a model has been learned, we use it to predict interconnect parasitics of pre-layout schematic nets that haven't been seen before. As illustrated in Fig. 7, we first parse the pre-layout netlist and extract the features of each net. Next, these features are fed to the trained Random Forest models for R_{eff} and C_{eff} yielding predicted values. Using these scalars, we synthesize a star topology for each net. This is pictorially shown

in Fig. 8, for an M-port net. For simulation purposes, we generate an industry-standard format called Standard Parasitic Exchange Format (SPEF) [11] to write estimated parasitic information. SPEF file contains star topology of every net in the circuit. Finally, the pre-layout netlist and SPEF file are consumed by SPICE-like simulators to perform circuit simulation. In analog design, it is important to have parasitic matching at some differential nodes to reduce variation. By design, since those nets have exactly the same features, MLParest honors analog matching.

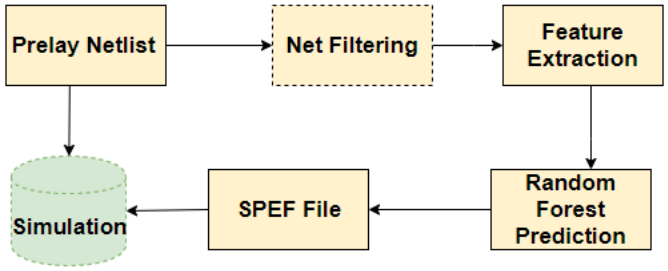

Fig. 7. MLParest inference flow diagram.

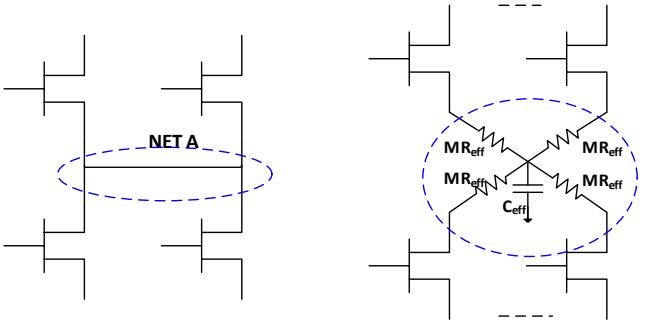

Fig. 8. Parasitic representation of a pre-layout net for simulation with MLParest.

IV. Results

A. *MLParest training*

We implemented the MLParest framework and associated model learning in Python with the scikit-learn package [12]. For MLParest, we used 212 and 647 analog blocks to train on a 14nm and 10nm process, respectively. The 10nm and 14nm datasets contain a wide variety of industrial analog circuits. Table I shows some statistics for both the 10nm and 14nm training and analysis. As part of data gathering, we first traverse design databases for archived data. Then, we learn R_{eff} and C_{eff} values from previous post-layout designs which is subsequently used for training the Random Forest models. It is important to note that, MLParest does not depend on test benches with stimuli. Thus, we can leverage early post-layout data available for certain key analog blocks and use it across the board for hundreds of other designs on the same process node without waiting for layout. MLParest has been used to gather data and train models for several different process technologies with very little modification to the overall flow.

TABLE I. Model Training Results

Criterion	10nm	14nm
Number of analog IP buildings blocks	686	212
Number of nets	276K	176K
Cap. R^2 score	0.95	0.94
Res. R^2 score	0.80	0.73

In Fig. 9-12, we compare actual and predicted values of effective resistance and capacitance for 10nm and 14nm nets. As can be seen, Random Forest models predict R and C well. It is observed that the variation in resistance predictions is greater than the variation seen in capacitance. This is potentially due to the fact that our feature list is missing placement information which affects wire-length and hence the resistance. As is the case with machine learning, we do find outliers in our prediction. Based on our investigation they usually occur on enable, clock, and global signals with several thousands of devices connected to them. In practice, once such signals are turned on, they do not contribute much to measurements which affect circuit performance due to their constant nature. However, we are working towards improving and capturing such outliers in our modeling.

