AnGeL: Fully-Automated Analog Circuit Generator Using a Neural Network Assisted Semi-supervised Learning Approach

Morteza Fayazi, *Graduate Student Member, IEEE,* Morteza Tavakoli Taba, *Graduate Student Member, IEEE,* Ehsan Afshari, *Senior Member, IEEE,* Ronald Dreslinski, *Senior Member, IEEE*

***Abstract*—Machine Learning (ML) has shown promising results in predicting the behavior of analog circuits. However, in order to completely cover the design space for today's complicated circuits, supervised ML requires a large number of labeled samples which is time-consuming to provide. Furthermore, a separate dataset must be collected for each circuit topology making all other previously gathered datasets useless. In this paper, we first present a database including labeled and unlabeled data. We use neural networks to determine the behavior of complicated topologies by combining the more simple ones. By generating such unlabeled data, the time for providing the training set is significantly reduced compared to the conventional approaches. Using this database, we propose a fully-automated analog circuit generator framework, AnGeL. AnGeL performs all the schematic circuit design steps from deciding the circuit topology to determining the circuit parameters *i.e.* sizing. Our results show that for multiple circuit topologies, in comparison to the state-of-the-art works while maintaining the same accuracy, the required labeled data is reduced by 4.7x - 1090x. Also, the runtime of AnGeL is 2.9x - 75x faster.**

***Index Terms*—Analog circuit design automation, topology selection, circuit sizing, neural network, semi-supervised learning.**

I. Introduction

THE existence of various design parameters and specifications in present-day complex circuits, in addition to severe process variations, have made the manual circuit design procedure challenging, time-consuming, and inefficient [1]. All these challenges make the automated analog circuit generation a necessity. Schematic circuit design includes two main steps: deciding the topology and determining the value of the circuit elements (*i.e.* sizing) to meet the desired specifications.

Model-based approaches are one of the main techniques in the automated sizing of circuits *e.g.* non-convex polynomial optimization, geometric programming, and Neural Network (NN) [2], [3]. In such approaches, a global model is built based on the collected training set, which makes the model reusable for other data. However, to maintain high accuracy while covering the whole design space, a large labeled training set is required [4]. SPICE simulation is used for gathering such a large set which is time-consuming. To make matters worse, a separate dataset is required for each circuit topology, even if a single element is added or removed.

All authors are with the Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI, 48109 USA (e-mail: fayazi@umich.edu, tmorteza@umich.edu, afshari@umich.edu, rdreslin@umich.edu).

Even though the model-based approaches are reusable, their accuracies are not usually high due to the large number of design parameters and nonlinearity of object and constraint functions [5]. On the other hand, the other approaches in automated sizing, simulation-based algorithms [5]–[8], optimize circuits directly by the gathered simulated data and usually have higher accuracy. However, they are more time-consuming in comparison with model-based methods and are non-reusable. Here, non-reusable means that even if the target specifications slightly changes, the whole process needs to reoccur. The third approach in circuit sizing is a hybrid of model- and simulation-based methods. In such approaches, after building the initial model, the model is gradually updated by running simulations during the optimization procedure instead of using an offline model.

In order to reduce the size of the labeled dataset, we propose a co-learning-based NN approach. The term co-learning here means passing the knowledge from usually less complex models to more complicated ones to reduce the training cost [9]. In our case, the previously gathered datasets of simpler circuits are leveraged to shrink the required labeled training sets for more complicated circuit topologies. In other words, we use NNs to determine the circuit behavior of complicated topologies by combining the simpler ones.

Using the presented database, we propose a fully-automated analog circuit generator framework, AnGeL. The goals of AnGeL are threefold: (a) achieve a reusable, accurate, and fast model to meet the given specifications of the overall circuit, (b) reduce the number of required labeled training samples, and (c) perform all schematic circuit design steps such as deciding the overall circuit topology, selecting the topology of sub-circuits, and sizing them. Since we are using both labeled and unlabeled data in the database, our approach is classified as semi-supervised learning.

In order to achieve a reusable, accurate, and fast model, AnGeL uses a hybrid (model-simulation-based) approach for sizing. Between model-based approaches, NNs have shown promising results in all circuit design levels such as single-board computer, sizing, layout, post-layout simulation, and system-on-chip design [10]–[14]. Also, once the model is trained, it is able to execute a large input set in a very short time. In our approach, we use an NN to estimate the functionality of circuits such that it determines the performance of interest (circuit's specifications) when the circuit parameters are given. We use such an NN as the global optimization

engine to determine the circuit parameters to meet the desired specifications. To improve the accuracy, we implement Particle Swarm Optimization (PSO), which is a simulation-based algorithm while the trained NN is invoked instead of the SPICE simulator to provide the necessary data. Indeed, PSO is a population-based approach that has a better precision compared to other methods *e.g.* genetic algorithms [7]. In this regard, our proposed approach is accurate because of implementing PSO in the local optimization as well as reusable and fast due to using an NN as the functional estimator of the circuit.

In addition to using an efficient database, in order to further reduce the size of the training data, AnGeL divides the overall circuit into multiple sub-circuits and analyzes each individually while meeting the given specifications of the overall circuit. This approach allows dealing with circuits with less design parameters. Therefore, a smaller training set is required to analyze them while maintaining equivalent accuracy in comparison with analyzing the overall circuit [15]. Moreover, the global and local optimum points are found faster since design spaces of sub-circuits are smaller. Furthermore, sub-circuits are analyzed in parallel which reduces the runtime. Another advantage of this circuit division is supporting many topologies while AnGeL is trained on a few of them. As an example, when AnGeL trains on 12 single-stage operational amplifier (OPAMP) topologies, it supports 12×12=144 two stages topologies. To this end, since the given specifications are for the overall circuit, AnGeL breaks down them to appropriate sub-circuits specifications while considering the effect of other sub-circuits *e.g.* loading effect. For further division, even a sub-circuit may be broken into smaller chunks. For instance, a two-stage OPAMP may consist of two differential stages where each stage may consist of a gain block, current source, and DC-biasing chunks. So, AnGeL hierarchically determines the specifications of each sub-circuit regarding the specifications of the parent circuit. This hierarchical approach enables AnGeL to design circuits with many design parameters ($\sim$40) in a short time and with high accuracy.

AnGeL also selects the most suitable high-level circuit topology (number of stages, type of sub-circuits *e.g.* gain block, DC-biasing) between available candidates as well as determining the target specifications of each sub-circuit. In a similar way, the topology of each sub-circuit is selected among the available candidates. The topology of sub-circuits as well as their sizing are determined based on the sub-circuit specifications.

To validate our proposed method, all the design steps of one-, two-, and three-stage OPAMPs as well as filters have been demonstrated. The performance of the synthesized circuits by AnGeL is validated by SPICE simulations. Our results show that for multiple circuit topologies, in comparison to the state-of-the-art works while maintaining the same accuracy, the required labeled data is reduced by 4.7x - 1090x. Also, the runtime of AnGeL is 2.9x - 75x faster. Moreover, testing on more than 1,450 different circuits illustrates that the performance of circuits that are determined using combining simpler topologies has an average percentage error of less than 0.043.

The main contributions of this paper can be summarized as follows:

- Creating a database using a co-learning-based NN model that significantly reduces the size of the required labeled training set compared to the state-of-the-art works.
- An efficient and intelligent fully-automated analog circuit generator framework that uses the created database and designs circuits from deciding the topology to sizing.
- Achieving a fast and accurate method using a hybrid of model- and simulation-based approaches.
- Designing circuits with many design parameters ($\sim$40) using a hierarchical approach.
- Supporting many circuit topologies while few topologies are used in the training by leveraging the circuit division into sub-circuits.

II. Background and Related Work

A. Problem Formulation

Estimating the functionality of circuits and optimizing them are two important areas in automating the design of analog circuits [16]. In estimating the functionality of circuits, the main goal is to find f as a function of circuit parameters, $\mathbf{x}$, to approximate the performance of interest, y. DC bias voltages and size of transistors (W, L) are examples of circuit parameters, and the voltage gain of an OPAMP is an example of the performance of interest: $y \approx f(\mathbf{x})$.

The goal of analog circuit optimization is to determine the design parameters such that

$$\begin{aligned} &\text{minimize } f_1(\mathbf{x}), \ldots, f_M(\mathbf{x}) \\ &\text{subject to: } c_1(\mathbf{x}) < 0, \ldots, c_N(\mathbf{x}) < 0, \end{aligned} \tag{1}$$

where $f_1, \ldots, f_M$ are the figure of merit of the circuit [17]. $c_1, \ldots, c_N$ are constraints such as $x_j \in [p_j^-, p_j^+]$ or bandwidth (BW) $>$ 1GHz.

B. Reducing Database Size

In estimating the functionality of circuits, decreasing the number of required labeled data is crucial. There are several works that have used the concept of co-learning for this purpose [9], [15], [18]. The main idea of them is to find two representations for modeling a circuit's specification where one of them is simpler and less accurate than the other. By passing the information from the simpler model to the more complicated one, they achieve an accurate model with fewer labeled data. Hassanpourghadi *et al.* [4] and Alawieh *et al.* [15] have divided the overall circuit into multiple sub-circuits to conclude simpler models, and as a result, the size of the training set is decreased. Even though the aforementioned approaches have helped reduce the number of labeled data, a completely new dataset is required, even with a small modification in the topology. Wang *et al.* [19] have mentioned the idea of transferring knowledge between topologies using graph neural networks. However, this transfer is applied from a two-stage to a three-stage OPAMP, not among single sub-circuits.

C. Circuit Design Framework

There are several automated circuit design frameworks that both create a topology and size the circuits [20]–[23]. Since they are mainly simulation-based approaches, they are slow and have relatively high computation costs due to the need of running a large number of simulations. Moreover, they are non-reusable.

Several studies have focused on the topology construction [24]–[26] as their main idea. They either use an external tool for sizing [24] or perform an initial sizing [25]. Topology construction is categorized into two types in general: (a) topology selection from a predefined library and (b) topology generation from basic building blocks [24], [26]. Generating topologies from basic building blocks is more creative as it may lead to creating circuit topologies that have not been seen before. Furthermore, traditional topology selectors support fewer topologies [25]. FEATS [24] represents circuits with a graph and uses abstract building blocks to generate circuit topologies. Even though it supports more than 10,000 topologies, it takes hours for it to synthesize. FUBOCO [25] performs hierarchical design using functional blocks. Although it is faster than the previous approaches [24], it takes it still several hours to design as it consequently designs stages and sizes many topologies before concluding. On the other hand, in our approach, using the idea of division and hierarchical parallel design, we are able to support more than 10,000 topologies while spending a few minutes to conclude the design. Indeed, our work is positioned between the traditional topology selector approaches and basic-building-blocks-based approaches since AnGeL still selects the sub-circuit topologies but works with them as basic blocks in the top-level topology.

