MMM: Machine Learning-Based Macro-Modeling for Linear Analog ICs and ADC/DACs

Yishuang Lin*, Yaguang Li†, Meghna Madhusudan‡, Sachin S. Sapatnekar‡, Ramesh Harjani‡, Jiang Hu†*
†Dept. of Electrical and Computer Engineering, Texas A&M University
*Dept. of Computer Science and Engineering, Texas A&M University
‡Dept. of Electrical and Computer Engineering, University of Minnesota

***Abstract*—Performance modeling is a key bottleneck for analog design automation. Although machine learning-based models have advanced the state-of-the-art, they have so far suffered from huge data preparation cost, very limited reusability, and inadequate accuracy for large circuits. We introduce ML-based macro-modeling techniques to mitigate these problems for linear analog ICs and ADC/DACs. On representative testcases, our method achieves more than 1700× speedup for data preparation and remarkably smaller model errors compared to recent ML approaches. It also attains 3600× acceleration over SPICE simulation with very small errors and reduces data preparation time for an ADC design from 40 days to 9.6 hours.**

I. Introduction

The lack of performance models that are simultaneously fast and accurate is a primary reason why AMS (analog/mixed signal) design automation has not achieved as much success as its digital counterpart. SPICE simulations are computationally so expensive that their use for performance estimation in circuit optimization has to be very limited. In order to overcome this bottleneck, fast circuit modeling techniques have been extensively studied. Typical early approaches are symbolic analysis [1] and MOR (Model Order Reduction) [2], where equations of circuit transfer functions or performance are derived mathematically. In [3], symbolic analysis is integrated with graph representations. However, symbolic analysis faces exponential growth of symbolic elements with respect to circuit sizes and hence is mostly restricted to small circuits. The effectiveness of MOR is mostly restricted to linear circuits and it is very difficult to apply MOR on ADC/DACs.

An alternative to simulation-based approaches uses data-fitted/trained models, e.g., the early work of posynomial models [4] and SVM (Support Vector Machine) models [5]. Recently, methods in this category have made significant progress largely due to the advancement of neural network technology. ANNs (Artificial Neural Networks) have been applied for AMS circuit modeling in [6], [7], [8]; analog placement solutions are assessed by CNNs (Convolutional Neural Networks) in [9]; and GCN (Graph Convolutional Network) methods have been employed for estimating performance in reinforcement learning-based analog transistor sizing [10]. In [11], [12], a customized GNN (Graph Neural Network) technique is developed to evaluate analog circuit performance. In [13], neural network-based transistor models are constructed and integrated with symbolic analysis. Overall, machine learning-based models become a widespread trend due to their promising results on fast performance estimation time and generally good accuracy.

However, machine learning-based models face their own challenges. The preparation of ML training data is computationally expensive as it relies on circuit simulations. Even assuming 20 minutes for simulating a large circuit (actual simulation times could be larger), obtaining 1000 data samples requires half a month of simulation time. The expensive data preparation is exacerbated by the diverse performance metrics for different analog circuits (e.g., performance metrics of OTA (Operational Transconductance Amplifier) include gain, bandwidth and phase margin, while LDO (Low Dropout Regulator) is evaluated by dropout voltage and power supply rejection ratio), unlike digital circuits, where the metrics are uniform (power/performance/area). This implies poor model reusability across analog circuits. A machine learning model trained from OTA is difficult to work for LDO. In addition, it is noticed in [7] that the accuracy of ANN models drops remarkably when circuit sizes or performance range increases.

A few prior efforts attempt to address data preparation cost. Model transferability from schematic to layout for the same circuit is shown in [6], but the model for one type of circuit does not apply for a different circuit type. In [10], knowledge transfer is restricted among different process technology nodes of the same circuit design. Transfer learning for CNN models is explored in [9]. However, CNN models need much more training data than GNN models [11]; moreover, the knowledge transfer in [11] is restricted between two different topologies of the same type of circuits. CCI-NN (Circuit Connectivity Inspired Neural Network) [7] pre-embeds some circuit knowledge into ANN models to reduce training samples.

We propose a new approach of sub-circuit level ML-based Macro-Modeling (**MMM**) with the *objectives of largely reducing ML performance model construction cost via model reuse and improving model accuracy* at the same time. Although in [13] transistor level ML models can also be reused, they are too fine-grained and the consequently frequent calls to such models substantially slow down the estimation of circuit level performance. A CCI-NN [7] is composed by a set of sub-NNs, which appear to be similar to our macro-models, but there is a critical difference: CCI-NN must train the entire NN for a whole circuit and cannot train sub-NNs individually. By contrast, each macro-model in MMM is independently trained. There are two significant consequences due to this difference. First, the output of a sub-NN in [7] is generally not associated with any physical meaning and therefore a sub-NN is very difficult, if not impossible, to be reused in a different type of circuit. Second, CCI-NN is restricted to use only neural network models while our MMM supports almost any ML models including random forest and XGBoost. As a result, the data preparation cost of CCI-NN is over 300× more than that of MMM and MMM achieves smaller errors than CCI-NN with 2.9× shorter circuit performance estimation time.

The contributions of this work include the following.

- We propose techniques for building sub-circuit level macro-models that can be reused in performance estimation of linear analog ICs and ADC/DACs, two types of common AMS circuits. Variable loading effects between sub-circuits are also considered in the MMMs.
- The effectiveness of MMM are validated on multiple linear analog ICs and ADC/DACs, including circuits with feedbacks and a circuit with over 20K devices.
- Comparisons are made with recent ML approaches of ANN [6], CCI-NN [7], GNN [11] and NN-based symbolic analysis [13] as well as MOR [14] to show the following advantages of MMM.
  - **Training data preparation time.** MMM achieves data preparation speedup of 1788×, 1791×, 357× and 885×

979-8-3503-0955-3/23/$31.00 ©2023 IEEE

vs. ANN, GNN, CCI-NN and NN-based symbolic analysis. For an 8-bit flash ADC design, MMM reduces the data preparation time from 40 days to 9.6 hours.
- **Accuracy of circuit performance estimation.** MMM achieves less than 1% modeling error compared to SPICE, significantly lower than 6% of ANN, 6% of CCI-NN, 4% of MOR, 4% of GNN and 9% of NN-based symbolic analysis.
- **Runtime cost of circuit performance estimation.** MMM is 3639×, 1873×, 42× and 2.9× faster than SPICE, NN-based symbolic analysis, MOR and CCI-NN, respectively.
- **ML model training cost.** MMM obtains training time reduction of 2462×, 4520×, 5469× and 18255× compared to ANN, CCI-NN, GNN and NN-based symbolic analysis.

