Automated Deep Learning Platform for Accelerated Analog Circuit Design

Rahul Dutta*, Ashish James†, Salahuddin Raju‡, Yong-Joon Jeon§,
Chuan Sheng Foo¶, Kevin Tshun Chuan Chai‖

*‡§‖Institute Of Microelectronics (IME), A*STAR, Singapore
†¶Institute for Infocomm Research (I2R), A*STAR, Singapore
Email: {*duttar, §jeony, ‖chaitc} @ime.a-star.edu.sg, {†ashish_james, ¶foo_chuan_sheng}@i2r.a-star.edu.sg, ‡sraju@ieee.org

***Abstract*—We present an analog design framework for circuit sizing selection using neural networks. The proposed automated deep learning (ADL) platform uses neural networks (NN) as a differentiable surrogate model for circuit performance to model the nonlinear and high dimensional relationships between sizing performance and circuit performance in analog circuits. Gradient-based constrained optimization is then used to propose new sizing parameters for the desired design closure, which are then verified using EDA tools. If circuit performance falls short of desired performance, the simulation results from the EDA tools are also used as additional training data to update the neural network model for the next design iteration. The tight coupling between NN and EDA tools in an iterative design loop achieves multi-variate design closure and has the capability to synthesize circuits with a significantly reduced number of circuit simulations.**

***Index Terms*—Computer aided design, integrated circuits, algorithms, analog circuit design, deep learning, neural networks, and optimization.**

I. Introduction

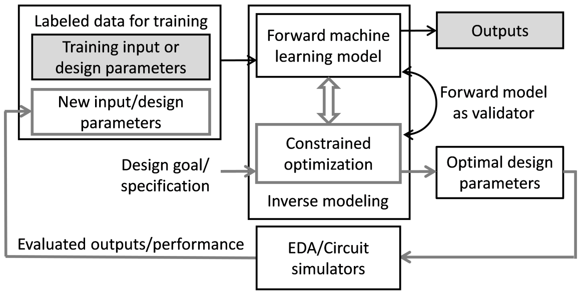
TRADITIONALLY, designing analog circuits has been a complicated process that requires tuning a large number of parameters and identifying underlying relationships among circuit performance metrics. In particular, an optimized design solution needs to balance trade-offs among performance metrics such as speed, area and power. To achieve overall design goals, designers typically go through many design iterations, guided by their theoretical knowledge and past experimental observations throughout the design process [1]. As circuit complexity increases with advanced technology nodes, the underlying design space that needs to be explored experiences a combinatorial explosion. As a result, designers will require more design iterations to achieve design goals. Moreover, in some cases even simulating these complex circuits is computationally expensive and time-consuming, further lengthening design cycles, increasing costs and time-to-market.

The analog circuit design process can be made more productive if knowledge of previous designs can be captured and used to automate the design optimization process [2]. This is a research topic of great interest to the community and there have been several efforts [3]–[13] towards this goal. In this context, analog design can be formulated as a nonlinear optimization problem where design parameters need to be tuned to achieve multiple objectives within certain design constraints. More concretely, an automated circuit synthesis system requires two key components: a performance model [3]–[6] that provides circuit performance given design parameters (which is used as the optimization objective), and an optimization algorithm [7]–[13] for subsequently finding the best design based on the performance model. One classic approach for analog circuit optimization utilizes the geometric programming framework [7], where circuit performance is modeled using posynomial and monomial functions and circuit sizing is done through convex optimization. Another approach is to use polynomials [8] for circuit performance modeling instead. Particle swarm optimization (PSO) was also previously used in [9], where circuit performance is defined by equations and used as a cost function for PSO to optimize. In these model-based approaches, the computational cost of evaluating the models is much lower than that of circuit simulations; however, such models are prone to deviate from real circuit performance. Bayesian optimization (BO) with more expressive Gaussian process as performance models have been shown to be extremely effective for circuit optimization [10]–[12]. Most recently, reinforcement learning has been proposed to optimize circuit design [13] and transistor sizing [14], in which a reward-based learning agent gradually learns towards an optimized solution. However, such approaches requires large number of data which makes it computationally expensive and time-consuming.