Many studies have assumed the circuit topology is given as an input [16], and they just have focused on sizing. NNs have a fast runtime and have a high degree of freedom due to their multiple hidden layers [4]. Several studies use fully-connected-NNs for sizing of analog circuits [27]–[29]. Their main disadvantages are requiring a large labeled training set which is time-consuming to provide, as well as inaccuracy. Liu *et al.* [30] use transfer learning to decrease the size of the post-layout training set by leveraging the schematic gathered data. Recently, reinforcement learning, graph neural network, and constraint programming [31]–[33] also have been used in analog circuit sizing. The most common method to automate circuit sizing is simulation-based approaches [34]. Bayesian optimization is one of the most popular techniques in this regard [5], [35]. However, because of the inherent expensive computation cost of simulation-based approaches, Zhang *et al.* [36] try to extract the "good" features and define the Gaussian Process based on them using NNs. PSO is another popular technique in circuit sizing [7], [37]–[39] however, the runtime is long. To alleviate these drawbacks, a hybrid model-simulation-based approach is implemented by replacing the slow SPICE simulation with a fast circuit functional model, which usually is an NN [40], [41]. Supporting only one topology, having a narrow design space, and requiring a large training set are their main shortcomings.

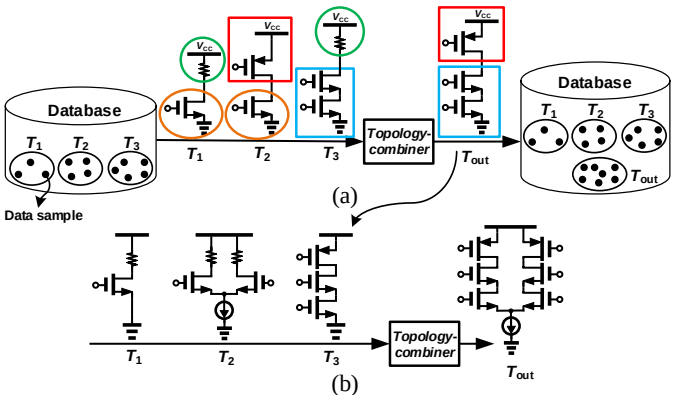

Fig. 1. (a) An example of three simpler topologies, T_1, T_2, T_3, that can be combined to determine the circuit behavior of T_{out}. To enhance visibility, we use identical colored boxes/circles to show bodies/loads that are the same in different topologies. (b) The output topology of Fig. 1(a) is used to make a more complicated topology.

III. Proposed Database Generation

A. Overview

Collecting datasets for simpler topologies is less expensive since they have smaller design spaces. On the other hand, in order to completely cover the design space for complex circuits with many design parameters (*e.g.* size of transistors), an abundant number of samples are needed. As an example, T_{out} in Fig. 1(b) has 21 design parameters *i.e.* width, length, v_{bias} for 6 transistors and a current source. Therefore, gathering only labeled data by running simulations would be computationally expensive.

Fig. 1(a) demonstrates the proposed co-learning-based database generation flow. The goal of the topology-combiner is to determine the circuit behavior (*i.e.* the effect of design parameters on output specifications) of complicated topologies using the simpler ones in order to reduce the size of the required labeled data. The datasets for more basic, simple topologies, *e.g.* T_1, T_2, T_3, are gathered using simulations. Such datasets contain the value of the specification associated with different circuit design parameters for each topology. Then, the circuit behavior of the more complicated topology (T_{out}) is determined using combinations of the simpler ones. For example, in Fig. 1(a), T_{out} is built by replacing the resistive load of T_3 with the PMOS load of T_2. Moreover, T_1 has the same body and resistive load as T_2 and T_3, respectively. So, intuitively, the behavior of T_{out} can be modeled by leveraging the previously gathered T_1, T_2, T_3 datasets information. Then, a large pseudo sample set that covers the design space of T_{out} is generated with almost zero cost using such a model (unlabeled data). So, our database is a combination of labeled and unlabeled data. Now that the dataset of T_{out} is generated, it can be used to build more complicated topologies. As an example, the output topology in Fig. 1(a) is used to make a more complicated topology in Fig. 1(b).

B. Topology Combiner Model

Fig. 2(a) demonstrates the general structure of three simpler topologies (T_1, T_2, and T_3) that can be combined to determine

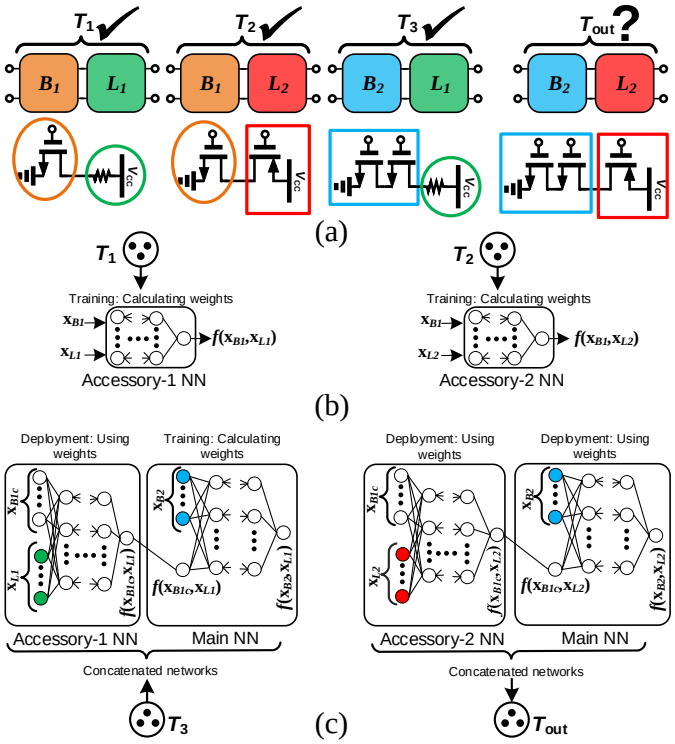

Fig. 2. The topology-combiner model. (a) The general structures of three simpler topologies, T_1, T_2, and T_3, which are combined to determine the circuit behavior of T_{out} along with the example of Fig. 1(a). We use the same boxes/circles with the same color for showing bodies/loads that are the same to make them more visible in different topologies. (b) Our proposed co-learning-based NN models: Training of accessory-1 and accessory-2 models using T_1, and T_2 dataset, respectively. (c) Our proposed co-learning-based NN models: Training and deployment of the main NN using T_3 dataset.

the circuit behavior of T_{out}. Each topology is composed of two parts: B (body) and L (load). T_1 and T_2 have the same body (B_1) and T_1, T_3 have the same load (L_1). Moreover, T_{out} has the same body (B_2) and load (L_2) as T_3 and T_2, respectively. We use the notation of $\mathbf{x}_B$ and $\mathbf{x}_L$ for showing the design parameters of body and load, respectively, and $f(\mathbf{x}_B, \mathbf{x}_L)$ for denoting a circuit specification when the body is B, and load is L. This f can be any of circuit specifications *e.g.* gain, BW, etc. The training process needs to be done individually for each circuit specification.

Our goal is to determine the mapping from $\mathbf{x}_{B_2}, \mathbf{x}_{L_2}$, *i.e.* the design parameters of T_{out} topology, to $f(\mathbf{x}_{B_2}, \mathbf{x}_{L_2})$ without having any direct mapping samples. That is, to determine the mapping $\xi_{T_{out}}$:

$$\xi_{T_{out}} : (\mathbf{x}_{B_2}, \mathbf{x}_{L_2}) \mapsto f(\mathbf{x}_{B_2}, \mathbf{x}_{L_2}) \tag{2}$$

Fig. 2(b) and (c) show the NN implementation for our proposed topology-combiner and how the dataset of each of T_1, T_2, and T_3 is used in the training of such NNs. The first step is to train the accessory-1 model using the training set of T_1. As it is depicted in Fig. 2(b) on the left side, inputs are the design parameters of T_1, and the output is the desired specification in T_1, *i.e.* $f(\mathbf{x}_{B_1}, \mathbf{x}_{L_1})$. Similarly, the accessory-2 model is trained by leveraging the T_2 dataset while inputs and outputs are the design parameters and the desired specification of T_2, respectively. The accessory-2 NN is shown in Fig. 2(b) on the right side.

Next, the accessory-1 and the main NNs are concatenated as shown in Fig. 2(c) on the left side. Note that the accessory-1 NN is used in the deployment phase now. By giving a fixed, constant $\mathbf{x}_{B_1}$ which we denote with $\mathbf{x}_{B_{1c}}$, the inputs variable of this concatenated network would be $\mathbf{x}_{B_2}$ and $\mathbf{x}_{L_1}$ which are the design parameters of T_3. As it is shown, the output is the desired specification in T_3, *i.e.* $f(\mathbf{x}_{B_2}, \mathbf{x}_{L_1})$. Since the accessory-1 NN is already trained, by training this concatenated network, the weights of the main NN are calculated. In other words, we will have:

$$\xi_{Main} : (\mathbf{x}_{B_2}, f(\mathbf{x}_{B_{1c}}, \mathbf{x}_{L_1})) \mapsto f(\mathbf{x}_{B_2}, \mathbf{x}_{L_1}). \tag{3}$$

The inputs of the main NN are $\mathbf{x}_{B_2}$, and $f(\mathbf{x}_{B_{1c}}, \mathbf{x}_{L_1})$ while the output is the desired specification of the topology with B_2 body and L_1 load. Therefore, during the deployment phase of the main NN, by replacing L_1 with L_2 (giving $f(\mathbf{x}_{B_{1c}}, \mathbf{x}_{L_2})$ instead of $f(\mathbf{x}_{B_{1c}}, \mathbf{x}_{L_1})$), we will have the desired specification of the topology with B_2 body and L_2 load, which is our goal, T_{out}. This means we will have:

$$\xi'_{Main} : (\mathbf{x}_{B_2}, f(\mathbf{x}_{B_{1c}}, \mathbf{x}_{L_2})) \mapsto f(\mathbf{x}_{B_2}, \mathbf{x}_{L_2}), \tag{4}$$

which has been derived by replacing $\mathbf{x}_{L_1}$ with $\mathbf{x}_{L_2}$ in (3). This is perfectly aligned with our goal in (2) as if we replace $\mathbf{x}_{L_2}$ with the corresponding $f(\mathbf{x}_{B_{1c}}, \mathbf{x}_{L_2})$ in (2) we will conclude the same equation as (4). So, the last step is to map $\mathbf{x}_{L_2}$ to the corresponding $f(\mathbf{x}_{B_{1c}}, \mathbf{x}_{L_2})$ which is achieved by using the trained accessory-2 NN. So, during the deployment phase of accessory-2 NN, by giving the $\mathbf{x}_{B_{1c}}$ as the body, we would have the mapping of $\mathbf{x}_{L_2}$ to $f(\mathbf{x}_{B_{1c}}, \mathbf{x}_{L_2})$. This procedure is illustrated in Fig. 2(c) on the right side. Hence, to determine the behavior of T_{out} from its design parameters, we first find $f(\mathbf{x}_{B_{1c}}, \mathbf{x}_{L_2})$ leveraging the accessory-2 NN. Next, we feed the result to the main NN along with the design parameters of B_2.