To the best of our knowledge, this is the first study on ML-based macro-models for linear analog ICs and ADC/DACs.

II. Goals, Strategy, and Scope of MMM

A macro-model is a model for a sub-circuit that appears in one or multiple different circuits as a component. In Figure 1, the two-stage OTA (Operational Transconductance Amplifier) is composed of a one-stage OTA sub-circuit and two sub-circuits of CSAs (Common Source Amplifiers). Similarly, the two-stage VGA (Variable Gain Amplifier) consists of a one-stage VGA and two CSAs. The performance of a circuit, e.g., either the two-stage OTA or the two-stage VGA, can be obtained by assembling the macro-models of corresponding sub-circuits. Since the CSA appears multiple times in the two circuits, its macro-model is reused multiple times.

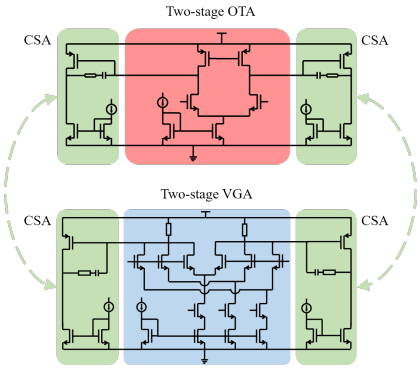


Fig. 1: A two-stage OTA composed by a one-stage OTA and two CSAs (Common Source Amplifiers). A two-stage VGA composed by a one-stage VGA and two CSAs.

The broad goal of this work is to develop ML-based analog performance modeling techniques that can simultaneously achieve high accuracy, fast estimation time, low model construction cost and high scalability. In general, ML techniques are intrinsically fast for circuit performance estimation and this is why we continue with ML techniques. We propose ML-based macro-models that can be reused across different circuits. The reuse can greatly amortize model construction cost. The sub-circuit level approach implies limited ML model size and complexity. Therefore, our approach helps improve both model accuracy and scalability. Since our macro-models are ML-based, they are quite different from conventional macro-models in early works [15].

Our ML model development for macro-models has a key difference from previous work of flat parameter-to-performance (P2M) mapping models [6], [7], [11]. For a flat P2M ML model, its outputs are simply the performance metrics of a circuit. For example, the outputs of an ML-based performance model for OTA can be gain, phase margin, bandwidth and unity gain frequency. By contrast, the outputs of ML-based macro-models need to be more general instead of specific performance metrics such that they can be applied for different circuits. In addition, we need to consider how to assemble macro-models to form the performance model for an entire circuit. To these ends, we develop parameters to output voltages (**P2V**) ML modeling techniques, whose mapping targets are time- and frequency-domain voltages. Another difference from flat models is that a macro-model needs to consider loading effects, which are resulted from input and output impedance from other connected sub-circuits.

MMM currently covers the following types of designs:

- *Linear analog ICs, such as OTAs.* For these circuits, frequency-domain modeling introduced in section III is able to estimate circuit performance including gain, unity gain frequency, bandwidth, etc. Besides, frequency-domain modeling is able to estimate circuit performance for any circuits that can be described by a linear time-invariant system.
- *ADC/DACs without feedback.* Our time-domain modeling in section IV can estimate the output voltages sequence of a circuit, given a sequence of input voltages in discrete time steps. We show the circuit performance estimation method for ADC/DACs without feedback.

To our knowledge, this is the first approach to scalable hierarchical modeling for AMS circuits. The above test cases provide a proof of concept that the method presents a promising direction that can be extended to other types of circuits in future.

The proposed MMM framework is structured as follows.

1) **Offline macro-model training.** Machine learning-based macro-models are trained for a set of commonly used sub-circuits. Almost any machine learning engines can be adopted, e.g., neural network and random forest. The labels of training data are obtained through circuit simulations.
2) **Macro-model-based circuit system performance estimation.** Given a circuit system netlist, its performance estimation is performed as follows.
   a) *Circuit partitioning and matching.* The given circuit system is partitioned into sub-circuits, whose corresponding macro-models are identified from the trained models. This step is similar to and simpler than technology mapping of digital ICs. Also, the number of sub-circuits in an analog or mixed-signal IC is substantially smaller than the logic cells in a digital IC.
   b) *Assembling macro-models for system performance estimation.* This step will be introduced with details in Sections III and IV for linear circuits and ADC/DACs.

III. Frequency-Domain Models for Linear ICs

The behavior of a linear time-invariant (LTI) system [15] can be described by its s-domain transfer function $H(s)$ as

$$H(s) = \frac{Y(s)}{X(s)} = K\frac{(s-z_1)(s-z_2)\cdots(s-z_m)}{(s-p_1)(s-p_2)\cdots(s-p_n)} \quad (1)$$

where $X(s)$ and $Y(s)$ are the input and output signals, $z_i, i = 1, \cdots, m$ and $p_i, i = 1, \cdots, n$ are zeros and poles, and K is a constant factor. In small signal analysis, many analog ICs can be treated as LTI systems at certain DC operating points. As such, the circuit behaviors can be described by the s-domain transfer function of Equation (1).

Our frequency-domain MMM for circuit performance estimation consists of the following main steps.

