Inverse Sizing and Exploration Based on Diffusion Models and Structural Knowledge

***Abstract*—Quick and effortless exploration of design spaces is of major importance in the face of rapid development cycles, varying specifications and increasingly complex circuits. It is also well known, that the exploitation of slack in specifications by making adjustments to transistor device sizes can lead to improved yield. We address those challenges by combining two state of the art machine learning approaches to achieve accurate generation of optimal performance points, while also allowing for the possibility to explore nearby sizings that might lead to more robust designs. To this end, a diffusion model is combined with an algorithm, that analyzes the circuit and splits the problem into simpler subproblems. The models are tested on a set of typical operational amplifiers.**

I. Introduction

Since the development of SPICE, the industry accepted method for designing analog circuits has not changed much. While the SPICE models keep becoming more complex [1], the approach of manual device tuning persists [2]. Key challenges of making analog CAD tools useful in practice are robustness, ease of use and applicability to a high range of use cases. All those aspects are not easy to grasp on a fundamental level. On the other hand, machine learning approaches excel in modeling nonlinear black box phenomena. Their combination with already existing conventional techniques has a high potential to bridge remaining gaps, that prevent the adoption of tools in industry.

Analog design is inherently an inverse problem. While the calculation of performances, yield etc., is satisfactorily possible via simulation, the inverse, i.e. getting a circuit with optimal parametrization requires experience, trial and error and complex numerical routines. As shown in previous works [3], [4], machine learning (ML) methods can be used to accurately reproduce sizings on (pareto) optimal fronts. Furthermore, it has been shown [5] that diffusion models are well adaptable to the inverse sizing problem. Other ML based works attempted to solve this same problem. In [6], artificial neural networks (ANNs) are used to guess circuit sizings, utilizing only one ANN for modeling an entire netlist, leading to a higher learning task complexity and thus large sample sizes (in [6] 16600 or roughly $16 - 160\,x$ sample sizes used in this work) are required to learn the inverse problem within a smaller performance range. In [7], the problem is broken down into smaller tasks, and one ANN is trained to sequentially size a single device, in the same vein as [3], feeding the outputs of the previous ANNs to the input of the next, producing a cascade of networks that can be sorted in different ways, until an optimal order is found. The drawback is that for more complex circuits of e.g. 15 devices, optimally ordering the ANNs for sizing leads to a complexity of 15!.

Other works follow the traditional approach of tackling the direct sizing problem, treating it as an optimization problem [8]–[11]. Reinforcement learning was also studied, with [12]–[14] replacing the optimizer with an artificial neural network (ANN), leading to improvements in prediction speed, accuracy, and scalability. However, to the best of our knowledge, none of them show strengths in all four categories, speed, accuracy, sample efficiency and exploration.

In this work, we aim for a combination of all those virtues by applying denoising diffusion probabilistic models (DDPM) to the sizing of OpAmps, utilizing the approach in [3] of subdividing the circuits into simpler structures, which has shown high accuracy for low sample sizes. DDPMs have proven their potential for generating strongly varied, but realistic results in other fields, OpenAi's DALL-E and Stable Diffusion being among the most prominent examples. [5] proved these model's efficacy in the inverse sizing problem.

Unlike most machine learning models, DDPMs can suggest different sizings for the same problem by repeatedly sampling them. As shown in Section IV, this approach to the inverse problem can produce a more varied set of possible sizings, which is very helpful when exploring sizing options in the vicinity of performance optima. This work contributes a machine learning approach that:

- can suggest valid sizings with resulting performance in the vicinity of the required metrics, allowing for exploration of sizings while still fulfilling specifications
- is highly data efficient
- has an accurate expectation of the generated sizings
- generates sizing solutions at push-button speed

In section II, we provide an overview of the proposed method and its connection to structural analysis. In sec. **??**, we describe the sizing process in detail. We conclude with a presentation of results for two OpAmps.

II. From Structural Analysis to Circuit Sizing via Machine Learning

A. Basics of Structural and Functional Blocks

Our method heavily relies on decomposing circuits into subcircuits, which we call structural building blocks. Most building blocks are known to the analog designer by names such as *current mirror* or *differential pair*.