To give more intuition about our models, it should be noted that the key is to have $f(\mathbf{x}_{B_1}, \mathbf{x}_{L_1})$ in the main NN instead of directly depending on the design parameters of the load. Indeed, $f(\mathbf{x}_{B_1}, \mathbf{x}_{L_1})$ abstracts the "effect" of a load in a topology with the B_1 body as single input for the main NN. This is the reason that we are able to replace L_1 with L_2 in the deployment phase of the main NN. Moreover, the other inputs of the main NN are the design parameters of B_2 which is the body of T_{out} too so, it works exactly as we want.

IV. Proposed AnGeL Framework

Using the presented database, we create a fully-automated analog circuit generator framework, AnGeL.

A. Overview

As it was mentioned earlier, AnGeL divides the overall circuit into multiple sub-circuits and analyzes each individually. This results in dealing with smaller circuits and hence, requiring a smaller training set size while keeping the same accuracy in comparison with analyzing the overall circuit. The

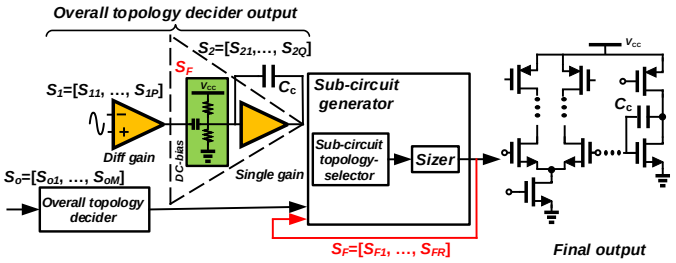

Fig. 3. High-level platform architecture of AnGeL.
S_o: Desired specifications for the overall circuit given by users.
S_1, S_2: The specifications for each sub-circuit that are determined by the overall topology decider module.
S_F: Specifications of sub-circuits that are determined by the bigger sub-circuits that they are part of.

other advantages of dividing the circuit are finding global and local optimum points faster since design spaces of sub-circuits are smaller, faster runtime due to analyzing sub-circuits in parallel, and supporting many topologies while only a few of them are used in the training set.

Fig. 3 shows a high-level platform architecture of the proposed circuit design flow of AnGeL. The desired specifications, *i.e.* $S_o = [S_{o1}, \ldots, S_{oM}]$, are given to the overall topology decider module. Here, we assume the given specifications are all in equality format *e.g.* $S_{o1} = C_1, S_{o2} = C_2, \ldots$. The goal of AnGeL is to design a circuit at the transistor level so that its output specifications are as close as possible to the given ones. Transforming constraints in inequality format *e.g.* $S_{o1} > C_1, S_{o1} < C_2, \ldots$ along with objective functions *e.g.* minimize S_{oM} are explained in Section IV-E.

The overall topology decider module determines the high-level topology of the circuit accordingly. Indeed, this module determines the number of stages, type of each sub-circuit (*e.g.* gain block, DC-biasing), how the sub-circuits are connected to each other (*e.g.* having feedback, being in parallel or series), and having single-ended or differential input/output ports for sub-circuits. Moreover, the overall topology decider module determines the specifications for each sub-circuit (*i.e.* $S_1, S_2, \ldots$). The determined specifications for each sub-circuit is passed to the sub-circuit generator module to build a circuit at the transistor level.

The feedback loop around the sub-circuit generator module shows that the specifications of some sub-circuits (*i.e.* S_F) are determined by the bigger sub-circuits that they are part of. For instance, the specifications of the DC-bias sub-circuit cannot be determined before determining the input DC voltage of the following gain stage. In this regard, AnGeL hierarchically determines the specifications of each sub-circuit regarding the specifications of the parent circuit. This hierarchical approach and circuit division enables AnGeL to design circuits with many design parameters ($\sim$40) in a short time and with high accuracy. For example, a band-pass active filter consists of low-pass and high-pass filters that each may include a two-stage OPAMP. All these sub-circuits are designed in a hierarchical approach and in parallel with each other.

The first step in the proposed sub-circuit generation flow is to determine the topology. The topology is selected among the available topologies in the database. Based on the selected topology, AnGeL determines the value of design parameters (sizing the sub-circuit). There is a separate model for sizing each topology. To apply AnGeL to a new technology node, the training process needs to be redone. In order to decrease the number of such new training samples, the idea of transfer learning [42] can be used which is out of the scope of this paper.

TABLE I
COMPARISON COST OF ANGEL AND THE CONVENTIONAL APPROACHES REGARDING THE CURSE OF DIMENSIONALITY. n_1 AND n_2: NUMBER OF DESIGN PARAMETERS IN TWO DIFFERENT SUB-CIRCUITS. k: NUMBER OF SAMPLES FOR EACH PARAMETER.

$[n_1, n_2, k]$	[5, 5, 3]	[5, 5, 5]	[5, 7, 4]	[7, 6, 4]	[7, 7, 5]
Conventional cost	1,000	100,000	20,736	28,561	537,824
AnGeL cost	250	6,250	3,026	3,697	33,614

Using the topology combiner and the overall topology decider module, we do not majorly face the curse of dimensionality. The reason is that we are able to determine the behavior of complicated topologies that require more training sets (by leveraging simpler topologies with almost zero cost) and efficiently break down multi-stage circuits into multiple single stages. However, the nonideality of the topology combiner may slightly increase inaccuracy in the whole design process. It should be noted that, as mentioned in Section V, the topology combiner has a very high accuracy, but this slight nonideality is in the trade-off with the curse of dimensionality. For a better cost comparison between AnGeL and the conventional approaches, we denote the number of design parameters in two different sub-circuits with n_1 and n_2, respectively. Assuming we take k samples for each parameter, the cost of AnGeL for analyzing a two-stage circuit with those two sub-circuits is ${n_1}^k + {n_2}^k$, considering only the effect of breaking down multi-stage circuits into multiple single stages. However, this cost for the conventional approach is $(n_1 + n_2)^k$. As summarized in Table I, the cost of AnGeL is significantly lower than the conventional approaches for different typical n_1, n_2, and k values.

B. Overall Topology Decider

The main goal of the overall topology decider module is to determine the high-level topology of the circuit and break down the overall circuit specifications into usable specifications for sub-circuits. Determining the sub-circuit specifications via overall specifications is challenging. The assigned sub-circuits' specifications need to meet the overall specification when they are assembled together as the associated high-level topology. This requires a proper training set that teaches the module how to assign sub-circuits' specifications such that the net impact of them in the associated high-level topology meets the overall specification. Furthermore, there are multiple ways for breaking down each overall specification, but many of them are not feasible for sub-circuits considering other specifications. For instance, assigning a large gain and BW to a sub-circuit may not be feasible simultaneously. Therefore,

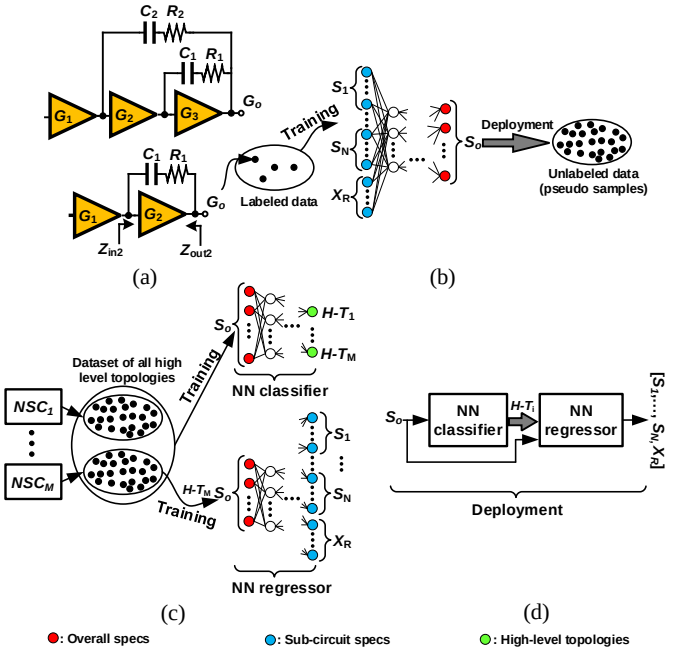

Fig. 4. Overall topology decider flow. (a) Two examples of high-level topologies. (b) Net-specification-calculator model. (c) Overall-specification-breaker model in the training phase. (d) Overall-specification-breaker model in the deployment phase.
S_o: The vector of overall circuit specifications.
$[S_1, \ldots, S_N]$: The associated specifications sets of N sub-circuits.
X_R: A vector of design parameters that are not included in any sub-circuits.
$[NSC_1, \ldots, NSC_M]$: The associated net specification calculator models of M high-level topologies.
$[H\text{-}T_1, \ldots, H\text{-}T_M]$: M high-level topologies.

a large, proper training set is needed for learning all these relationships.

To overcome the aforementioned challenges we implement two main models: 1) net-specification-calculator model for calculating the net impact of different sub-circuits' specifications at the module-level on the overall circuit; 2) overall-specification-breaker model which uses a large dataset generated by the net-specification-calculator model for properly breaking down the overall specifications into sub-circuits specifications as well as determining the high-level topology. The overall-specification-breaker model consists of two NN models: A) a classifier for selecting the most suitable high-level circuit topology; B) a regressor for determining the target specifications of each sub-circuit.