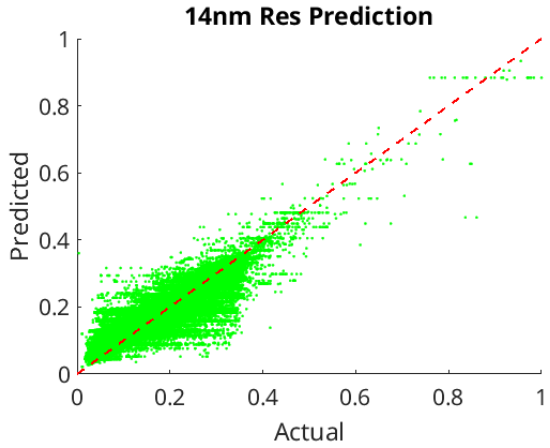

Fig. 9. Scatter plot comparing normalized actual and predicted resistance values for 14nm

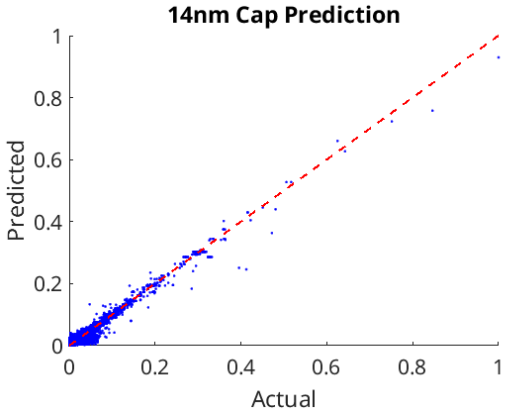

Fig. 10. Scatter plot comparing normalized actual and predicted capacitance values for 14nm

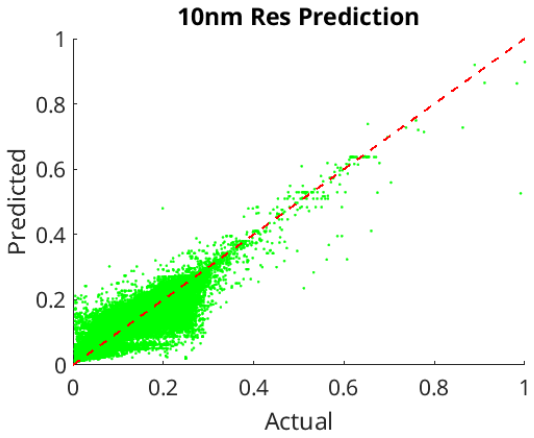

Fig. 11. Scatter plot comparing normalized actual and predicted resistance values for 10nm

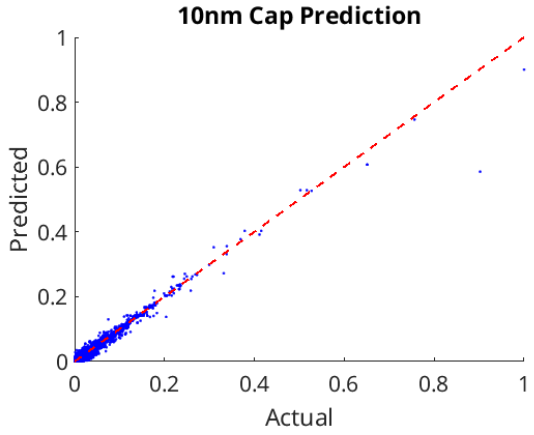

Fig. 12. Scatter plot comparing normalized actual and predicted capacitance values for 10nm

B. *Accuracy*

In this section, we present our results from using MLParest on the nine 10nm analog circuits introduced earlier. These nine circuits are not part of the training set. Fig. 13 shows the average of absolute error (%) (MAE) across simulation measurements on the 10nm test cases for pre-layout and MLParest against the post-layout results. These metrics include delay, rise/fall time, bandwidth, gain, average current over time, leakage current, and oscillator frequency. Fig. 14 shows the maximum absolute error (%) comparison between pre-layout and MLParest against post-layout simulation results. Fig. 13 and Fig. 14 clearly show that MLParest significantly improves simulation accuracy over pre-layout simulations.

Fig. 15 shows the AC output response waveforms from the Circuit 5 test case. It is clear that the MLParest predicted RC parasitics track post-layout results for metrics such as DC gain, peaking frequency, and bandwidth much better than the pre-layout schematic only results. In Fig. 16, we compare transient response of a signal. As is evident from the figure, MLParest closes the gap between pre-layout and post-layout circuit.