Parts of this work have been done within the collaborative project HoLoDEC funded by the Federal Ministry of Education and Research Germany (BMBF) under the funding code 16ME0705.

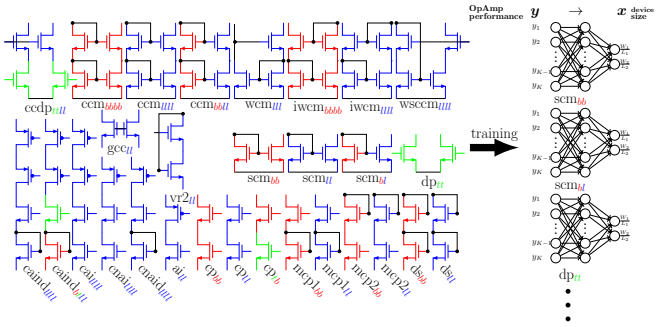


Fig. 1: Splitting various OpAmps into canonical building blocks with different functional characteristics and accumulating intrinsic design know-how by training each building block with random optimal samples from all available OpAmps. Indices b, l and t represent *bias*, *load* and *transconductance* functionality of devices. Every building block maps the overall OpAmp performance to its sizing.

We further distinguish different functional variants of one and the same structural block. To this end, we assign to every transistor in a circuit its functional property, out of three different possibilities: *bias, load* or *transconductance*. For example, a *simple current mirror* (scm) can act as *load*, but in other parts of a circuit, the same building block can act as *bias*. And sometimes, some part of a current mirror can act as *load*, while another part acts as *bias*. Figure 1 shows all structural blocks and their functional variants from our dataset. An example of decomposing a Folded-Cascode OpAmp into blocks can be seen in figure 2a. Its structural building blocks are shaded, the names annotated close by. Structural and functional block recognition is fully automated. The time required for this is much less than one second.

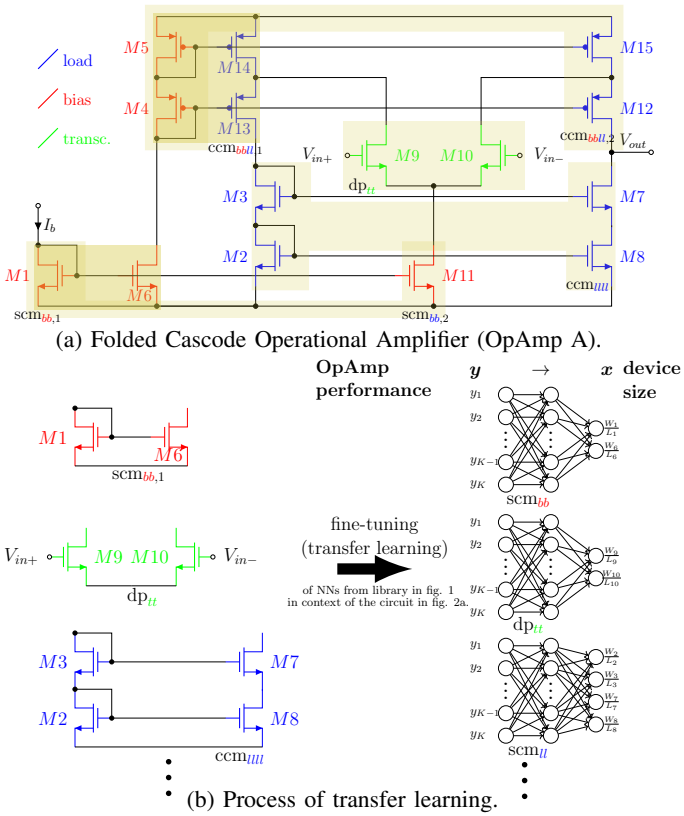
B. Overview of the Method

III. DDPM Background and Implementation

As mentioned in sec I, several DDPMs are trained to learn the sizing and respective performances distribution of the different structures that make up the opamp. DDPMs were first introduced in [15] as a new diffusion model parametrization, which are a class of NNs mostly used in an imaging context, like data generation and restauration. These models are composed of 3 different processes, which are briefly highlighted here:

1) Forward Process: First, the training data x_t is systematically destroyed by the addition of noise over T sequential timesteps. This noise is sampled from a Gaussian distribution and at the end of the T timesteps, the resulting training data distribution resembles itself a Gaussian distribution. This process is demonstrated by equation 1, where β_t is the variance scheduler, a hyperparameter that controls how much noise is added to the data x at each time timestep t.