1) *DC voltage modeling.* An ML model is built for each sub-circuit to estimate its DC output voltages for given DC input voltages. The models for all sub-circuits of a system are connected to obtain the DC operating points for all sub-circuits.
2) *Parametric transfer function-based macro-modeling.* According to the DC operating point, transfer function of each sub-circuit is derived, whose parameters are obtained through ML models.

3) *Obtaining the circuit/system transfer function* by assembling the macro-models.
4) *Circuit performance estimation* through the circuit system transfer function.

This methodology considers both variable loading effect and feedback structure in circuits. Although multiple ML models need to be built in these steps, they all share the same training data and the model training time is much shorter than the data preparation time. We use an example of two-stage OTA to illustrate these steps.

A. ML-Based DC Voltage Modeling

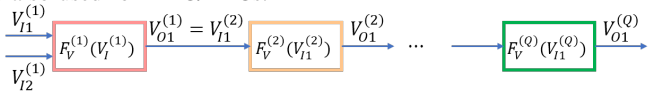
For an M-input N-output sub-circuit, an ML model F_V is built to estimate its DC output voltages for the input features of (i) DC input voltages, (ii) device sizes, (iii) current source/voltage bias and loads from its surrounding circuits, and can be described by

$$\begin{aligned}\boldsymbol{V}_O &= [V_{O1}, V_{O2}, \cdots, V_{ON}]^T \\ &= F_V(V_{I1}, V_{I2}, \cdots, V_{IM}; \boldsymbol{S}, \boldsymbol{L}) = F_V(\boldsymbol{V}_I; \boldsymbol{S}, \boldsymbol{L})\end{aligned} \tag{2}$$

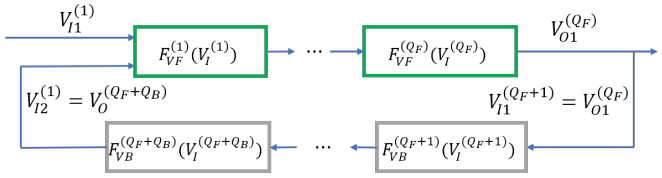
where $\boldsymbol{V}_I$ is an M-dimensional vector of DC input voltages, $\boldsymbol{V}_O$ is an N-dimensional vector of DC output voltages, $\boldsymbol{S}$ is the vector for transistor sizes and $\boldsymbol{L}$ is a vector representing loads and current source/voltage bias.

Once the ML-based DC voltage models are constructed for all sub-circuits, the DC operating point for each sub-circuit is obtained based on whether there is a cascade connection, feedback connection, or a mix of these two kinds of connections.

Cascade connections are illustrated in Figure 2(a), where each rectangle indicates the DC voltage model of a sub-circuit. The given primary DC input voltages at the first sub-circuit can be propagated through all cascade stages using the ML models of Equation (2) to obtain the DC input/output voltage of all sub-circuits, i.e., their DC operating points. The ML-based DC voltage models in Equation (2) is also used for ADC/DACs.



(a) Cascade connection



(b) Feedback connection

Fig. 2: Two examples of connection structures: a cascade connection and a feedback connection.

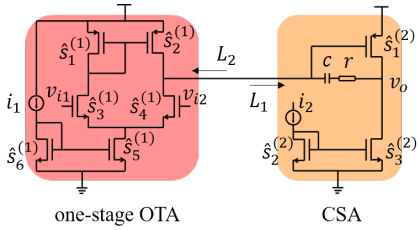
A feedback connection is shown in Figure 2(b), which has Q_F feedforward stages and Q_B feedback stages. The derivation of DC operating points for circuits with feedback connections is an iterative process. The feedback output, $V_O^{(Q_F+Q_B)} = V_{I2}^{(1)}$ in Figure 2(b), is initialized with a guessed value. Then, the ML-based sub-circuit DC voltage models (2) are applied through each stage of the loop iteratively till the voltage values converge, which is always observed in our experiments.

Figure 3 shows an example of two-stage OTA composed by a one-stage OTA and a CSA (Common Source Amplifier), whose DC voltage models are represented by $F_V^{(1)}$ and $F_V^{(2)}$ in Figure 3(b), respectively. $V_{I1}^{(1)}$ and $V_{I2}^{(1)}$ are two inputs of the one-stage OTA, while $V_{O1}^{(1)}$ is its output. Similarly, $V_{I1}^{(2)}$ and $V_{O1}^{(2)}$ are input and output of the CSA.

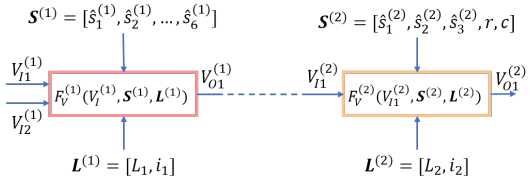
The models $F_V^{(1)}$ and $F_V^{(2)}$ are trained individually, i.e., training data samples are generated separately for the two macro-models. With the two trained DC voltage macro-models, the system-level DC voltage model is:

$$\begin{aligned}V_{O1}^{(2)} &= F_V^{(2)}(V_{I1}^{(2)}, \boldsymbol{S}^{(2)}, \boldsymbol{L}^{(2)}) \\ &= F_V^{(2)}(V_{O1}^{(1)}, \boldsymbol{S}^{(2)}, \boldsymbol{L}^{(2)}) \\ &= F_V^{(2)}(F_V^{(1)}(\boldsymbol{V}_I^{(1)}, \boldsymbol{S}^{(1)}, \boldsymbol{L}^{(1)}), \boldsymbol{S}^{(2)}, \boldsymbol{L}^{(2)})\end{aligned} \tag{3}$$

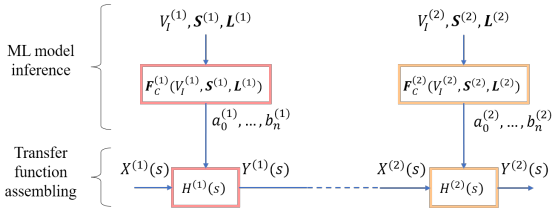
where $\boldsymbol{V}_I^{(1)} = [V_{I1}^{(1)}, V_{I2}^{(1)}]$ is the DC input voltage vector of one-stage OTA. $\boldsymbol{S}^{(1)}$ is the vector of device sizes in one-stage OTA, while $\boldsymbol{L}^{(1)}$ is the vector of current source/voltage bias and surrounding circuits' loads in one-stage OTA. $\boldsymbol{S}^{(2)}$ and $\boldsymbol{L}^{(2)}$ have similar definitions in CSA.