In this work, we explore the use of neural networks (NN) as performance models, different from previous efforts, to develop an automated deep learning (ADL) platform where learning, design optimization, and exploration can be done seamlessly with minimal inputs from the designers. Our ADL platform comprises a feed-forward NN for performance modeling and gradient-based optimization algorithm for design space exploration, coupled together with an off-the-shelf EDA platform. We validated the design acceleration by designing several analog circuits, such as operational amplifiers and integrated voltage regulator, using the ADL platform.

II. Overview of the ADL Platform

In this section, the ADL platform shown in Fig. 1 is introduced and the different building blocks are explained. We consider the input design $\mathbf{x}_C \in \mathbb{R}^D$ for a given circuit C to be controlled by D design parameters, for instance, device sizing and bias point. The figure of merit (FOM) of the analog circuit



Labeled data for training
Training input or design parameters
New input/design parameters
Forward machine learning model
Outputs
Forward model as validator
Constrained optimization
Inverse modeling
Design goal/ specification
Optimal design parameters
Evaluated outputs/performance
EDA/Circuit simulators

Fig. 1. Overview of ADL showing the process flow in the platform.

$\mathbf{y}_C \in \mathbb{R}^P$ is characterized by a combination of P performance metrics such as gain, noise, power etc.

The first and crucial phase in the ADL platform is accurate modeling of analog circuit performance. We formulate this as a multi-output regression task where we map a function from design inputs $\mathbf{x}_C$ to the performance outputs $\mathbf{y}_C$ from a set of N training samples $\{(\mathbf{x}_C^{(i)}, \mathbf{y}_C^{(i)})\}_{i=1}^N$. We choose to use feed-forward NNs as our function approximator due to their ability to model complex, non-linear and high dimensional relationships and as they are also differentiable, which enables us to use efficient gradient-based optimization procedures to find an optimal design using this performance model.

Briefly, NNs learn a function f that transform inputs $\mathbf{x}$ into outputs via a series of layers; formally, let $\mathbf{h}^{(l)}$ denote the vector of outputs from layer l of the neural network, which is also the input to layer $l+1$ (we define $\mathbf{h}^{(0)} = \mathbf{x}$ to be the inputs). Our networks consist of fully-connected layers, such that the output of layer l is computed as a composition of a matrix multiply of inputs with layer parameters $\mathbf{W}^{(l)}$ added to a bias parameter $\mathbf{b}^{(l)}$, followed by an element-wise non-linear activation function α: $\mathbf{h}^{(l)} = \alpha(\mathbf{W}^{(l)}\mathbf{h}^{(l-1)} + \mathbf{b}^{(l)})$. The sizes of layers l are defined by the dimension of the respective $\mathbf{h}^{(l)}$; these correspondingly determine the dimensions of $\mathbf{W}^{(l)}$ and $\mathbf{b}^{(l)}$. Sizes of network layers are model hyperparameters to be specified by the user. The output of an L layer network on input $\mathbf{x}$ is then $f(\mathbf{x};\theta) = \mathbf{h}^{(L)}$; in our experiments we used networks with 3 (hidden) layers. We then learn the parameters of the NN, $\theta = \{(\mathbf{W}^{(l)}, \mathbf{b}^{(l)})\}_{l=1}^L$, by minimizing the multi-output mean-squared error (MSE) between the NN's output and training samples as $\arg\min_\theta \sum_{i=1}^N \frac{1}{P}||\mathbf{y}_C^{(i)} - f(\mathbf{x}_C^{(i)};\theta)||_2^2$ using backpropagation [15].

Having learnt a performance model $f(\mathbf{x};\theta)$, we then use it as a surrogate for true circuit performance to find designs that achieve desired performance goals while satisfying design constraints, without the need for expensive simulation runs. We formulate this design optimization phase as a feasibility problem: given a set of performance goals $\mathcal{Y}_C$ and design constraints $\mathcal{X}_C$, we wish to find feasible $\mathbf{x} \in \mathcal{X}_C$ such that $f(\mathbf{x};\theta) \in \mathcal{Y}_C$. $\mathcal{X}_C$ is typically defined by bound constraints, for instance, for design currents I, $10\mu A \leq I \leq 100\mu A$, while $\mathcal{Y}_C$ is defined by specifying a target or one-sided bound constraints like Gain $\geq$ 55dB. Formally, we have $\mathcal{X}_C = \{\mathbf{x} \mid \mathbf{x}_{min} \leq \mathbf{x} \leq \mathbf{x}_{max}\}$ and without loss of generality we let $\mathcal{Y}_C = \{\mathbf{y}\}$ as the performance goals to achieve. We formulate this feasibility problem as the following optimization problem

$$\min_{\mathbf{x}} \ \lambda^T (\mathbf{y} - f(\mathbf{x};\theta))^2 \ \text{s.t.}\ \mathbf{x}_{min} \leq \mathbf{x} \leq \mathbf{x}_{max}. \qquad (1)$$