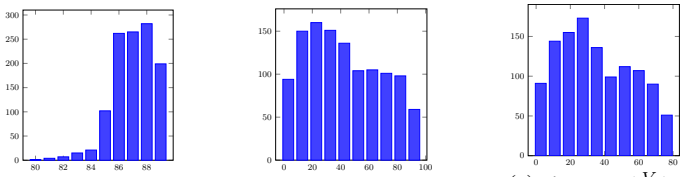
$$q(x_t|x_{t-1}) := \mathcal{N}(x_t; \mu, \Sigma) := \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t I) \tag{1}$$



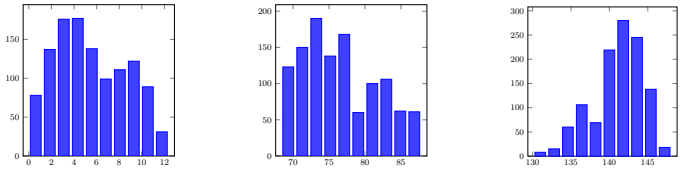
(a) Folded Cascode Operational Amplifier (OpAmp A).

(b) Process of transfer learning.

Fig. 2: Instead of training one huge network per OpAmp, we can split an Opamp, e.g. OpAmp A, into blocks (shading and coloring in fig. 2a) and use weights from our network-database as initial training points to merely fine-tune our networks (2b).



(a) gain (dB) (b) trns. fr. (Mhz) (c) slew rate ($\frac{V}{\mu s}$)



(d) power (mW) (e) phase margin (f) CMRR

Fig. 3: Histograms of some performance features for the Folded-Cascode OpAmp in Fig. 2a.

2) Reverse Process: After the data is destroyed at x_T, an ANN is trained to try to reverse the forward process, either by predicting the original data x_0 up front, or by predicting another value that can be used to reconstruct the original data, like the added noise, etc. This process is represented by equation 2, where $\alpha_t = 1 - \beta_t$ and $\overline{\alpha}_t = \prod_{s=1}^{t} \alpha_s$.

$$\rho_\theta(x_{t-1}|x_t) := \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t)) := \mathcal{N}(x_t; \frac{1}{\sqrt{\alpha_t}}(x_t - \frac{1-\alpha_t}{\sqrt{1-\overline{\alpha}} \tag{2}$$

The correlation between the forward and reverse process to

noise and denoise images can be seen in Fig 4.

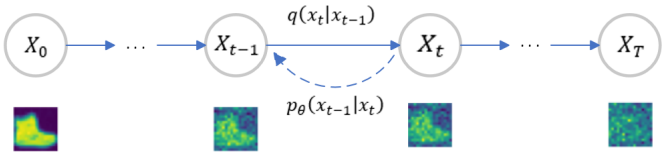


Fig. 4: Correlation between forward and reverse

3) Sampling Process: Finally, after the model is trained, the sampling process is used to generate new data. The model learned to reconstruct the original data, and so by giving it pure Gaussian noise it tries to reconstruct data. Except this time, there is no actual data in the first place, marking this a form of generative AI.

It was shown by [5] that DDPMs produce interesting results when applied to the inverse sizing problem. In this work, a similar implementation is considered, with a cosine schedule for β and classifier-free guidance [16], but with 2 key differences:

- instead of predicting the noise ϵ, the implemented model predicts a velocity equation v which has been shown to enable a true signal-to-noise ratio of 0 at the last timestep T, removing an important discrepancy between training and inference [17];
- the backbone of the model implemented here is a transformer, with an architecture similar to [18], as opposed to the simple multi-layer perceptrons implemented by [5]. To predict a more complex value in v a stronger architecture was required.