(a) Two-stage OTA schematic



(b) DC voltage macro-model



(c) Transfer function macro-model

Fig. 3: A two-stage OTA, corresponding DC voltage macro-model connection and system transfer function modeling.

B. ML-Based Parametric Transfer Function Macro-Models

The transfer function (1) can be rewritten as

$$H(s) = \frac{a_0 + a_1 s + \cdots + a_m s^m}{1 + b_1 s + \cdots + b_n s^n} \tag{4}$$

where $a_i, i = 0, \cdots, m$ and $b_j, j = 1, \cdots, n$ are constant coefficients. The values of m and n are constants determined by the circuit topology and can be obtained from circuit simulation by running SPICE pole/zero analysis. Typically, m and n are less than 20. The macro-model for each sub-circuit is described by transfer function (4), where the coefficients are functions described by ML models instead of constants. Therefore, the transfer function is parametric. Specifically, an ML model is built for each coefficient a_i or b_j using input features (i) DC input voltage of the sub-circuit, (ii) device sizes, and (iii) current source/voltage bias and loads from its surrounding circuits, i.e.,

$$\begin{aligned}&a_0, a_1, \cdots, a_m, b_1, \cdots, b_n \\ &= \boldsymbol{F}_C(\boldsymbol{V}_I; \boldsymbol{S}, \boldsymbol{L}) \\ &= [F_{C_0}(\boldsymbol{V}_I; \boldsymbol{S}, \boldsymbol{L}), ..., F_{C_{m+n}}(\boldsymbol{V}_I; \boldsymbol{S}, \boldsymbol{L})]\end{aligned} \tag{5}$$

where $\boldsymbol{F}_C$ indicates a vector of ML models. Each element in $\boldsymbol{F}_C$ is an ML model for one coefficient a_i or b_j in the transfer function. $\boldsymbol{S}$ is the vector for device sizes, $\boldsymbol{L}$ denotes the vector of loads and current

source/voltage bias, and $\boldsymbol{V}_I$ is the DC input voltage of the sub-circuit. Please note that DC input voltages of sub-circuits are obtained according to Section III-A. Labels of training data are obtained from pole/zero analysis through circuit simulations.

The transfer functions of the one-stage OTA and the CSA in Figure 3 are represented by $H^{(1)}(s)$ and $H^{(2)}(s)$, respectively, which are parametric upon ML models $\boldsymbol{F}_C^{(1)}$ and $\boldsymbol{F}_C^{(2)}$.

C. Circuit System Transfer Function

For a circuit composed by Q cascaded sub-circuits (Figure 2(a)) with macro-models $H^{(i)}(s), i = 1, \cdots, Q$, the system transfer function is given by

$$G(s) = H^{(1)}(s)H^{(2)}(s)\cdots H^{(Q)}(s) \tag{6}$$

For the two-stage OTA in Figure 3, its circuit system transfer function is $G(s) = H^{(1)}(s)H^{(2)}(s)$, as shown in Figure 3(c).

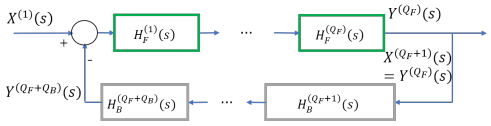


Fig. 4: A system with feedback connections among sub-circuits. Each block indicates a sub-circuit.

For a circuit with one feedback loop, whose block diagram is shown in Figure 4, its transfer function consists of two parts:

- Feedforward transfer function
  $H_F(s) = H_F^{(1)}(s)H_F^{(2)}(s)\cdots H_F^{(Q_F)}(s)$
- Feedback transfer function
  $H_B(s) = H_B^{(Q_F+1)}(s)H_B^{(Q_F+2)}(s)\cdots H_B^{(Q_F+Q_B)}(s)$

The transfer function of the overall circuit is given by [16]

$$G(s) = \frac{Y^{(Q_F)}(s)}{X^{(1)}(s)} = \frac{H_F(s)}{1 + H_F(s)H_B(s)} \tag{7}$$

For a circuit with multiple cascaded paths and feedback loops, the transfer function can be obtained according to [16].

D. Circuit Performance Estimation

Given a circuit transfer function $G(s)$, its performance estimation varies depending on different circuits and we illustrate this process using OTA as an example. Since $s = j\omega$, $G(s)$ can be expanded as

$$G(j\omega) = |G(j\omega)|\angle G(j\omega) \tag{8}$$

where $|G(j\omega)|$ and $\angle G(\omega)$ are the amplitude and phase angle, respectively. Next, circuit performance of OTA, including gain, UGF (Unity Gain Frequency), BW (Bandwidth) and PM (Phase Margin), can be obtained by sweeping frequency ω and

$$\begin{aligned}
&\text{Gain} := 20\log|G(j\cdot 0)| \text{ dB}\\
&\text{UGF} := \omega \text{ when } |G(j\cdot\omega)| = 1\\
&\text{BW} := \omega \text{ when } 20\log|G(j\cdot\omega)| - 20\log|G(j\cdot 0)| = -3\text{dB}\\
&\text{PM} := \angle G(j\omega) + 180^\circ \text{ when } |G(j\cdot\omega)| = 1
\end{aligned} \tag{9}$$

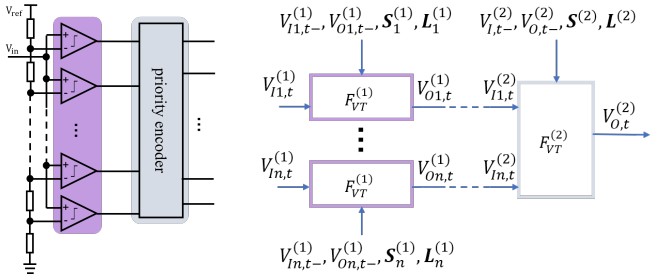
The performance of other kinds of LTI circuits can be estimated in a similar manner.