Here λ denotes a vector of non-negative weights that can be used to weight the various performance constraints. We set these to 1 in our experiments. We then solve this optimization problem with projected gradient descent (PGD) to handle the design constraints; projection onto bound constraints takes the following simple form which makes it extremely efficient to compute: $\Pi_{\mathcal{X}_C}(\mathbf{x}) = \min(\mathbf{x}_{max}, \max(\mathbf{x}_{min}, \mathbf{x}))$ where min, max are taken element-wise.

The candidate design $\mathbf{x}^*$ obtained through the constrained optimization in equation (1) is then fed into EDA tools for cross verification of circuit performance as depicted in Fig. 1. This step serves the dual role of verifying whether the desired circuit performance has been achieved (in which case the ADL platform design is complete), while at the same time generating an additional training sample for the performance model otherwise; in the latter case the performance model is re-trained using this enlarged training set for the next design iteration to generate a new candidate design. This performance model training, candidate generation, and verification plus additional training data generation loop continues until desired performance is achieved.

III. Performance Evaluation of the ADL Platform

1) Implementation of ADL: The implementation of the ADL platform is done using Google's TensorFlow and commercial EDA tool Cadence in a closed-loop so that the iterative process can be done seamlessly without any external assistance. To enable this closed-loop automated design, we have created a customized platform based on Python, in which we dynamically modify an OCEAN script; this OCEAN script contains the design variables and the script is to be executed in the Spectre simulator within Cadence to evaluate circuit performance; here only the design variables are modified by the API. Once the simulation is completed, the output performance data is used to update the feed-forward NN model, thereby closing the loop automatically without any help from the designer.

The feed-forward NN used in this paper has three hidden layers with size of 128, 256 and 128 respectively. The learning rate is set as 10^{-3} with a batch size of 20. In order to ensure good generalization ability of the model, the available data sets are split into 70% for training, 20% for testing and 10% for validation.

2) Experiment Setup: Initial training samples were generated with the help of designers. The allowable ranges for design inputs are given as $x_C^{min} \leq x_C \leq x_C^{max}$. For example, if the length of a transistor is considered as an input parameter in the design, there has to be an upper and lower bound for the length, because if the transistor length is so small, it cannot be manufactured; on the other hand, it should not be so large to occupy an unreasonably large area. These bounds are introduced to keep the machine learning model within the limits that are practically realizable. Initially, the feed-forward

NN trained on the data is used to predict the multivariate behavior of the circuit. Then the ADL platform is used for synthesizing the circuit for a new design goal as explained in Section II. The performance of the ADL platform is benchmarked against Bayesian optimization methods that have been proposed as an alternative to the current approach [10]–[12]. This is achieved by using the Bayesian optimization algorithm from the BayesOpt python package [16]. The function that need to be optimized is considered as the MSE between the desired and achieved performance metrics for the inputs constrained by the bounds. In this paper, only the exploration strategy based on upper confidence bound (UCB) is utilized for Bayesian optimization.

3) Performance Modeling Evaluation: We first validate the ability of NNs to model circuit performance. Fig. 2 shows the schematic of a two-stage operational amplifier design in 55nm CMOS process. In this example, there are 7 design parameters i.e. $x_C \in \mathbb{R}^7$, dedicated to device sizing and setting up the current bias. Devices for the biasing network were kept unchanged to ensure reasonable circuit operation. The performance metric of this circuit is defined by Gain, Bandwidth (BW), Phase Margin (PM), Slew Rate (SR), Power Supply Rejection Ratio (PSRR), and Common Mode Rejection Ratio (CMRR), i.e. $y_C \in \mathbb{R}^6$. An initial dataset of 6912 labeled data are available for training the NN model.