Fig. 4(a) illustrates two examples of high-level circuit topologies. According to Fig. 4, the overall specifications (S_o) is a function of sub-circuits' specifications ($S_1, \ldots, S_N$), and other circuit's design parameters that are not included in any sub-circuits (X_R). For example, the overall DC gain (G_o) is a function of sub-circuits' DC gain (G_1, G_2, G_3) and input/output impedance of each sub-circuit (Z_{in}, Z_{out}). It should be mentioned that each of S_i $\forall i \in \{1, \ldots, N\}$ is a vector of specifications. So, to calculate the overall gain, we have the input/output resistance of each sub-circuit in addition to their DC gains, as an example. R_1, R_2, C_1, C_2 are examples of design parameters that are not included in any sub-circuits (*i.e.* X_R) in Fig. 4(a) that are used for calculating phase margin in amplifiers for instance. By taking the input/output impedance of sub-circuits as well as X_R into account, the loading effects are properly considered in our model.

Fig. 4(b) depicts the NN implementation of the net-specification-calculator model. During training, a labeled dataset is used for learning the net impact of different sub-circuits specifications at the module-level. A separate regression NN model is used for learning such net impacts on each high-level topology. Once the net-specification-calculator models are trained, in the deployment phase, a large set of different unlabeled data $(S_1, \ldots, S_N, X_R)$ are fed to them to generate pseudo samples ($\{S_1, \ldots, S_N, X_R\}$, S_o). Indeed, the output is the overall circuit specifications, S_o, associated with each unlabeled input in different high-level topologies. Generating such a large set is with almost zero cost since the net specification calculator is already trained. This large unlabeled dataset is used for training another model, overall specification breaker. This unlabeled dataset addresses both challenges of being large and being able to teach how to assign sub-circuits' specifications that the net impact of them meet the overall specifications. The reason is that for all of the large dataset samples ($\{S_1, \ldots, S_N, X_R\}$, S_o), the overall specification equals the net impact of sub-circuits' specifications by definition.

The overall specification breaker learns how to properly break down the overall specifications into sub-circuits' specifications as well as how to determine the high-level topology. Fig. 4(c) demonstrates the overall-specification breaker-model in training phase. During training, using a dataset of all high-level topologies populated by the net-specification-calculator model, an NN classifier [43] learns how to select the most suitable high-level circuit topology between available candidates regarding the overall specifications. Moreover, a separate NN regressor is implemented to break down the overall specifications into sub-circuits' specifications for each high-level topology using the large provided training set. This large training set works as a lookup table for the regressor model. As shown in Fig. 4(d), during deployment, first, the classifier selects the high-level topology. Then, based on such a high-level topology, the regressor model breaks down the overall specifications into sub-circuits' specifications. It should be mentioned that with increasing the number sub-circuits or elements that are not included in sub-circuits (X_R), both the net-specification-calculator and overall-specification-breaker models become more complicated. However, our approach in such cases still outperforms the conventional methods which do not have the overall topology decider module as the conventional models become much more complicated than ours.

To summarize, these are the overall topology decider steps and models:

- The net-specification-calculator model calculates the net impact of different sub-circuits' specifications at the module-level on the overall circuit.
- In the deployment phase, for each overall circuit topology, the net-specification-calculator model generates a large set of overall specs by getting sub-circuit specs as inputs.
- Using the generated datasets by the net-specification-calculator models as the training set, the overall-

specification-breaker model breaks down the overall specifications into sub-circuits specifications and determines the high-level topology as well.
- The overall-specification-breaker model consists of two NN models:
  1) A classifier for selecting the most suitable high-level circuit topology between available candidates.
  2) A regressor for determining the target specifications of each sub-circuit.
- With taking input and output impedances of sub-circuits as well as X_R into account, the loading effects are properly considered in our model.

C. Sub-circuit Topology-Selector

In order to decide the transistor level topology for each sub-circuit, we input the target specifications of the sub-circuit to a classifier model, and it selects the most suitable topology. There are multiple candidates in the database where the classifier decides which one is the most appropriate for the given specifications. To this end, three different classification models *i.e.* random forest, Support Vector Machine (SVM), and NN classifier [43]–[45] are analyzed and the one which gives the highest accuracy is selected as the sub-circuit topology-selector model. Moreover, the SVM is analyzed using four different kernels *i.e.* linear, polynomial, sigmoid, and Radial Basis Function (RBF). During the training, the circuit parameters and the associated topology are given as input and output, respectively.

More complicated topologies in the database have more instances since their design spaces are larger, which causes imbalanced data. To solve this, a subset of the database is considered for training that has the same number of samples for all topologies. This method is called down-sampling [46].

D. Sub-circuit Sizing

When the topology of sub-circuits is decided, a proper sizing is needed to determine the design parameters value *e.g.* size of transistors and value of voltages. The goal of sizing is to determine the design parameters such that:

$$\begin{aligned} &\text{minimize } \sum_i \omega_i |\frac{S_{si} - S^*_{si}}{S_{si}}|, \\ &\text{subject to: } |\frac{S_{si} - S^*_{si}}{S_{si}}| < \varepsilon_i, \end{aligned} \tag{5}$$

where S_{si} are the desired specifications and S^*_{si} are the determined specifications by the sub-circuit sizing module. ω_i are the weights that are used to prioritize specifications that are more important for users. $|\frac{S_{si}-S^*_{si}}{S_{si}}| < \varepsilon_i$ in Equation (5), ensures each output specification is as close as possible to the associated desired specification.

In order to minimize Equation (5), we implement global and local optimization engines. For this purpose, first, we train a separate regression NN for each topology to estimate the functionality of sub-circuits when the circuit parameters are given. These NNs work as a fast circuit simulator instead of invoking time-consuming SPICE. We have used the idea of multiple starting points [47] in our optimization, which means we apply the local optimization on the multiple designs that have resulted in the minimum of Equation (5) in the global phase. The final result is the best one among these local optimums.

At the global optimization phase, we find the closest designs to the desired specifications by implementing a grid search that covers the design space. To this end, we apply the NN functionality estimator to each set of parameters in the design space. Then, the estimated result specifications are compared with the desired ones. Note that the design spaces of sub-circuits are relatively small since they have only a few design parameters (4-8). Moreover, once the NN model is trained, it is able to execute a large input set in a very short time. So, the global optimization phase takes only a few seconds.

Particle Swarm Optimization (PSO) is implemented as the local optimization engine. In PSO, if we denote the position of the i^{th} particle at iteration k as X_i^k, we will have:

$$X_i^{k+1} = X_i^k + V_i^{k+1}, \tag{6}$$

where V_i^k is the velocity of particle i^{th} at iteration k. The velocity is updated as:

$$V_i^{k+1} = wV_i^k + b_1 r_1 (pbest_i - X_i^k) + b_2 r_2 (gbest - X_i^k), \tag{7}$$

where r_1, r_2 are random numbers and w, b_1, b_2 are constant hyperparameters. Furthermore, $pbest_i$ denotes the position that gives the best-explored value for the i^{th} particle while $gbest$ is the best-explored value by all the particles [38]. In order to accelerate the local optimization process, we use the trained NN to estimate the functionality of sub-circuits instead of invoking SPICE.

As explained in more detail in Section V, we have tested multiple algorithms as the global and local optimization engines. Using any of the simulation-based algorithms as the global engine makes the process slow as it requires many iterations ($\sim$ 300-1500), and such iterations happen sequentially. Moreover, for the local engine, one of the advantages of PSO over other optimization algorithms (*e.g.* Bayesian Optimization (BO), Simulated Annealing (SA), etc.) is that PSO processes multiple particles at each iteration which causes much faster design space exploration. Especially since we are using the NN regression model as the circuit simulator, this process is very fast. We use only 5-10 iterations in the local phase using PSO.

E. Constraints Transformation

As it was mentioned earlier, AnGeL is designed to get the desired overall specifications, $S_o = [S_{o1}, \ldots, S_{oM}]$, as an input while all constraints are defined in equality format *e.g.* $S_{o1} = C_1, S_{o2} = C_2, \ldots$. The goal of AnGeL is to implement a circuit that its output specifications are as close as possible to the given ones. However, it is common to have constraints in

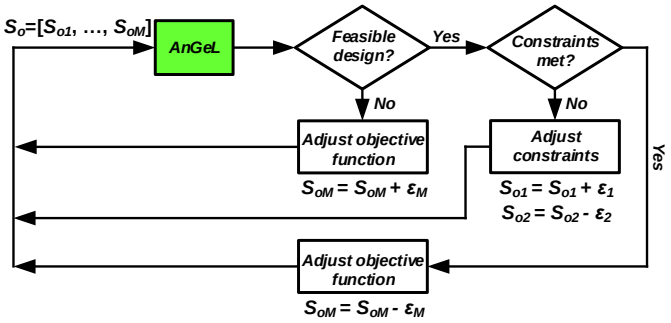

Fig. 5. Constraints transformation algorithm flow chart based on Equation (8). The initial target is set as $S_o = [C_1, \ldots, C_{M-1}, C_M]$ where $C_M = \frac{S_{oM}^{high} + S_{oM}^{low}}{2}$.

TABLE II
STATISTICS OF TESTED SPECIFICATIONS FOR DIFFERENT OPAMP SUB-CIRCUITS.

Sub-circuit	Specification	Min	Max	Average	SD
Gain block	BW [GHz]	0.001	15	0.17	0.68
	Gain [dB]	-0.89	53.2	33.48	9.46
	Power [mW]	0.02	28.8	2.55	2.29
	Noise [mV]	0.009	1.31	0.09	0.06
	Swing [V]	0	0.95	0.62	0.21
	PM [°]*	-121	170	99.8	35.4
	GM [dB]**	-20	20	3.2	4.5
DC-biasing	V_{out} [V]	0.25	0.8	0.55	0.12
	C_{in} [μF]	5	99.6	40.1	10.5
	R_{out} [kΩ]	35	400	250	50
Current source	Current [mA]	0.01	20.4	1.6	1.5

* Phase margin; PM is defined only for two-stage OPAMPs.
** Gain margin; GM is defined only for two-stage OPAMPs.

inequality format along with an objective function, as shown in Equation (8).

$$\begin{aligned} &\text{minimize } S_{oM} \\ &\text{subject to: } \begin{cases} S_{o1} > C_1 \\ S_{o2} < C_2 \\ \vdots \\ S_{oM-1} > C_{M-1} \end{cases} \end{aligned} \tag{8}$$

We use the algorithm depicted in Fig. 5 to solve Equation (8) using AnGeL. At each iteration, $S_o = [S_{o1}, \ldots, S_{oM}]$ is given to AnGeL as the overall specifications where the values of $S_{o1}, \ldots, S_{oM}$ are determined by the algorithm. We use the idea of binary search [48] for optimizing the objective function, S_{oM}. Initially, $S_o = [C_1, \ldots, C_{M-1}, C_M]$ is given where C_M is the average of supporting values for S_{oM}, *i.e.* $C_M = \frac{S_{oM}^{high} + S_{oM}^{low}}{2}$. If AnGeL is able to design a circuit with the given specifications, it means the design is feasible and we check if the constraints are met as the next step. Otherwise, this means the chosen value for the objective function is too small. Therefore, a bigger value *i.e.* $S_{oM} = S_{oM} + \epsilon_M$, will be targeted for the next iteration where $\epsilon_M > 0$.