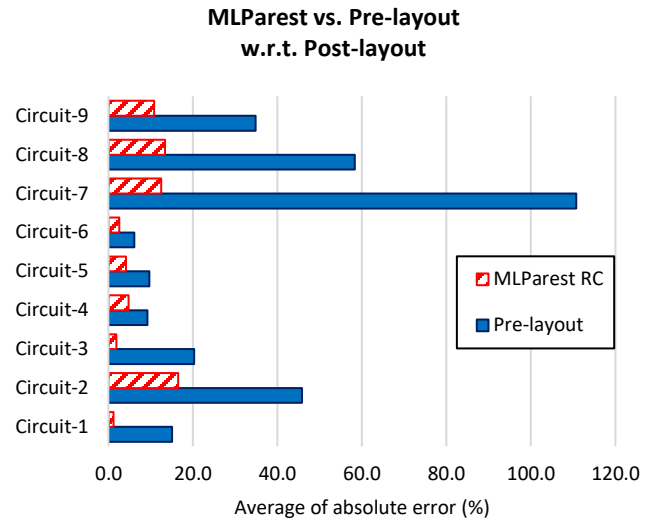

Fig. 13. Average MLParest error with respect to post-layout simulation.

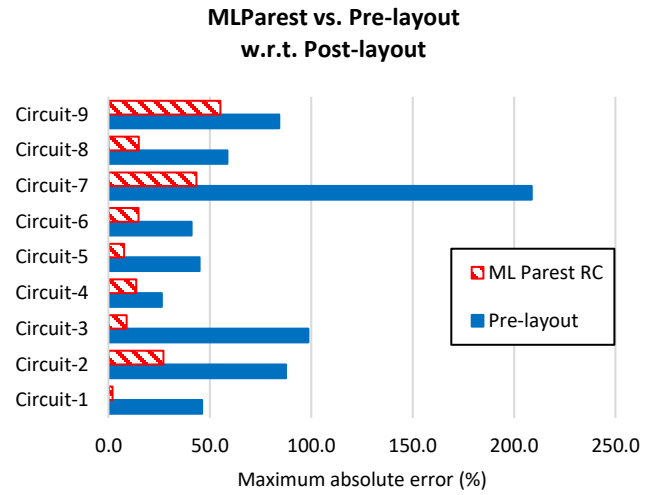

Fig. 14. Maximum MLParest error with respect to post-layout simulation.

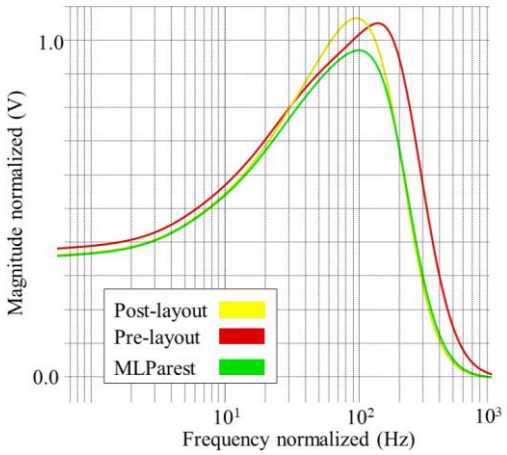

Fig. 15. AC response of an output signal for Circuit 5.

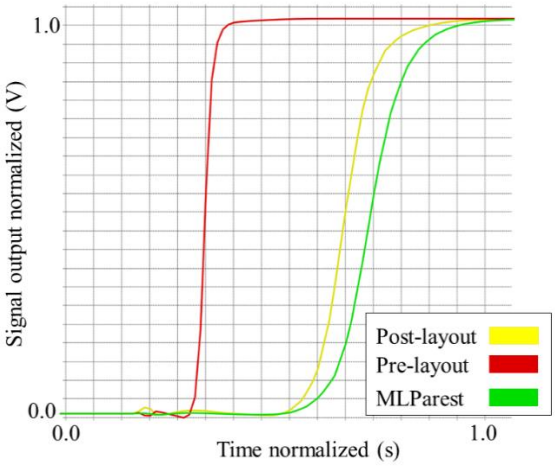

Fig. 16. Transient response of a Circuit 9

Table 2 shows data from two specific 14nm test cases. RC estimation using MLParest significantly reduces the accuracy gap with respect to post-layout results.

TABLE II. PAREST ACCURACY ON 14NM TEST CASES.