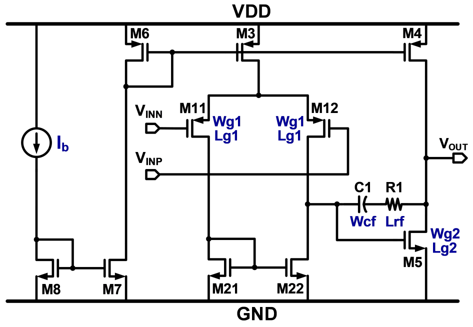


Fig. 2. Schematic diagram of a two-stage operational amplifier.

Table I shows the prediction performance of the feed-forward NN regressor in terms of the mean square error (MSE) and the mean absolute percentage error (MAPE) on the test set. To benchmark the prediction capability of the feed-forward NN, its performance is compared with Gaussian process and random forest (RF) regressors. Overall, the prediction error for feed-forward NNs is significantly lower than other regressors. This corroborates the use of NN as the forward model (validator during inverse optimization) in the ADL platform.

TABLE I
ACCURACY OF THE NN REGRESSOR FOR THE TWO-STAGE OPERATIONAL AMPLIFIER

Outputs	GP MSE	GP MAPE	RF MSE	RF MAPE	NN MSE	NN MAPE
BW	2.45×10^9	29.17	1.3×10^9	14.1	1.1×10^8	3.1
Gain	0.13	0.53	1.95	1.99	0.007	0.12
SR	2.25×10^{12}	2.43	1.5×10^{12}	1.83	9.3×10^{11}	1.79
PM	40.09	6.97	47.7	8.02	1.17	1.35
PSRR	46.10	5.70	20.01	3.41	0.248	0.29
CMRR	35.70	7.35	16.02	4.54	0.128	0.45

Next, we perform experiments on a more complex folded cascade operational amplifier, a schematic of which is shown in Fig. 3. In this example, 13 design parameters, i.e. $x_C \in \mathbb{R}^{13}$, are used for device sizing and current biasing. The performance metric of this circuit is defined using Gain Bandwidth Product (GPW), PM, SR, Input Common-Mode Range (ICMR) voltage, PSSR, CMMR, Noise and Power, i.e. $y_C \in \mathbb{R}^8$. We note that there are more design parameters and performance specifications used in this example compared to the two-stage op-amp circuit. The number of labeled samples available for this example is around 2000 with results presented on the 400 sample test set (20% of data). The trained NN model predicts all 8 outputs on the test set more accurately than the RF and GP models, as can be seen from Table II.

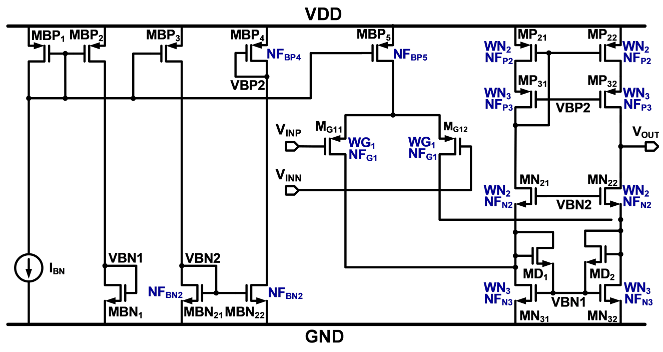


Fig. 3. Schematic diagram of a folded cascode operational amplifier.

TABLE II
ACCURACY OF THE NN REGRESSOR FOR THE FOLDED CASCODE OPERATIONAL AMPLIFIER

Outputs	GP MSE	GP MAPE	RF MSE	RF MAPE	NN MSE	NN MAPE
GBWP	6.45×10^{11}	0.66	7.2×10^{11}	1.16	1.5×10^{11}	0.56
PM	3.68	1.85	5.43	3.04	0.169	0.48
Noise	1.96×10^{-17}	0.95	3.5×10^{-16}	5.4	1.1×10^{-17}	0.79
SR	4.06×10^{11}	1.05	1.4×10^{11}	1.0	3.1×10^{10}	0.52
CMRR	25.92	2.49	34.54	3.7	10.2	1.43
PSRR	0.86	1.09	8.33	4.07	0.27	0.69
ICMR	2.31×10^{-5}	0.19	3.9×10^{-4}	0.84	9.5×10^{-6}	0.12
Power	13.6×10^{-10}	6.65	7.2×10^{-10}	9.89	3.4×10^{-11}	2.16