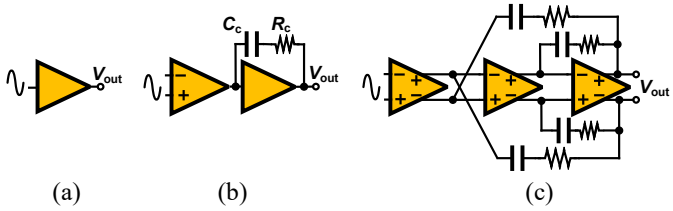

Fig. 6. The supported high-level topologies for OPAMPs. All stages for one- and two-stage OPAMPs can have either single-ended or differential inputs and outputs. There is a DC-biasing sub-circuit before the second stage gain block that is not shown here. (a) One-stage OPAMP. (b) Two-stage OPAMP. (c) There-stage OPAMP.

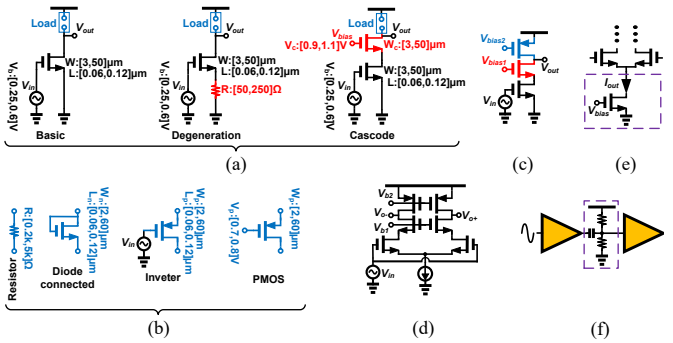

Fig. 7. The supported sub-circuits for OPAMPs. (a) Gain block sub-circuit body structures; The range of supported design parameters is written next to each in the form of [low, high]. (b) Gain block sub-circuit loads; The range of supported design parameters is written next to each in the form of [low, high]. (c) An example of a body structure with a load. (d) An example of the differential mode of a single-ended topology (circuit (c)). (e) The topology of the current source type sub-circuit. (f) The topology of DC-biasing type sub-circuit.

To check if the constraints are met, the output of AnGeL is simulated either by SPICE or by AnGeL's functional estimator models. If the constraints are met, the target value of the objective function will be reduced *i.e.* $S_{oM} = S_{oM} - \epsilon_M$. Otherwise, the constraints that are not met would be adjusted. For this purpose, target specifications are increased (if $S_{oi} > C_i$) or decreased (if $S_{oi} < C_i$) to give more margin for constraint satisfaction in the next iteration. For example, in Equation (8), if S_{o1} and S_{o2} are not met, $S_{o1} = S_{o1} + \epsilon_1$ and $S_{o2} = S_{o2} - \epsilon_2$ in the next iteration. The maximum number of iterations for adjusting the constraints and the objective function, as well as the values of $\epsilon_i > 0 \; \forall i \in \{1, \ldots, M\}$ are given by users.

V. EVALUATION

In this section, the implementation of one-, two-, and three-stage OPAMPs as well as filters using AnGeL is evaluated as examples. The performance of the synthesized circuits by AnGeL is validated by SPICE simulations. The design parameters include all transistor sizes (W, L, v_{bias}) as well as capacitor and resistor values. All the SPICE simulations are in the 55nm technology node and based on the pre-layout parasitic analysis. Going from pre-layout parasitic to post-layout can be done using transfer learning [30], which is out of the scope of this work. It should be noted that in the post-layout, the value of our work would be even shown more as more time would be saved using our method in comparison to

TABLE III
NUMBER OF SIMULATIONS AND GENERATED DATA USING THE TOPOLOGY-COMBINER MODEL FOR SINGLE-STAGE OPAMPS.

Sub-circuit	Simulations #*	Unlabeled generated data #
Single-ended gain block	7,350	16,400
Differential gain block	450	23,300
Current source	450	0
DC-biasing	250	0

* #: Number.

TABLE IV
MAPE OF THE DETERMINED SPECIFICATIONS BY THE TOPOLOGY COMBINER. MORE THAN 650 INSTANCES FROM 17 DIFFERENT SINGLE-STAGE OPAMP TOPOLOGIES ARE TESTED FOR EACH SPECIFICATION.

Specification	BW	Gain	Power	Noise	Swing
MAPE	0.070	0.029	0.049	0.051	0.038

the conventional approaches. Also, all the NN models are built using the TensorFlow platform with the Adam optimizer. The learning rate is set to 0.001 and RELU is used as the activation function for all hidden layers. In order to avoid overfitting, the idea of early stopping [43] with the patience parameter of 150 is implemented. For this purpose, 10% of the data are used for validation during the training phase. In order to validate the results properly, a random separate test set with the size of 10% of the training set is used. Moreover, Scikit-learn is used for the training and testing of all random forest and SVM models. We use [49] for implementing PSO. All the training and testing of our models are run on a server with an NVIDIA GA102 GPU.

A. OPAMPs

1) Database Generation & Overview: Fig. 6 demonstrates the supported high-level topologies. The supported sub-circuits (*i.e.* gain block, current source, and DC-biasing) are shown in Fig. 7. For the gain block sub-circuits, all combinations of 3 body structures with 4 loads that are shown in Fig. 7(a), and (b) are supported in the single-ended and differential modes. So, in total, $3 \times 4 \times 2 = 24$ different topologies are supported for the gain block sub-circuits. The range of supported design parameters is written next to each in the form of [low, high] in Fig. 7(a), and (b). The specifications for such DC-biasing and current source sub-circuits are determined by the associated gain stage through the feedback loop that is explained in Fig. 3. The supporting specifications statistics for each circuit type are summarized in Table II.

From twelve supporting single-ended gain block sub-circuits in Fig. 7(a)-(b), the topology-combiner model is used for generating the dataset of six of them *i.e.* the degeneration and Cascode body structures with all loads except the resistor. Moreover, the topology-combiner model generates the datasets of all twelve differential gain stages except the one with the basic body and resistive load. The main and accessory NNs in our topology combiner model have 3 hidden layers with 64 nodes. As indicated in Table III, for single-ended and differential gain blocks, the generated data are 2.23x and 51.74x more than the labeled data (simulations), respectively. Assuming each SPICE run takes 4s, that results in a time savings of more than 44h. The reason that the required number of labeled data for complicated differential sub-circuits is less than single-ended, is that they are generated by combining single-ended topologies. This shows the beauty of our topology combiner model that when topologies get more complicated, we need less labeled data to process them.

TABLE V
REQUIRED NUMBER OF LABELED DATA PER TOPOLOGY COMPARISON BETWEEN THE STATE-OF-THE-ART WORKS TO ACHIEVE AN AVERAGE MAPE OF 0.045 IN ESTIMATING THE FUNCTIONALITY OF OPAMPS.

Work	CCI-NN [4]			CCI-NN + TC*			AnGeL		
Number of stages	1	2	3	1	2	3	1	2	3
Data per topology	2,000	4,200	6,200	355	750	1,200	355	20	6

* TC: Topology Combiner.

Testing on more than 650 instances from 17 different topologies for 5 specifications shows the topology combiner has an average Mean Absolute Percentage Error (MAPE) of 0.047. MAPE is calculated as $\sum_{i=1}^{n} \frac{1}{n} |\frac{y_{ti}-y_{pi}}{y_{ti}}|$, where y_{ti} and y_{pi} are the i^{th} true and predicted instance, respectively and n is the total number of instances. Table IV lists MAPE of each specification.

The training data for all of the following evaluations are gathered from both labeled and unlabeled generated data. In order to validate the results properly, a random separate test set is built by SPICE runs.

2) Overall Topology Decider: In total, 14,000 labeled samples are used for the training of the net specification calculator models in two high-level topologies *i.e.* Fig. 6(b) and (c). 250,000 pseudo samples of two-, and three-stage OPAMP specifications are generated and fed to the specification breaker model.

The net specification calculator and specification breaker models enable us to support $24 \times 24 = 576$ two-stage OPAMP topologies as we support 24 topologies for each stage. Moreover, we support $12 \times 12 \times 12 = 1,728$ three-stage OPAMP topologies (we only support differential gain for three-stage OPAMPs). This means with only having datasets of 7 topologies from labeled data, we support 2,328 different one-, two, and three-stage OPAMP topologies. The number of design parameters in such OPAMPs varies between 4-31.

The required number of labeled training data of AnGeL is compared with the state-of-the-art work [4]. Hassanpourghadi *et al.* [4] divide the circuit into sub-circuits and implement a Circuit-Connectivity-Inspired ANN (CCI-NN) model. However, they use neither the idea of the topology combiner nor the net specification calculator model. In order to measure the usefulness of the net specification calculator model in multistage circuits, we also analyze when CCI-NN is integrated with the topology combiner. For this purpose, we test all works on 3,500 instances while all have the same

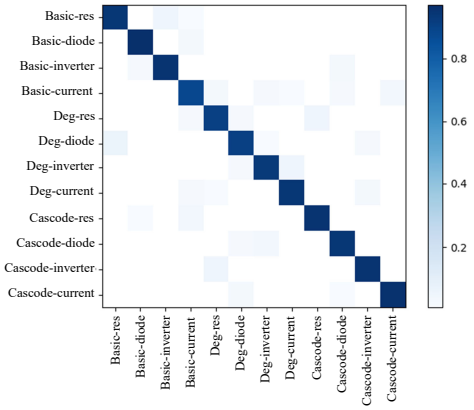

Fig. 8. The confusion matrix of OPAMPs topology selection. The NN model with [128, 128, 128, 128] nodes is used for this purpose which gives the highest accuracy among the evaluated methods.

TABLE VI
AVERAGE RUNTIME AND MAPE COMPARISON OF DIFFERENT SUB-CIRCUIT SIZING MODELS.