Test case, Metric	Post-layout	Pre-layout	MLParest RC
Ring Oscillator, Frequency (normalized)	1.000	2.341	1.222
Charge Pump Output Voltage (normalized)	1.000	1.054	0.988

C. Runtime

For the above test cases, MLParest estimation time is very small (typically seconds) as compared to overall simulation time on the order of minutes or hours. We observed a runtime increase of about 20% as compared to pre-layout simulations which is still significantly lower than full post-layout simulations. Also, we would also like to make a note on the theoretical aspects of circuit simulation runtime in conjunction with MLParest. Circuit simulation runtime depends on the number of nodes, the number of dense rows in the circuit matrix, and the Number of Non-Zeros (NNZ) in the circuit matrix. In practice, the number of nodes and NNZs in pre-layout and corresponding post-layout circuits differs by two orders of magnitude. Due to the star topology of the MLParest RC model, the additional number of new nodes and NNZs inserted in the pre-layout circuit matrix is strictly bounded by four times the number of MOSFETs as one resistor maximum is attached to each pin (drain, gate, source, bulk). Moreover, MLParest does not increase the number of dense rows in the circuit matrix. Hence, we do not expect a significant increase in runtime with MLParest usage as compared to the pre-layout simulation without it.

V. CONCLUSIONS

We presented a machine learning based parasitic estimation tool called MLParest which can predict both interconnect resistance and capacitance for pre-layout schematics. MLParest works seamlessly across multiple industry-standard analog simulators. MLParest is not limited to analog design and can be used in mixed-signal validation flows as well. MLParest does not depend on test benches with stimuli and is easily scalable across different technology nodes. Finally, MLParest supports industry-standard input and output formats allowing it to be incorporated alongside a variety of EDA tools.

The experimental results show a significantly improved accuracy with respect to pre-layout simulations. Our future work includes early prediction for electromigration and IR drop to avoid potential issues before the layout design phase is started. Another possible extension of our work includes an enhanced placement based parasitic estimation for achieving better accuracy on designs that are sensitive to device placement.

VI. ACKNOWLEDGEMENT

We would like to thank Jian Gong, Mikalai Kisialiou, Renuka Lokare, Sunder Kankipati, Reshma Kamat, and Kunal Kishore for sharing background knowledge on parasitic estimation and circuit simulator integration.

REFERENCES

[1] H. Y. Foo, K. W. C. Leong, and R. Mohd-Mokhtar, "Density aware interconnect parasitic estimation for mixed signal design," in *2012 IEEE International Conference on Circuits and Systems (ICCAS)*, 2012, pp. 258–262.

[2] S. Shah and A. Nunez, "Pre-layout parasitic estimation in interconnects," in *48th Midwest Symposium on Circuits and Systems, 2005.*, 2005, pp. 1442–1445.

[3] R. Martins, N. Lourenço, A. Canelas, R. Póvoa, and N. Horta, "AIDA: Robust layout-aware synthesis of analog ICs including sizing and layout generation," in *2015 International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD)*, 2015, pp. 1–4.

[4] E. Chang *et al.*, "BAG2: A process-portable framework for generator-based AMS circuit design," in *2018 IEEE Custom Integrated Circuits Conference (CICC)*, 2018, pp. 1–8.

[5] "The International Roadmap for Devices and Systems," *IEEE*, 2017 [Online]. Available: https://irds.ieee.org/images/files/pdf/2017/2017IRDS_MM.pdf

[6] E. C. Barboza, N. Shukla, Y. Chen, and J. Hu, "Machine learning-based pre-routing timing prediction with reduced pessimism," in *2019 56th ACM/IEEE Design Automation Conference (DAC)*, 2019, pp. 1–6.

[7] Y. S. Abu-Mostafa, M. Magdon-Ismail, and H.-T. Lin, *Learning from Data*, vol. 4. AMLBook New York, NY, USA:, 2012.

[8] A. Odabasioglu, M. Celik, L. T. Pileggi, L. T. Pileggi, and L. T. Pileggi, "PRIMA: Passive reduced-order interconnect macromodeling algorithm," in *Proceedings of the 1997 IEEE/ACM international conference on Computer-aided design*, 1997, pp. 58–65.

[9] T. K. Ho, "Random decision forests," in *Proceedings of 3rd international conference on document analysis and recognition*, 1995, vol. 1, pp. 278–282.

[10] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[11] "1481-2009 - IEEE Standard for Integrated Circuit (IC) Open Library Architecture (OLA)." [Online]. Available: https://ieeexplore.ieee.org/document/5430852

[12] Pedregosa, Fabian, et al. "Scikit-learn: Machine learning in Python." Journal of machine learning research 12.Oct (2011): 2825-2830.