4) Design Optimization Evaluation: Having validated that NNs can perform as good circuit performance models, we proceed to evaluate the overall ADL platform. This was done across 3 seeds to validate the proposed approach. We first start with the two-stage op-amp. The performance objective of the circuit sizing optimization or the design goals were set as: $\hat{Y}_C \in \{\mathrm{BW} \geq 1\ \mathrm{MHz}, \mathrm{Gain} \geq 55\mathrm{dB}, \mathrm{PM} \geq 50°, \mathrm{SR} \geq 35\ \mathrm{V}/\mu\mathrm{s}, \mathrm{PSRR} \geq 70\mathrm{dB},\ \text{and}\ \mathrm{CMRR} \geq 60\mathrm{dB}\}$. Here, the BW requirement is set aggressively compared to other design goals. The range of input parameters are set as: $10\mu\mathrm{A} \leq I_b \leq 100\mu\mathrm{A}$, $60\mathrm{nm} \leq L_{g1} \leq 10\mu\mathrm{m}$, $0.5\mu\mathrm{m} \leq W_{g1} \leq 50\mu\mathrm{m}$, $60\mathrm{nm} \leq L_{g2} \leq 10\mu\mathrm{m}$, $0.5\mu\mathrm{m} \leq W_{g2} \leq 50\mu\mathrm{m}$, $2\mu\mathrm{m} \leq L_{rf} \leq 100\mu\mathrm{m}$, and $2\mu\mathrm{m} \leq W_{cf} \leq 100\mu\mathrm{m}$. The simulation iterations of the ADL platform to achieve the design goal $\hat{Y}_C$ is shown in Fig. 4. There were 6 output objectives that had to be met concurrently. Here, the achieved design goal is defined as the average percentage of each design objective that has been achieved. As an example, if the achieved design goals are: $\hat{Y}_C \in \{\mathrm{BW} = 0.9\ \mathrm{MHz}, \mathrm{Gain} = 45\mathrm{dB}, \mathrm{PM} = 30°, \mathrm{SR} = 40\ \mathrm{V}/\mu\mathrm{s}, \mathrm{PSRR} = 74\mathrm{dB},\ \text{and}\ \mathrm{CMRR} = 63\mathrm{dB}\}$, the percentage of design goal achieved is $(\frac{0.9}{1.0} + \frac{45}{55} + \frac{30}{50} + 1 + 1 + 1)/6 \times 100\% = 88.64\%$. Only when all design objectives are satisfied, the design goal achieved will be 100%. Both the

ADL and Bayesian optimization start with the same number of initial samples (100) before starting the design optimization process. The ADL platform took only 200 iterations to achieve the desired design goal $\hat{Y}_C$ compared to around 780 by the Bayesian optimization approach.

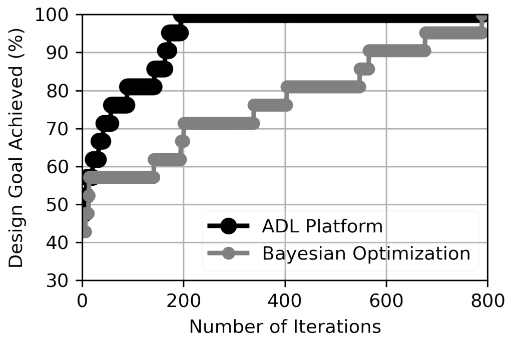


Fig. 4. Simulation iterations for the design closure of two-stage operational amplifier using the ADL platform and Bayesian optimization.