Optimization engine		Parameters	Estimator	Runtime[s]	MAPE
Global	Local				
NN	PSO	Iterations:10	NN	34.15	0.059
PSO	PSO	Iterations:50, Particles: 3,000	NN	100.4	0.09
			SPICE	7,800,000	0.018
SA	SA	Iterations: 5,000	NN	2,562	0.17
			SPICE	237,465	0.04
BO	BO	Iterations:500	NN	510	0.179
			SPICE	26,912	0.08

average MAPE of 0.045 in estimating the functionality of circuits. AnGeL that integrates both the topology combiner and net specification calculator model requires 22,500 labeled data in total which means it needs around 5.5 labeled data per topology for covering 1728 three-stage OPAMP topologies. Using the net specification calculator model, we estimate the functionality of multistage OPAMPs with different topologies without requiring a separate dataset for each topology. As summarized in Table V, for three-stage OPAMPs, CCI-NN [4] and CCI-NN integrated with the topology combiner need 1,090x and 200x more labeled data than AnGeL, respectively.

3) Sub-circuit Topology-Selector: Random forest, SVM (linear, polynomial, sigmoid, and RBF kernels), as well as NN classifier models, are analyzed for the topology-selector. For the gain block sub-circuits, there are 40,000 and 4,000 samples for the training and test sets, respectively. Our evaluation shows the NN with four layers and 128 nodes at each layer gives an accuracy of 92.9% which is the highest among the compared methods. This NN is used as the topology-selector model. Fig. 8 shows the confusion matrix of the predicted topologies using this model. It should be noted that even if a topology is not classified correctly, it does not mean that AnGeL cannot meet the specifications with that topology. This is because there may be more than one topology that can achieve the same specifications.

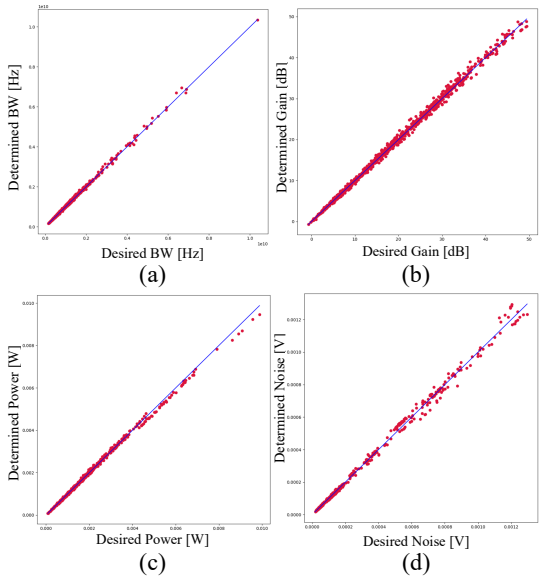

Fig. 9. Desired vs determined values by running the entire AnGeL platform for different specifications. More than 1,200 instances including one-, two-, and there-stage OPAMPs are tested. (a) BW. (b) Gain. (c) Power. (d) Noise.

TABLE VII
AN EXAMPLE OF THE OPAMP SPECIFICATIONS GIVEN TO ANGEL AND THE SIMULATED SPECIFICATIONS OF THE ASSOCIATED GENERATED CIRCUIT BY ANGEL.

Spec	BW	Gain	GM	PM	Noise	Swing	Power
1^{st} stage desired	1.35 GHz	16.5 dB	-*	-*	0.15 mV	0.4V	1.5 mW
1^{st} stage AnGeL	1.4 GHz	17.1 dB	-	-	0.14 mV	0.43V	1.6 mW
2^{nd} stage desired	0.9 GHz	17.5 dB	-	-	0.15 mV	0.50V	2.4 mW
2^{nd} stage AnGeL	0.94 GHz	17.3 dB	-	-	0.15 mV	0.53V	2.2 mW
3^{rd} stage desired	1.8 GHz	15 dB	-	-	0.15 mV	0.75V	2.4 mW
3^{rd} stage AnGeL	1.7 GHz	15 dB	-	-	0.15 mV	0.73V	2.3 mW
Overall desired	15 MHz	49 dB	5 dB	60°	0.23 mV	0.75V	6.3 mW
Overall AnGeL	15.5 MHz	49.4 dB	5.3 dB	62.1°	0.22 mV	0.73V	6.1 mW

* GM & PM is defined only for the overall circuit not for the sub-circuits.

4) Sub-circuit Sizing: As it was mentioned earlier, we use NN and PSO as the global and local optimization engines, respectively. Table VI summarizes the runtime and MAPE as well as the parameters of different methods when tested on more than 30 samples. Our approach results in the minimum runtime while it has a reasonable MAPE of 0.059. In fact, the runtime of our approach is 2.93x - 228,400x faster than other methods while it has better or comparable MAPE. The NN for the sub-circuit sizing model has 3 hidden layers with 64 nodes. For a better comparison, training time is taken into account too, which is 34,000s assuming each SPICE run takes 4s on average for generating each of the 8,500 labeled data. When the estimator is the NN, it means the same training set

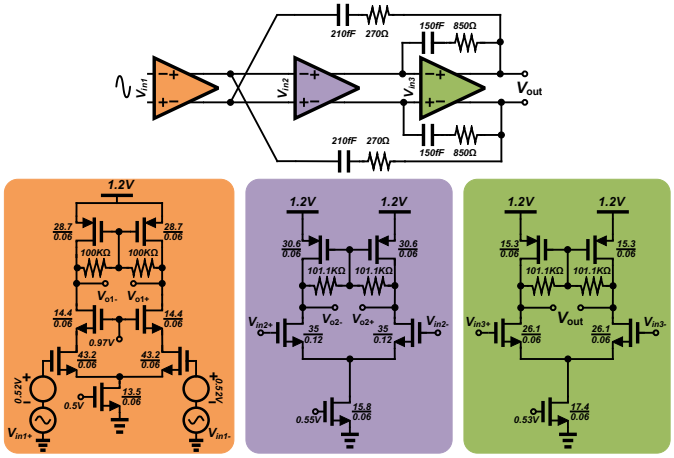

Fig. 10. The generated OPAMP circuit by AnGeL with its sizing to meet Table VII specifications. Size of transistors are shown as $\frac{W(\mu)}{L(\mu)}$.

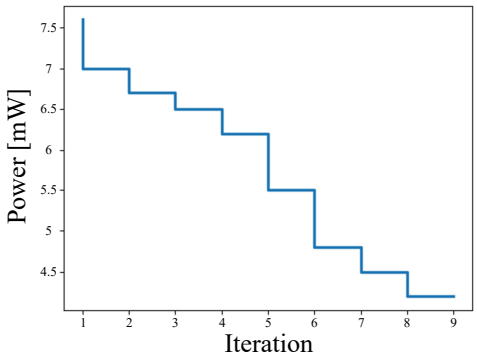

Fig. 11. Power at each iteration step while constraints in Equation (9) are satisfied.

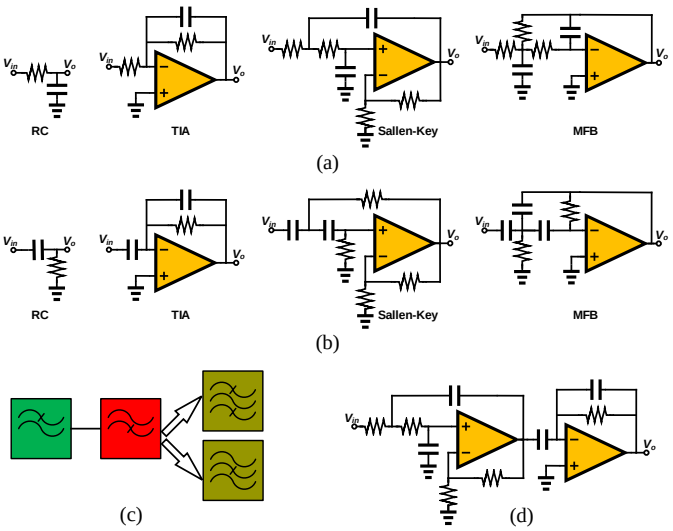

Fig. 12. The supported sub-circuits for filters. (a) Low-pass filters. (b) High-pass filters. (c) Cascading LP and HP filters results in band-pass or band-stop filters based on the relative position of the corner frequency of HP/LP filters. (d) An example of a band-pass/band-stop filter which is created by cascading a Sallen-Key LP with a TIA HP.

TABLE VIII
STATISTICS OF TESTED SPECIFICATIONS FOR FILTERS.

Specification	Min	Max	Average	SD
Pass-band BW (ω_{3dB}) [MHz]	10^{-4}	46.5	0.3	0.4
Stop-band frequency [MHz]*	1.5×10^{-6}	2.5×10^{3}	0.12	0.19
Gain [dB]	-13.4	18.6	2.3	5.2
Power [mW]	0.02	24.5	1.9	1.8
Noise [mV]	10^{-10}	8	2.2	4.9
Overshoot [dB]	6×10^{-13}	12.1	0.32	1.26
Group delay [μs]	1.5×10^{-4}	20.7	0.28	1.3

* The frequency that the gain has 40dB attenuation compared to the ω_{3dB} pass-band frequency.

as ours is used for other approaches (*i.e.* when both global and local engines are PSO, SA, and BO). So, our approach still outperforms them. However, considering both training and deployment time on a single run when SPICE is used as the estimator, BO as both global and local engines is faster than our method. In such conditions based on Table VI, even though our MAPE is better, BO (which has the lowest SPICE runtime) takes 26,912s while our method takes 34,034s. In fact, the NN estimator shows its actual advantage when the tool is used multiple times for generating circuits in the inference mode. For instance, if we run the tool 10 times, our approach takes 34,341.5s while BO takes 269,120s which means we save 65.21 hours during these runs using our method.

5) AnGeL Entire Platform: The entire AnGeL platform, from selecting the topology to sizing, is tested on more than 1,200 samples including one-, two-, and there-stage OPAMPs. The samples cover Table II specifications. The circuits that are generated by AnGeL for the given specifications are simulated to evaluate the performance. Fig. 9 shows the desired vs determined values for different specifications. An average MAPE of 0.027 is achieved in total.

An example of these 1,200 desired specifications is as follows:

$$\text{minimize power}$$
$$\text{subject to: } \begin{cases} \text{BW} > 15\text{ MHz}, \text{Gain} > 48\text{ dB}, \\ \text{GM} > 0\text{ dB}, \text{Noise} < 0.25\text{ mV}, \\ \text{PM} > 60^\circ, \text{Swing} > 70\text{ mV}. \end{cases} \quad (9)$$

Fig. 10 demonstrates the generated circuit by AnGeL in one of the iterations for minimizing the power while the constraints are satisfied. There are 27 design parameters in this design. Table VII summarizes the desired and determined specifications of each stage as well as the overall circuit performance. Each stage desired specifications are generated by the overall topology decider module. The overall circuit desired specifications are determined by the constraint transformer module (Fig. 5). Fig. 11 shows how the power gets smaller at each iteration. 4.2mW is the minimum achieved power while all constraints are met.