IV. Time-Domain Models for ADC/DACs

A. Time-Domain Macro-Models

The sub-circuits of ADCs are comparators and an encoder, which are illustrated in Figure 5. The sub-circuits of most ADCs can be identified in a similar way.

Given an input signal sequence where voltage varies at discrete time steps, the time-domain macro-model estimates the output voltage



(a) Flash ADC schematic (b) Flash ADC macro-models

Fig. 5: A flash ADC with its sub-circuits and macro-models.

vector $\boldsymbol{V}_{O,t}$ at current time step t based on features including (i) input voltage vector $\boldsymbol{V}_{I,t}$ of time t, (ii) input voltage vector $\boldsymbol{V}_{I,t^-}$ of the previous time step t^-, (iii) output voltage vector $\boldsymbol{V}_{O,t^-}$ of the previous time step t^-, (iv) device size vector $\boldsymbol{S}$, and (v) load vector $\boldsymbol{L}$. This can be described as

$$\boldsymbol{V}_{O,t} = F_{VT}(\boldsymbol{V}_{I,t}, \boldsymbol{V}_{I,t^-}, \boldsymbol{V}_{O,t^-}; \boldsymbol{S}, \boldsymbol{L}) \tag{10}$$

where each element of $\boldsymbol{V}_{I,t}$ or $\boldsymbol{V}_{O,t}$ indicates input/output voltage of one sub-circuit terminal at time step t. This is an ML model trained by data from circuit simulation. After this model is built, for a given sequence of input voltages in discrete time steps, the output voltage sequences can be obtained. Unlike DC voltage modeling in Equation (2), the time-domain macro-model also considers the voltage in the previous time step because it affects subsequent voltages, e.g., a capacitor voltage depends on the voltage in the previous time step as well as the accumulated electric charge in the current time step.

Figure 5(b) demonstrates the macro-models of a flash ADC. The flash ADC contains n comparators modeled by an ML model $F_{VT}^{(1)}$ and one priority encoder modeled by another ML model $F_{VT}^{(2)}$. The n comparators are identical and share the same model $F_{VT}^{(1)}$. The use of this shared macro-model greatly improves data preparation efficiency compared to flat ML models, where the same comparator is simulated n times. For the i-th comparator, its macro-model's input features includes input voltage $V_{Ii,t}^{(1)}$ at time step t as well as input voltage $V_{Ii,t^-}^{(1)}$ and output voltage $V_{Oi,t^-}^{(1)}$ of the previous time step t^-. In addition, device size $\boldsymbol{S}_i^{(1)}$ and surrounding circuits' load $\boldsymbol{L}_i^{(1)}$ are also included in the features. The macro-model estimates the output voltage $\boldsymbol{V}_{O,t}$ at time step t. The macro-model of priority encoder has similar input features while the difference is that the size of its input voltage $[\boldsymbol{V}_{I1,t}^{(2)}, ..., \boldsymbol{V}_{I1,t}^{(2)}]$ is significantly greater than the size of a comparator macro-model's input feature.

B. ADC/DAC Performance Estimation

For an ADC/DAC without feedback, its output waveform in discrete time can be obtained through propagating the input waveform through the sub-circuits using corresponding macro-models described in Section IV-A. Next, a Fast Fourier transform (FFT) is performed on the waveforms at the output node to obtain the spectrum. Then, the SFDR (Spurious Free Dynamic Range) and SNDR (Signal to Noise + Distortion Ratio) performance metrics are computed by [16]

$$\begin{aligned}
SFDR &= 20 \times log\frac{A_F}{A_W}\\
SNDR &= 20 \times log\frac{A_F}{(\sum A_N^2 + \sum A_D^2)^{\frac{1}{2}}}
\end{aligned} \tag{11}$$

where A_F is the amplitude of fundamental component in the spectrum, A_W is the amplitude of the worst spur signal, and A_N and A_D are amplitudes of noise and distortion, respectively.

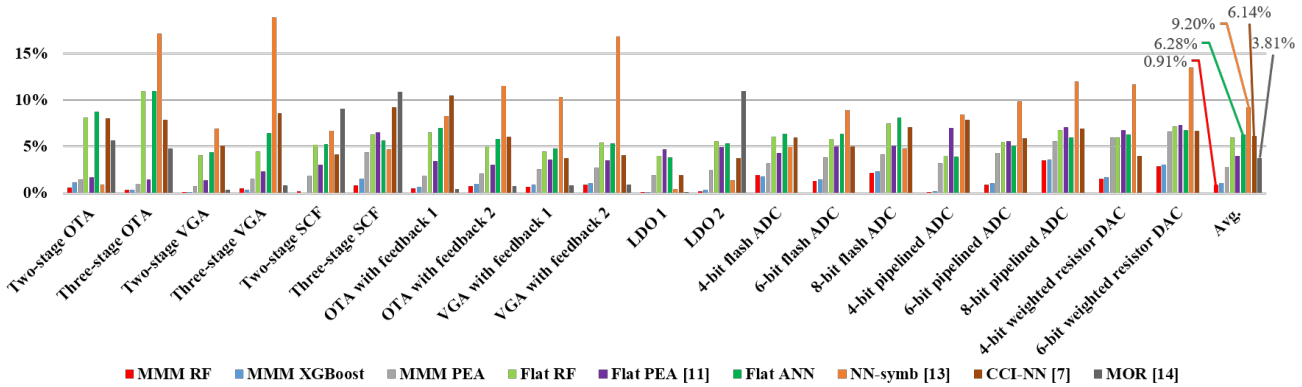


Fig. 6: Circuit performance estimation error compared with SPICE.