Similarly, the ADL platform was used to design a folded cascade operational amplifier with the following design goals: $\hat{Y}_C \in \{\mathrm{GBWP} \geq 150\,\mathrm{MHz}, \mathrm{PM} \geq 50°, \mathrm{SR} \geq 45\,\mathrm{V}/\mu\mathrm{s}, \mathrm{PSRR} \geq 60\,\mathrm{dB}, \mathrm{CMRR} \geq 80\,\mathrm{dB}, \mathrm{ICMR} \geq 1.9\mathrm{V}, \mathrm{Noise@1KHz} \leq 0.15\mu\mathrm{V}/\sqrt{Hz},$ and $\mathrm{Power} \leq 300\,\mu\mathrm{W}\}$. Here, 8 outputs objective has to be met concurrently out of which three outputs, GBWP, Noise, and SR, requirement were set aggressively compared to other design goals. The ADL platform took around 300 iterations for this design closure, as shown in Fig. 5. However, the Bayesian optimization approach can only close 95% of design goal even after 1000 iterations.

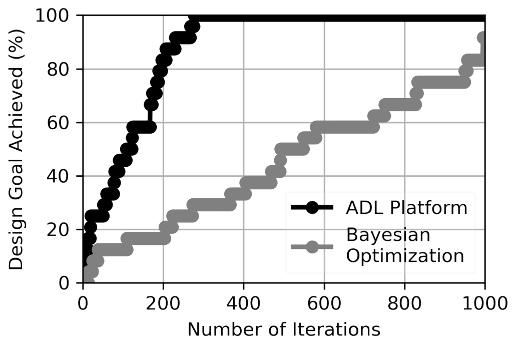


Fig. 5. Simulation iterations for the design closure of folded cascade operational amplifier using the ADL platform and Bayesian optimization.

It is evident from these two examples that ADL can perform design optimization of analog circuits with less number of simulation iterations when compared to other techniques such as Bayesian optimization. It has to be noted that the NNs can accurately capture the behavioral model of the analog circuit which can be carried forward to the top level hierarchy in the design flow leading to faster design closures.

5) Performance Improvement: The accelerated ability to optimize and synthesize the analog circuits as demonstrated earlier, also enables us to explore the performance limit of existing analog designs. As a test case, the integrated voltage regulator (IVR) for on-chip power management application as shown in Fig. 6 was used. Maximizing the conversion efficiency is the target, hence it is crucial to design the circuit so it can achieve its performance limit for a given technology; the IVR described is designed in 55-nm CMOS technology.

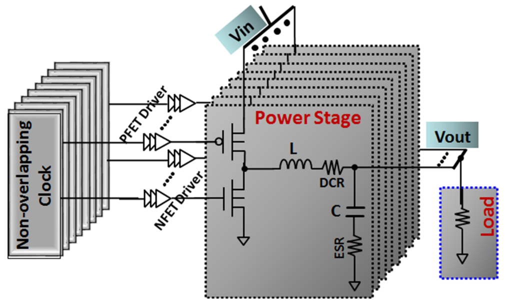


Fig. 6. System architecture of an 8-phase integrated voltage regulator (IVR).

To test the capability of the ADL platform for performance improvement, at first, 65 training samples that had a maximum efficiency of 79% from the workbench of an analog designer were used. Then, the ADL platform was invoked to perform design optimization. The maximization of efficiency was done in an iterative manner. In each successive iteration, the efficiency design goal was set higher than at the previous iteration, i.e. $\hat{Y}_C^{(i+1)} > \hat{Y}_C^{i}$. As can be seen in Fig. 7, the efficiency plateaus after 1000 iterations and the maximum efficiency achieved by the ADL platform is about 88% (9% higher than in the training samples).

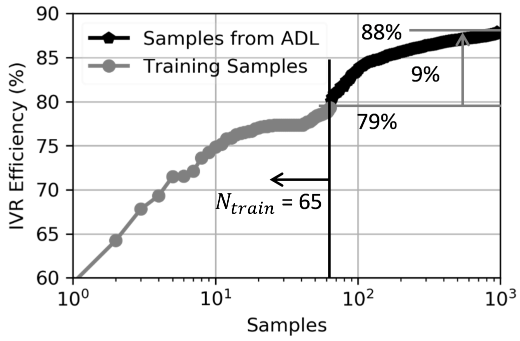


Fig. 7. Maximizing the efficiency of integrated voltage regulator in ADL platform.

In addition, the ADL platform yielded multiple solutions for the same maximum efficiency providing the designer with additional flexibility to choose based on other design considerations, such as area minimization and load current. In summary, the ADL platform can not only accelerate the analog design, but can also explore the performance limit of analog circuits for a given technology node.