B. Filters

1) Database Generation & Overview: In general, there are two types of filters: passive and active. Passive filters use

TABLE IX
REQUIRED NUMBER OF LABELED DATA PER TOPOLOGY COMPARISON BETWEEN THE STATE-OF-THE-ART WORKS TO ACHIEVE AN AVERAGE MAPE OF 0.06 IN ESTIMATING THE FUNCTIONALITY OF FILTERS.

Work	CCI-NN [4]	Hierarchical CCI-NN	AnGeL
Non-ideal OPAMP models	1,012	1,012	212
Transistor-level OPAMPs models	4,800	0.0253	0.0032

TABLE X
MODEL ACCURACY COMPARISON FOR THE FILTER TOPOLOGY-SELECTOR. THERE ARE 8 TOPOLOGIES FOR HP & LP AND 32 TOPOLOGIES FOR BAND-PASS & BAND-STOP FILTERS IN TOTAL.

Model	Hyperparameters	Accuracy	
		HP & LP	Band-pass & band-stop
Random Forest	n_estimators = 10	99.68%	99.41%
	n_estimators = 100	99.71%	99.70%
	n_estimators = 400	99.69%	99.63%
SVM	Linear, C = 1	92.16%	86.54%
	Polynomial, C = 1, γ = 1, degree = 3	95.44%	93.12%
	Sigmoid, C = 1, γ = 1	40.17%	27.95%
	RBF, C = 1, γ = 1	95.45%	90.98%
DNN	NN layers: [64, 64, 64]	99.49%	94.40%
	NN layers: [128, 128, 128]	99.42%	95.22%
	NN layers: [128, 128, 128, 128]	99.74%	94.83%

only passive elements *e.g.* resistors, inductors, and capacitors, while active filters use active components such as OPAMPs in addition to passive elements. Fig. 12 shows the supported filter sub-circuits. There are 4 topologies for each of Low-Pass (LP) and High-Pass (HP) filters. Moreover, band-pass and band-stop filters are achievable by cascading LP and HP filters as illustrated in Fig. 12(c). So, there are $4 \times 4 = 16$ different topologies for each of band-pass and band-stop filters which leads to support 40 filters in total. The specifications for the OPAMP in active filters are determined by the associated filter stage through the feedback loop that is explained in Fig. 3. Taking one- and two-stage OPAMP topologies into account results in supporting more than 6,400,000 topologies. The number of design parameters varies between 2-42. The supporting specifications statistics for filters are listed in Table VIII.

In total, 8,500 simulations are performed as the labeled dataset for all filters assuming a non-ideal OPAMP model is used in active filters. By a non-ideal OPAMP model we mean an OPAMP with limited gain and bandwidth whose characteristic is modeled however, it is not made by circuit elements *e.g.* transistors. Moreover, 32,000 unlabeled data are generated using the topology-combiner model. The labeled data is used for generating the database of LP and HP filters. The datasets of band-pass and band-stop filters are generated by the topology-combiner model (except when RC is used in both stages of the band-pass or band-stop filters). Testing on more than 800 instances from 32 different topologies for 5 specifications shows the topology combiner has an average MAPE of 0.04.

TABLE XI
MAPE OF DETERMINED SPECIFICATIONS BY RUNNING THE ENTIRE ANGEL PLATFORM. MORE THAN 1,200 FILTER INSTANCES ARE TESTED.

Spec	Pass-band BW	Stop-band frequency	Gain	Overshoot	Group delay
MAPE	0.044	0.065	0.056	0.068	0.067

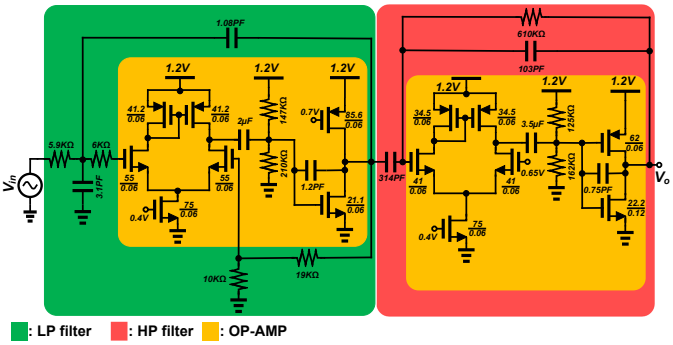

Fig. 13. The generated band-pass filter by AnGeL with its sizing to meet Table XII specifications. Size of transistors are shown as $\frac{W(\mu)}{L(\mu)}$.

Similar to OPAMPs, the required number of labeled training data of AnGeL is compared with the state-of-the-art works while all approaches achieve an average MAPE of 0.06 in estimating the functionality of filters testing on more than 1,200 samples. We also, analyze when the idea of the hierarchical design is used in CCI-NN meaning the specifications of OPAMPs in active filters are determined based on the specifications of the filter. This hierarchical approach allows for smaller training set as it breaks down the circuit. As it is summarized in Table IX, when actual transistor-level OPAMPs are used, hierarchical CCI-NN requires significantly less data than the normal CCI-NN [4]. Moreover, AnGeL requires 7.9x less labeled data in comparison with the hierarchical CCI-NN.

2) Sub-circuit Topology-Selector: There are 35,000 and 3,500 samples for the training and test sets, respectively. Some specifications of band-pass/band-stop filters are different from LP/HP filters so, they are analyzed separately. The accuracy of different models for topology classification is summarized in Table X.

3) AnGeL Entire Platform: Similar to OPAMPs, the entire AnGeL platform is tested on 1,200 samples which cover Table VIII. Actual transistor-level OPAMPs that are created by AnGeL in Section V-A are used in all active filter samples. It should be noted that more than 35,000 are used for the training of the sub-circuit sizing module. The NN model of the sub-circuit sizing module has 3 hidden layers with 64 nodes. Table XI summarizes the average MAPE for different specifications when the entire AnGeL is tested. An average MAPE of 0.06 is achieved in total.

Fig. 13 demonstrates an example of a band-pass filter designed by AnGeL, which is among such 1,200 testing samples and it is generated to meet specifications that are shown in Table XII. AnGeL has selected a Sallen-key topology for the LP filter and a TIA topology for the HP filter. There

TABLE XII
AN EXAMPLE OF THE FILTER SPECIFICATIONS GIVEN TO ANGEL AND THE SIMULATED SPECIFICATIONS OF THE ASSOCIATED GENERATED CIRCUIT BY ANGEL.

Specification	ω^{*}_{3dB} LP [MHz]	ω_{3dB} HP [KHz]	ω^{+}_{s} LP [MHz]	ω_s HP [Hz]	Gain [dB]	Power [mW]	Noise [mV]	Overshoot LP [μdB]	Overshoot HP [mdB]	Group delay [μs]
Band-pass desired	15	2	200	20	13	15	6	35	1	3
LP desired	15	-	170	-	7	8	3.5	35	-	$6 \times 10{-}3$
LP AnGeL output	15.9	-	172.7	-	7.7	7.56	3.3	32	-	$6.4 \times 10{-}3$
HP desired	-	2	-	20	6	7	3	-	1	3
HP AnGeL output	-	1.9	-	18.9	6.03	6.8	2.9	-	1.09	3.2
Band-pass AnGeL	15.9	1.9	205.1	18.9	13.73	14.36	5.9	32	1.09	3.2

* The 3dB bandwidth of pass-band frequencies.
+ The frequency that the gain has 40dB attenuation compared to the ω_{3dB} pass-band frequency.

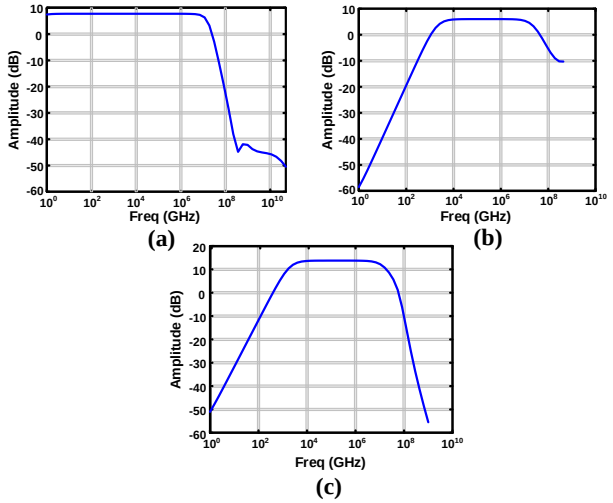

Fig. 14. The frequency responses of Fig. 13 filters to meet Table XII specifications. (a) LP filter. (b) HP filter. (c) Overall band-pass filter.

are 31 design parameters in this design. The LP and HP stage desired rows in Table XII, are generated by the overall topology decider module. The average MAPE is 0.054. Fig. 14 also illustrates the frequency responses of each LP and HP filter as well as the overall band-pass filter of Fig. 13.

VI. CONCLUSION

In this work, we first use neural networks to determine the behavior of complicated circuit topologies by combining the more simple ones and presenting a database. The number of labeled data in this dataset is 4.7x - 1090x less than the state-of-the-art works while maintaining the same accuracy. Using this database, we propose a fully-automated analog circuit generator framework, AnGeL. AnGeL performs all the necessary schematic circuit design steps, from deciding the circuit topology to sizing using NN models. To validate our work, multiple designs are generated by AnGeL which includes one-, two-, and three-stage OPAMPs as well as different filters. The results show that the runtime of AnGeL is 2.9x - 75x faster than the state-of-the-art works while maintaining the same accuracy.