For ADCs, DNL (Differential nonlinearity) and INL (Integral nonlinearity) are obtained by sweeping the input voltage V_I in full scale range (or all digital code for DAC) [17]:

$$DNL = (V_i - V_{i-1})/V_{LSB} - 1$$
$$INL = (V_i - V_i^{ideal})/V_{LSB} \quad (12)$$

where $0 \leq i \leq 2^N - 1$, V_i is the input voltage for ADC's i-th code (or output voltage for DAC's i-th code), $V_{LSB} = \frac{V_{FSR}}{2^N}$, V_i^{ideal} is the ideal voltage for the i-th code. V_{FSR} is the full scale range voltage. When sweeping V_I, only Equation (2) is required to estimate the output voltage because the input voltage is in DC steady state.

V. Experiments

We evaluate the approach on the testcases listed in Table I, where SCF is a switched capacitor filter and LDO is a low dropout regulator. The upper part covers linear analog ICs and the lower part is a list of ADC/DACs. The designs are based on a commercial 12nm process technology. All training data are obtained through SPICE simulations. The experiments are conducted on a Linux machine with a Xeon E5-2680 V2 processor, 2.8GHz frequency and 256G memory.

Circuit	#Transistors	Sub-circuits/Macro-models
Two-stage OTA	9	5-transistor OTA, common source amplifier
Three-stage OTA	13	5-transistor OTA, common source amplifier
Two-stage VGA	15	One-stage VGA, common source amplifier
Three-stage VGA	19	One-stage VGA, common source amplifier
Two-stage SCF	29	One-stage SCF, common source amplifier
Three-stage SCF	33	One-stage SCF, common source amplifier
OTA with feedback 1	10	5-transistor OTA, common source amplifier
OTA with feedback 2	12	5-transistor OTA, common source amplifier
VGA with feedback 1	16	One-stage VGA, common source amplifier
VGA with feedback 2	18	One-stage VGA, common source amplifier
LDO (Regulator) 1	7	5-transistor OTA, single transistor
LDO 2	13	One-stage VGA, single transistor
4-bit flash ADC	1076	Comparator, 4-bit priority encoder
6-bit flash ADC	4842	Comparator, 6-bit priority encoder
8-bit flash ADC	20468	Comparator, 8-bit priority encoder
4-bit pipelined ADC	276	Comparator, 5-transistor OTA
6-bit pipelined ADC	414	Comparator, 5-transistor OTA
8-bit pipelined ADC	552	Comparator, 5-transistor OTA
4-bit weighted resistor DAC	31	5-transistor OTA, inverter
6-bit weighted resistor DAC	35	5-transistor OTA, inverter

TABLE I: Circuit testcases.

A. Results of ML-Based Macro-Model Accuracy

Domains	Sub-circuits	Error RF	XGBoost	PEA [11]	ANN	CCI-NN [7]
Frequency	5-transistor OTA	0.00%	0.00%	0.54%	0.68%	0.48%
	One-stage VGA	0.01%	0.02%	0.85%	0.92%	0.76%
	One-stage SCF	0.01%	0.01%	0.64%	0.81%	0.69%
	CSA	0.03%	0.02%	0.27%	0.39%	0.28%
Time	4-bit encoder	1.10%	1.45%	4.35%	4.87%	3.84%
	6-bit encoder	1.26%	1.35%	4.29%	4.98%	4.09%
	8-bit encoder	1.43%	1.32%	5.86%	6.14%	5.61%
	Comparator	1.94%	2.35%	4.03%	5.06%	4.97%
Average		0.72%	0.81%	2.60%	2.98%	2.59%

TABLE II: Macro-model error compared to SPICE.

We study several options of ML engines for the macro-models, including RF (Random Forest), XGBoost and PEA [11], which is a GNN technique extended from classification to regression. Each macro-model is trained and tested for the same sub-circuit with 1000 data samples, of which 80% are for training and the other 20% are for testing. Please note that test data is not seen in training. Table II compares the errors of macro-models based on different ML engines. The result of each sub-circuit is averaged on all its macro-models. One can see that RF and XGBoost are more accurate than PEA while RF is the most accurate among them.

B. Results of Circuit Performance Estimation Accuracy

We compare the following methods:

- **MMM RF**. This is our proposed approach where macro-models are based on RF (Random Forest). Forest size and tree depth are set as 20 and 10, respectively.
- **MMM XGBoost**. Almost the same as MMM RF except that its ML engine is XGBoost.
- **MMM PEA**: Almost the same as MMM RF, but the ML engine is PEA [11] regression.
- **Flat RF**. A random forest model is trained for directly estimating the performance of an entire circuit. Forest size and tree depth are set as 400 and 10, respectively.
- **Flat PEA**. Almost the same as Flat RF except that a PEA [11] model is used as the ML engine.
- **Flat ANN**. An ANN model is trained for directly estimating the performance of an entire circuit like in [6]. The number of hidden layers and the number of neurons in each layer are set as 5 and 256, respectively.
- **NN-symb**. The previous work [13], which is symbolic analysis using neural network-based transistor models.
- **CCI-NN**. The recent previous work [7] aimed to training sample reduction. Similar experiment setup is used as provided in [7].
- **MOR**. An model order reduction technique [14], which is only tested on linear analog ICs.