IV. Conclusion

An automated deep learning (ADL) platform for accelerating analog design is presented, where the NN and EDA tools are integrated in a closed-loop to perform simulation verification for a new design closure. Analog design acceleration has been demonstrated with several functional analog circuits and benchmarked against other state-of-the-art methodology. The results indicate that the ADL platform can close multivariate analog design with a significantly reduced number of simulations. This work has laid the preliminary groundwork for IC design using ADL platform; in future it can be extended to other avenues such as placement and routing, high-speed channel design and SoC verification.

ACKNOWLEDGEMENT

This research is supported by the Agency for Science, Technology and Research (A*STAR) under Grant No. A20H6b0151 and A19E8b0102.

REFERENCES

[1] R. A. Rutenbar, G. G. E. Gielen, and J. Roychowdhury, "Hierarchical modeling, optimization, and synthesis for system-level analog and rf designs," *Proc. IEEE*, vol. 95, no. 3, pp. 640–669, 2007.

[2] M. Javaheripi, M. Samragh, and F. Koushanfar, "Peeking into the black box: A tutorial on automated design optimization and parameter search," *IEEE Solid-State Circuits Magazine*, vol. 11, no. 4, pp. 23–28, 2019.

[3] M. B. Yelten, T. Zhu, S. Koziel, P. D. Franzon, and M. B. Steer, "Demystifying surrogate modeling for circuits and systems," *IEEE Circuits and Systems Magazine*, vol. 12, no. 1, pp. 45–63, 2012.

[4] R. Dutta, et al., "Learning of multi-dimensional analog circuits through generative adversarial network (GAN)," in *32nd IEEE International System-on-Chip Conference (SOCC)*, 2019, pp. 394–399.

[5] H. Lin and P. Li, "Circuit performance classification with active learning guided sampling for support vector machines," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 9, pp. 1467–1480, 2015.

[6] F. Wang and X. Li, "Correlated bayesian model fusion: Efficient performance modeling of large-scale tunable analog/rf integrated circuits," in *53rd ACM/EDAC/IEEE Design Automation Conference (DAC)*, 2016, pp. 1–6.

[7] M. d. Hershenson, S. P. Boyd, and T. H. Lee, "Optimal design of a cmos op-amp via geometric programming," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 20, no. 1, pp. 1–21, 2001.

[8] Y. Wang, M. Orshansky, and C. Caramanis, "Enabling efficient analog synthesis by coupling sparse regression and polynomial optimization," in *51st ACM/EDAC/IEEE Design Automation Conference (DAC)*, 2014, pp. 1–6.

[9] R. Vural and T. Yildirim, "Analog circuit sizing via swarm intelligence," *AEU - International Journal of Electronics and Communications*, vol. 66, no. 9, pp. 732–740, 2012.

[10] W. Lyu, P. Xue, F. Yang, C. Yan, Z. Hong, X. Zeng, and D. Zhou, "An efficient bayesian optimization approach for automated optimization of analog circuits," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 65, no. 6, pp. 1954–1967, 2018.

[11] W. Lyu, F. Yang, C. Yan, D. Zhou, and X. Zeng, "Multi-objective bayesian optimization for analog/rf circuit synthesis," in *55th ACM/ESDA/IEEE Design Automation Conference (DAC)*, 2018, pp. 1–6.

[12] S. J. Park, B. Bae, J. Kim, and M. Swaminathan, "Application of machine learning for optimization of 3-d integrated circuits and systems," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, no. 6, pp. 1856–1865, 2017.

[13] H. Wang, J. Yang, H.-S. Lee, and S. Han, "Learning to design circuits," in *NeurIPS Workshop on Machine Learning for Systems*, 2018.

[14] H. Wang, J. Yang, L. Shen, N. Sun, H.-S. Lee, and S. Han, "Gcn-rl circuit designer: Transferable transistor sizing with graph neural networks and reinforcement learning," in *Design Automation Conference (DAC)*, 2020.

[15] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, *Learning Internal Representations by Error Propagation*. Cambridge, MA, USA: MIT Press, 1986, p. 318–362.

[16] R. Martinez-Cantin, "BayesOpt: A bayesian optimization library for nonlinear optimization, experimental design and bandits," *Journal of Machine Learning Research*, vol. 15, no. 115, pp. 3915–3919, 2014.