IV. Conclusion

References

[1] C. Gatermann and R. Sommer, "Teaching the mosfet: A circuit designer's view," in *2022 18th International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD)*, 2022, pp. 1–4.

[2] G. Gielen, "Analog synthesis 3.0: Ai/ml to synthesize and test analog ics: hope or hype ?" in *2023 ACM/IEEE 5th Workshop on Machine Learning for CAD (MLCAD)*, 2023, pp. 1–1.

[3] M. Leibl and H. Graeb, "Optimizer-free sizing of opamps leveraging structural and functional properties," in *2024 20th International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD)*, 2024, pp. 1–4.

[4] N. Lourenço, E. Afacan, R. Martins, F. Passos, A. Canelas, R. Póvoa, N. Horta, and G. Dundar, "Using polynomial regression and artificial neural networks for reusable analog ic sizing," in *2019 16th International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD)*, 2019, pp. 13–16.

[5] P. Eid, F. Azevedo, R. Martins, and N. Lourenço, "Solving the inverse problem of analog integrated circuit sizing with diffusion models," in *2024 20th International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD)*, 2024, pp. 1–4.

[6] N. Lourenço, J. Rosa, R. Martins, H. Aidos, A. Canelas, R. Póvoa, and N. Horta, "On the exploration of promising analog ic designs via artificial neural networks," in *2018 15th International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD)*, 2018, pp. 133–136.

[7] P.-O. Beaulieu, E. Dumesnil, F. Nabki, and M. Boukadoum, "Analog rf circuit sizing by a cascade of shallow neural networks," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pp. 1–1, 2023.

[8] M. Fayazi, M. T. Taba, E. Afshari, and R. Dreslinski, "Angel: Fully-automated analog circuit generator using a neural network assisted semi-supervised learning approach," *IEEE Transactions on Circuits and Systems I: Regular Papers*, pp. 1–14, 2023.

[9] A. F. Budak, D. Smart, B. Swahn, and D. Z. Pan, "Apostle: Asynchronously parallel optimization for sizing analog transistors using dnn learning," in *2023 28th Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2023, pp. 70–75.

[10] G. Wolfe and R. Vemuri, "Extraction and use of neural network models in automated synthesis of operational amplifiers," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 22, no. 2, pp. 198–212, 2003.

[11] K. Hakhamaneshi, M. Nassar, M. Phielipp, P. Abbeel, and V. Stojanovic, "Pretraining graph neural networks for few-shot analog circuit modeling and design," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 42, no. 7, pp. 2163–2173, 2023.

[12] J. Gao, W. Cao, and X. Zhang, "Rose: Robust analog circuit parameter optimization with sampling-efficient reinforcement learning," in *2023 60th ACM/IEEE Design Automation Conference (DAC)*, 2023, pp. 1–6.

[13] K. Settaluri, Z. Liu, R. Khurana, A. Mirhaj, R. Jain, and B. Nikolic, "Automated design of analog circuits using reinforcement learning," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 41, no. 9, pp. 2794–2807, 2022.

[14] J. Zhang, J. Bao, Z. Huang, X. Zeng, and Y. Lu, "Automated design of complex analog circuits with multiagent based reinforcement learning," in *2023 60th ACM/IEEE Design Automation Conference (DAC)*, 2023, pp. 1–6.

[15] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," 2020. [Online]. Available: https://arxiv.org/abs/2006.11239

[16] J. Ho and T. Salimans, "Classifier-free diffusion guidance," 2022. [Online]. Available: https://arxiv.org/abs/2207.12598

[17] S. Lin, B. Liu, J. Li, and X. Yang, "Common diffusion noise schedules and sample steps are flawed," 2024. [Online]. Available: https://arxiv.org/abs/2305.08891

[18] W. Peebles and S. Xie, "Scalable diffusion models with transformers," 2023. [Online]. Available: https://arxiv.org/abs/2212.09748