REFERENCES

[1] R. Mina, C. Jabbour, and G. E. Sakr, "A Review of Machine Learning Techniques in Analog Integrated Circuit Design Automation," *Electronics*, vol. 11, no. 3, p. 435, 2022.
[2] S.-H. Lui, H.-K. Kwan, and N. Wong, "Analog circuit design by nonconvex polynomial optimization: Two design examples," *International Journal of Circuit Theory and Applications*, vol. 38, pp. 25–43, 2010.
[3] N. Lourenço *et al.*, "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*. IEEE, 2018, pp. 133–136.
[4] M. Hassanpourghadi *et al.*, "Circuit Connectivity Inspired Neural Network for Analog Mixed-Signal Functional Modeling," in *ACM/IEEE Design Automation Conference (DAC)*. IEEE, 2021, pp. 505–510.
[5] W. Lyu *et al.*, "An Efficient Bayesian Optimization Approach for Automated Optimization of Analog Circuits," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 65, no. 6, 2017.
[6] R. Phelps *et al.*, "Anaconda: Simulation-Based Synthesis of Analog Circuits Via Stochastic Pattern Search," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 19, no. 6, 2000.
[7] P. P. Prajapati and M. V. Shah, "Two Stage CMOS Operational Amplifier Design Using Particle Swarm Optimization Algorithm," in *IEEE UP Section Conference on Electrical Computer and Electronics (UPCON)*. IEEE, 2015, pp. 1–5.
[8] E. S. Ochotta, R. A. Rutenbar, and L. R. Carley, "Synthesis of High-Performance Analog Circuits in ASTRX/OBLX," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 15, no. 3, pp. 273–294, 1996.
[9] F. Wang *et al.*, "Co-learning Bayesian Model Fusion: Efficient Performance Modeling of Analog and Mixed-Signal Circuits Using Side Information," in *2015 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 2015, pp. 575–582.
[10] M. Fayazi *et al.*, "FASCINET: A Fully Automated Single-Board Computer Generator Using Neural Networks," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2022.
[11] Z. Wang, X. Luo, and Z. Gong, "Application of Deep Learning in Analog Circuit Sizing," in *Proceedings of the 2018 2nd International Conference on Computer Science and Artificial Intelligence*, 2018, pp. 571–575.
[12] A. F. Budak *et al.*, "Joint Optimization of Sizing and Layout for AMS Designs: Challenges and Opportunities," in *Proceedings of the 2023 International Symposium on Physical Design*, 2023, pp. 84–92.
[13] K. Settaluri *et al.*, "AutoCkt: Deep Reinforcement Learning of Analog Circuit Designs," in *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2020, pp. 490–495.
[14] R. Dreslinski *et al.*, "Fully-autonomous SoC Synthesis using Customizable Cell-Based Synthesizable Analog Circuits," University of Michigan, Ann Arbor, United States, Tech. Rep., 2019.
[15] M. B. Alawieh, F. Wang, and X. Li, "Efficient Hierarchical Performance Modeling for Analog and Mixed-Signal Circuits via Bayesian Co-Learning," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 12, pp. 2986–2998, 2018.
[16] M. Fayazi, Z. Colter, E. Afshari, and R. Dreslinski, "Applications of Artificial Intelligence on the Modeling and Optimization for Analog and Mixed-Signal Circuits: A Review," *IEEE Transactions on Circuits and Systems I: Regular Papers*, 2021.

[17] A. F. Budak *et al.*, "Cad for Analog/Mixed-Signal Integrated Circuits," *Advances in Semiconductor Technologies: Selected Topics Beyond Conventional CMOS*, pp. 43–60, 2022.

[18] M. Alawieh, F. Wang, and X. Li, "Efficient Hierarchical Performance Modeling for Integrated Circuits via Bayesian Co-Learning," in *Proceedings of the 54th Annual Design Automation Conference*, 2017.

[19] H. Wang *et al.*, "GCN-RL Circuit Designer: Transferable Transistor Sizing with Graph Neural Networks and Reinforcement Learning," in *57th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 2020.

[20] J. R. Koza, F. H. Bennett, D. Andre, M. A. Keane, and F. Dunlap, "Automated Synthesis of Analog Electrical Circuits by Means of Genetic Programming," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 2, pp. 109–128, 1997.

[21] F. Jiao and A. Doboli, "A Low-Voltage, Low-Power Amplifier Created by Reasoning-Based, Systematic Topology Synthesis," in *International Symposium on Circuits and Systems*. IEEE, 2015, pp. 2648–2651.

[22] E. Kaya, E. Afacan, and G. Dundar, "An Analog/RF Circuit Synthesis and Design Assistant Tool for Analog IP: DATA-IP," in *International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD)*. IEEE, 2018, pp. 1–9.

[23] R. S. Zebulum, M. A. Pacheco, and M. M. B. Vellasco, *Evolutionary Electronics: Automatic Design of Electronic Circuits and Systems by Genetic Algorithms*. CRC press, 2018.

[24] M. Meissner and L. Hedrich, "FEATS: Framework for Explorative Analog Topology Synthesis," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 2, 2014.

[25] I. Abel and H. Graeb, "FUBOCO: Structure Synthesis of Basic Op-Amps by FUnctional BlOck COmposition," *ACM Transactions on Design Automation of Electronic Systems*, vol. 27, no. 6, pp. 1–27, 2022.

[26] J. Lu, L. Lei, F. Yang, L. Shang, and X. Zeng, "Topology Optimization of Operational Amplifier in Continuous Space via Graph Embedding," in *2022 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2022, pp. 142–147.

[27] N. Kahraman and T. Yildirim, "Technology Independent Circuit Sizing for Fundamental Analog Circuits Using Artificial Neural Networks," in *Ph.D. Research in Microelectronics and Electronics*. IEEE, 2008.

[28] R. M. Hasani *et al.*, "Compositional Neural-Network Modeling of Complex Analog Circuits," in *2017 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2017, pp. 2235–2242.

[29] M. Fukuda, T. Ishii, and N. Takai, "OP-AMP sizing by inference of element values using machine learning," in *2017 International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS)*. IEEE, 2017, pp. 622–627.

[30] J. Liu *et al.*, "From Specification to Silicon: Towards Analog/Mixed-Signal Design Automation using Surrogate NN Models with Transfer Learning," in *2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*. IEEE, 2021, pp. 1–9.

[31] H. Wang, J. Yang, H.-S. Lee, and S. Han, "Learning to Design Circuits," *arXiv preprint arXiv:1812.02734*, 2018.

[32] Z. Dong *et al.*, "CktGNN: Circuit Graph Neural Network for Electronic Design Automation," in *The Eleventh International Conference on Learning Representations*, 2023.

[33] I. Abel, M. Neuner, and H. Graeb, "Constraint-programmed initial sizing of analog operational amplifiers," in *2019 IEEE 37th International Conference on Computer Design (ICCD)*. IEEE, 2019, pp. 413–421.

[34] E. Afacan, N. Lourenço, R. Martins, and G. Dündar, "Review: Machine learning techniques in analog/RF integrated circuit design, synthesis, layout, and test," *Integration*, vol. 77, pp. 113–130, 2021.

[35] J. Huang *et al.*, "An Analog Circuit Building Block Generator via Nested Multi-Fidelity Modeling," *IEEE Transactions on Circuits and Systems I: Regular Papers*, 2023.

[36] S. Zhang *et al.*, "Bayesian Optimization Approach for Analog Circuit Synthesis Using Neural Network," in *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2019, pp. 1463–1468.

[37] R. A. Thakker, M. S. Baghini, and M. B. Patil, "Low-Power Low-Voltage Analog Circuit Design using Hierarchical Particle Swarm Optimization," in *International Conference on VLSI Design*. IEEE, 2009, pp. 427–432.

[38] D. Joshi, S. Dash, U. Agarwal, R. Bhattacharjee, and G. Trivedi, "Analog Circuit Optimization Based on Hybrid Particle Swarm Optimization," in *2015 International Conference on Computational Science and Computational Intelligence (CSCI)*. IEEE, 2015, pp. 164–169.

[39] D. Joshi, S. Dash, S. Reddy, R. Manigilla, and G. Trivedi, "Multi-objective Hybrid Particle Swarm Optimization and its Application to Analog and RF Circuit Optimization," *Circuits, Systems, and Signal Processing*, pp. 1–27, 2023.

[40] Y. Li *et al.*, "An Artificial Neural Network Assisted Optimization System for Analog Design Space Exploration," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 10, 2019.

[41] 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.

[42] Z. Wu and I. Savidis, "Transfer Learning for Reuse of Analog Circuit Sizing Models Across Technology Nodes," in *IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2022, pp. 1033–1037.

[43] A. Gulli and S. Pal, *Deep Learning with Keras*. Packt Publishing Ltd, 2017.

[44] G. Biau and E. Scornet, "A random forest guided tour," *Test*, vol. 25, no. 2, pp. 197–227, 2016.

[45] M. Somvanshi, P. Chavan, S. Tambade, and S. Shinde, "A Review of Machine Learning Techniques using Decision Tree and Support Vector Machine," in *2016 International Conference on Computing Communication Control and Automation (ICCUBEA)*. IEEE, 2016, pp. 1–7.

[46] F. Provost, "Machine Learning from Imbalanced Data Sets 101," in *Proceedings of the AAAI'2000 Workshop on Imbalanced Data Sets*, vol. 68, no. 2000. AAAI Press, 2000, pp. 1–3.

[47] Y. Yang *et al.*, "Smart-MSP: A Self-Adaptive Multiple Starting Point Optimization Approach for Analog Circuit Synthesis," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 3, pp. 531–544, 2017.

[48] J. L. Bentley, "Multidimensional Binary Search Trees Used for Associative Searching," *Communications of the ACM*, vol. 18, no. 9, pp. 509–517, 1975.

[49] L. J. V. Miranda, "PySwarms, a research-toolkit for Particle Swarm Optimization in Python," *Journal of Open Source Software*, vol. 3, 2018. [Online]. Available: https://doi.org/10.21105/joss.00433

Morteza Fayazi was born in Tehran, Iran, in 1994. He received the B.Sc. major degree in electrical engineering and minor degree in computer science from Sharif University of Technology (SUT), Tehran, Iran, in 2017. He Also received the M.S. degree in electrical engineering and computer science in 2020 from University of Michigan, Ann Arbor, MI. He is currently working toward the Ph.D. degree at University of Michigan, Ann Arbor, MI. His research interests include circuit design automation and machine learning.

Morteza Tavakoli Taba received his B.S. degree in electrical engineering from Sharif University of Technology, Iran, in 2018 and M.S. from the University of Michigan, USA, in 2021. He is currently pursuing the Ph.D. degree at the University of Michigan, Ann Arbor, MI, USA. His research interests include mm-wave and Terahertz circuit design for applications such as FMCW radars and communication systems.

Ehsan Afshari was born in 1979. He received the B.Sc. degree in electronics engineering from Sharif University of Technology, Tehran, Iran, and the M.S. and Ph.D. degrees in electrical engineering from the California Institute of Technology, Pasadena, in 2003 and 2006, respectively. In Fall 2016, he joined the University of Michigan, Ann Arbor, MI, USA, as a Professor with the Department of Electrical Engineering and Computer Science. His research interests are high-speed and low-noise integrated circuits for applications in communication systems, sensing, and biomedical devices.

Ronald Dreslinski received the BSE degrees both in electrical engineering and computer engineering, and the MSE and PhD degrees in computer science and engineering from the University of Michigan, Ann Arbor, Michigan. He is currently an assistant professor of computer science and engineering with the University of Michigan. His research focuses on architectures that enable emerging low-power circuit techniques.