The average performance estimation errors compared with SPICE are depicted in Figure 6. The error of one test circuit is the average percentage error among all its performance metrics, e.g., gain, UGF, bandwidth and phase margin for OTAs, VGAs and SCFs. For LDOs, PSRR (Power supply rejection ratio) is the performance metric; for ADC/DACs, performance metrics include SFDR, SNDR, DNL and INL. We use a sine wave at 1MHz as the input signal for ADC/DACs and apply an FFT at the output signal to obtain SFDR and SNDR. We see that MMM RF and MMM XGBoost achieve the smallest errors, less than 1% on average and significantly smaller than the 9% and 6% average errors from NN-symbolic and CCI-NN, respectively. Estimation errors for some specific performance metrics are shown in Figure 7 and Figure 8, for UGF of linear analog ICs and SFDR (Spurious Free Dynamic Range) of ADC/DACs, respectively, where similar trends can be observed.

These results partially confirm the observation [7] that the accuracy of flat ANN performance models tends to degrade for large circuits or wide performance range. MMM attains significantly higher accuracy as its ML models are trained for sub-circuits that have lower complexity than entire circuits.

Circuit	Training data preparation time (hour)					ML model training time (sec)						Performance estimation time (sec)							
	MMM RF	Flat PEA	Flat RF/ANN	CCI-NN [7]	NN-symb [13]	MMM RF	Flat RF	Flat PEA	Flat ANN	NN-symb [13]	CCI-NN [7]	MMM RF	Flat RF	Flat PEA	Flat ANN	NN-symb [13]	SPICE	CCI-NN [7]	MOR [14]
Two-stage OTA	0.11	0.76	0.70	0.14	0.67	0.30	2.05	45.56	15.37	10.76	5.53	0.004	0.032	0.015	0.001	0.07	1.22	0.01	0.001
Three-stage OTA	0.15	0.99	0.95	0.19	0.97	0.36	3.53	47.10	21.51	14.43	5.29	0.010	0.039	0.018	0.001	0.08	1.63	0.02	0.004
Two-stage VGA	0.14	0.77	0.72	0.14	1.11	0.13	4.18	43.11	18.30	19.35	13.31	0.005	0.049	0.023	0.003	0.08	1.24	0.04	0.021
Three-stage VGA	0.18	0.81	0.78	0.16	1.41	0.19	5.46	46.66	23.23	25.46	13.82	0.010	0.076	0.027	0.003	0.08	1.32	0.02	0.022
Two-stage SCF	0.22	0.87	0.86	0.17	2.15	0.50	5.35	43.75	35.62	43.50	3.07	0.004	0.097	0.064	0.003	0.08	1.43	0.03	1.458
Three-stage SCF	0.26	0.84	0.83	0.17	2.45	0.57	6.50	46.81	43.41	39.27	8.01	0.011	0.035	0.083	0.002	0.08	1.35	0.02	1.527
OTA with feedback 1	0.11	0.66	0.62	0.12	0.74	0.30	2.23	48.95	25.47	11.50	4.04	0.005	0.033	0.012	0.003	0.15	1.03	0.03	0.004
OTA with feedback 2	0.11	0.93	0.91	0.18	0.89	0.30	2.48	45.18	30.75	14.04	7.16	0.008	0.039	0.016	0.002	0.19	1.54	0.04	0.022
VGA with feedback 1	0.14	0.83	0.81	0.16	1.19	0.13	3.44	48.56	43.74	21.44	1.70	0.006	0.049	0.023	0.003	0.25	1.32	0.02	0.022
VGA with feedback 2	0.14	0.79	0.82	0.16	1.34	0.13	3.73	43.48	52.28	22.86	7.22	0.007	0.054	0.029	0.004	0.28	1.29	0.02	0.004
LDO 1	0.06	0.86	0.86	0.17	0.52	0.20	1.65	35.94	39.47	6.510	13.49	0.011	0.030	0.008	0.002	0.15	1.43	0.01	0.001
LDO 2	0.08	0.82	0.84	0.17	0.97	0.06	3.01	33.16	43.38	12.35	9.48	0.009	0.022	0.012	0.003	0.45	1.37	0.03	0.004
4-bit flash ADC	5.29	76.15	76.85	15.37	80.00	4.81	112	63.84	480.7	860.8	57.24	0.007	0.083	8.500	0.004	18.54	136.9	0.02	-
6-bit flash ADC	7.34	291.3	292.1	58.42	359.8	7.69	650	211.3	688.9	4648	189.20	0.026	0.188	27.48	0.002	188.1	523.6	0.06	-
8-bit flash ADC	9.56	966.0	965.9	193.2	1522	8.00	2081	914.5	763.6	15146	714.98	0.084	0.364	109.9	0.003	1745	1736	0.17	-
4-bit pipelined ADC	0.005	21.46	21.42	4.28	20.51	0.003	23.4	32.84	8.90	245.6	23.20	0.023	0.060	1.039	0.003	13.85	38.53	0.02	-
6-bit pipelined ADC	0.005	65.07	65.01	13.00	30.75	0.004	42.0	43.19	9.89	434.7	29.82	0.037	0.076	2.851	0.002	37.78	116.98	0.03	-
8-bit pipelined ADC	0.005	93.54	93.46	18.69	40.99	0.006	51.5	58.61	13.07	623.8	48.22	0.053	0.059	5.690	0.003	56.81	168.19	0.05	-
4-bit weighted resistor DAC	0.008	7.01	6.98	1.40	2.30	0.001	5.27	35.08	18.90	30.69	30.74	0.017	0.038	0.059	0.005	10.85	12.50	0.02	-
6-bit weighted resistor DAC	0.008	16.13	16.10	3.22	2.60	0.001	6.75	39.85	20.07	36.05	35.76	0.026	0.042	0.073	0.006	26.95	28.91	0.03	-
Normalized average	1.0	1791.0	1788.9	357.8	885.5	1.0	1975.2	5469.1	2462.0	18255.5	4520.5	1.0	6.2	193.6	0.3	1873.9	3639.1	2.9	42.3

TABLE III: Comparisons of data preparation time, model training time and performance estimation time.

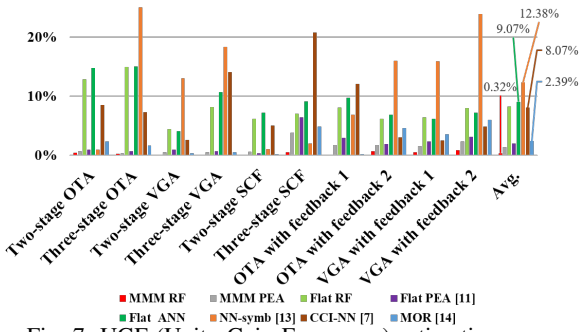


Fig. 7: UGF (Unity Gain Frequency) estimation errors.

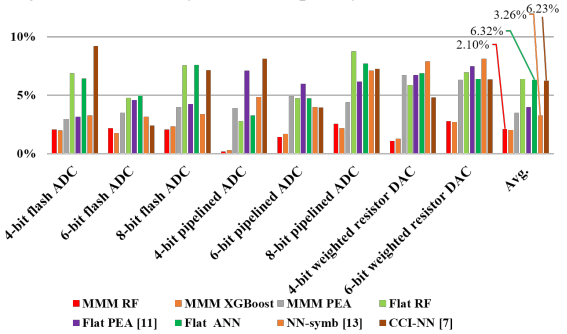


Fig. 8: SFDR estimation errors.

C. Model Construction and Circuit Performance Estimation Runtime

Circuit performance estimation runtime comparisons for different methods are provided in the rightmost section of Table III. The runtime of MMM XGBoost is almost the same as MMM RF and not included here. Among the 8 methods being compared, MMM RF is in the second place for performance estimation runtime and $2.9\times$ faster than the recent approach CCI-NN [7]. It is also $42\times$ than MOR for linear circuits. Flat ANN is faster than MMM RF, however, its performance estimation errors are significantly larger. Flat RF is slower than MMM RF because it requires more trees to model an entire circuit system.

Model construction time consists of the time for obtaining training data and model training. The former dominates the latter one as training data preparation needs to run many circuit simulations. Since our macro-models are reused in different circuits, their data preparation time and model training time reported in Table III are amortized, e.g., the time is scaled by $1/k$ if the model is reused for k circuits in our testcases. The advantage of our proposed MMM RF on model construction time is huge, orders of magnitude smaller than the other methods. In particular, the data preparation time of our MMM RF is $357\times$ less than CCI-NN [7], which has a similar goal as ours. For the largest case of 8-bit flash ADC, MMM RF can reduce the data preparation time from about 40 days to 9.6 hours. The model construction time of MMM XGBoost is similar to MMM RF and not included due to space limit.

VI. Conclusions

Although machine learning-based models has advanced the state-of-the-art for fast analog performance estimation, existing approaches are mostly flat models that suffer from huge model construction cost and low reusability. This work introduces macro-model level machine learning techniques to address the problems for linear analog ICs and ADC/DACs. Experimental results on circuits with up to 20K devices show that our approach can reduce model construction cost by three orders of magnitude compared to recent ML techniques. At the same time, it achieves significantly smaller errors and is three orders of magnitude faster than circuit simulation.

Acknowledgement

This project is partially supported by NSF CCF-2106725, CCF-2212346 and CCF-2212345.

References

[1] G. Gielen *et al.*, "Symbolic analysis methods and applications for analog circuits: A tutorial overview," *Proceedings of the IEEE*, 1994.

[2] M. Celik *et al.*, *IC Interconnect Analysis*. Boston, MA: Kluwer, 2002.

[3] C.-J. Shi *et al.*, "Canonical symbolic analysis of large analog circuits with determinant decision diagrams," *IEEE TCAD*, 2000.

[4] W. Daems *et al.*, "Simulation-based automatic generation of signomial and posynomial performance models for analog integrated circuit sizing," in *Proc. ICCAD*, 2001.

[5] F. De Bernardinis *et al.*, "Support vector machines for analog circuit performance representation," in *Proc. DAC*, 2003.

[6] J. Liu *et al.*, "Transfer learning with Bayesian optimization-aided sampling for efficient ams circuit modeling," in *Proc. ICCAD*, 2020.

[7] M. Hassanpourghadi *et al.*, "Circuit connectivity inspired neural network for analog mixed-signal functional modeling," in *Proc. DAC*, 2021.

[8] S. Kamineni *et al.*, "AuxcellGen: A framework for autonomous generation of analog and memory unit cells," in *Proc. DATE*, 2023.

[9] M. Liu *et al.*, "Towards decrypting the art of analog layout: Placement quality prediction via transfer learning," in *Proc. DATE*, 2020.

[10] H. Wang *et al.*, "GCN-RL circuit designer: Transferable transistor sizing with graph neural networks and reinforcement learning," in *Proc. DAC*, 2020.

[11] Y. Li *et al.*, "A customized graph neural network model for guiding analog ic placement," in *Proc. ICCAD*, 2020.

[12] Y. Lin *et al.*, "Are analytical techniques worthwhile for analog ic placement?" in *Proc. DATE*, 2022.

[13] Z. Zhao *et al.*, "Efficient performance modeling for automated CMOS analog circuit synthesis," *IEEE TVLSI*, 2021.

[14] U. Baur *et al.*, "Model order reduction for linear and nonlinear systems: a system-theoretic perspective," *Archives of Computational Methods in Engineering*, 2014.

[15] R. A. Rutenbar *et al.*, "Hierarchical modeling, optimization, and synthesis for system-level analog and RF designs," *Proceedings of the IEEE*, 2007.

[16] A. V. Oppenheim *et al.*, *Signals & Systems*. Prentice hall Upper Saddle River, NJ, 1997.

[17] N. Karmokar *et al.*, "Constructive placement and routing for common-centroid capacitor arrays in binary-weighted and split DACs," *IEEE TCAD*, 2023.