Accelerating OTA Circuit Design: Transistor Sizing Based on a Transformer Model and Precomputed Lookup Tables

Subhadip Ghosh[1], Endalk Y. Gebru[1], Chandramouli V. Kashyap[2], Ramesh Harjani[1], Sachin S. Sapatnekar[1]
[1]Department of Electrical and Computer Engineering, University of Minnesota, Minneapolis, MN, USA
[2]Cadence Design Systems, Portland, OR, USA

***Abstract*—Device sizing is crucial for meeting performance specifications in operational transconductance amplifiers (OTAs), and this work proposes an automated sizing framework based on a transformer model. The approach first leverages the driving-point signal flow graph (DP-SFG) to map an OTA circuit and its specifications into transformer-friendly sequential data. A specialized tokenization approach is applied to the sequential data to expedite the training of the transformer on a diverse range of OTA topologies, under multiple specifications. Under specific performance constraints, the trained transformer model is used to accurately predict DP-SFG parameters in the inference phase. The predicted DP-SFG parameters are then translated to transistor sizes using a precomputed look-up table-based approach inspired by the g_m/I_d methodology. In contrast to previous conventional or machine-learning-based methods, the proposed framework achieves significant improvements in both speed and computational efficiency by reducing the need for expensive SPICE simulations within the optimization loop; instead, almost all SPICE simulations are confined to the one-time training phase. The method is validated on a variety of unseen specifications, and the sizing solution demonstrates over 90% success in meeting specifications with just one SPICE simulation for validation, and 100% success with 3–5 additional SPICE simulations.**

I. Introduction

A "perfect storm" of events is driving the need for greater automation in analog design: increased demand for analog circuits within larger systems; a diminishing designer workforce that is hard-pressed to fulfill this demand; and increased design complexity due to the complexity of circuit models in advanced nodes. Today, it is imperative to develop automation techniques that improve designer productivity.

One of the most critical blocks in many analog systems is the operational transconductance amplifier (OTA), which performs tasks such as amplification, filtering, and signal conditioning. Transistor sizing for OTAs has long been a time-consuming process, relying on human experts to optimize circuit metrics under strict performance constraints. Early automated approaches employed knowledge-based methods [1], but codifying expertise into exhaustive rules is challenging, especially with the need for updates across technology nodes.

Methods based on stochastic search (genetic algorithms [2], [3], simulated annealing [4], and particle swarm optimization [5]) have been explored for OTA optimization, but accuracy requires numerous expensive SPICE simulations within the optimization loop. Equation-based techniques [6], [7] and approaches based on convex optimization, such as geometric programming using posynomial-form models [8], work for simplified MOS models, but can falter in the face of the complex device models in advanced nodes. Recent works have proposed ML-based techniques such as DNN-Opt [9], AutoCkt [10], and GCN-RL [11], and an RL approach using sensitivity analysis [12]. For these methods, OTA sizing under each new set of performance specifications requires numerous SPICE simulations.

We present a transistor sizing approach for OTA circuits using a transformer architecture [13]. This method employs an attention mechanism to capture complex nonlinear relationships between device parameters and circuit performance, addressing the intricacies of modern technology nodes. A key feature of our approach, in contrast with prior methods, is that the cost of SPICE simulations is confined to a one-time training phase. *In contrast with prior methods, for a given set of specifications, in the inference phase, the sizing solution can be obtained rapidly with very few SPICE simulations.*

The transformer requires the circuit characteristics to be represented as a character sequence that forms a set of input tokens to the transformer. We achieve this by utilizing the driving-point signal flow graph (DP-SFG) from [14], later modified in [15], to facilitate direct mapping of the schematic to a graph. We note that the DP-SFG representation of a circuit serves as a descriptive language, translating the behavior of the circuit into a character string that encapsulates its parameters and structure. Based on this observation, we map the transistor sizing problem to a language-translation task akin to natural language processing (NLP). For a given query representing desired performance specifications, the transformer is trained to output a string with the DP-SFG parameter values that meet these specifications. We translate DP-SFG parameters to predict the transistor sizes based on precomputed look-up tables (LUTs).

An ML-based approach does not guarantee perfect accuracy, and occasionally, the predicted design point may show minor violations in the specifications in some test cases. In these instances, the designer can re-invoke the fast transformer-based method with tighter design specifications until all requirements are met. Thus, the method acts as a copilot, keeping the designer in the loop.

The key contributions of our work are as follows:

- We implement an automated framework to map an OTA circuit to its equivalent DP-SFG. The paths of this DP-SFG are encoded and concatenated in a language that describes circuit behavior.
- We create a labeled training set for our ML model, corresponding to a range of device sizes, over multiple OTA topologies, and evaluate performance metrics using SPICE simulations.
- We customize a transformer-based encoder-decoder to encode paths in the DP-SFG to predict the DP-SFG parameters such as transconductance (g_m) and capacitance values.
- We utilize our LUTs, precharacterized using SPICE simulations, to translate the outputs of the transformer into device sizes, thus providing the sizing solution that meet specifications.

Our approach is flexible enough to be used within a layout optimization loop. After sizing, a layout engine updates parasitics, updating the parasitic values in the DP-SFG. Our model, trained on a range of values, can then be re-invoked without further SPICE simulations.

The paper is structured as follows. Section II overviews a set of core building blocks used in our approach, and is followed by Section III, which details our proposed sizing framework and its implementation. Next, Section IV details our experimental setup and findings, and finally, Section V concludes the paper.

This work was supported in part by NSF (award 2212345) and SRC.

978-3-9826741-0-0/DATE25/© 2025 EDAA

II. Background

In this section, we first overview the principles governing transformer architecture. Next, we present a concise overview of DP-SFGs, which we employ to map OTA circuits into transformer-friendly sequential data. Finally, we describe a precomputed LUT-based width estimator to translate DP-SFG parameters to transistor widths.

A. The transformer architecture

The transformer [13] is viewed as one of the most promising deep learning architectures for sequential data prediction in NLP. It relies on an attention mechanism that reveals interdependencies among sequence elements, even in long sequences. The architecture takes a series of inputs $(x_1, x_2, x_3, \ldots, x_n)$ and generates corresponding outputs $(y_1, y_2, y_3, \ldots, y_n)$.

The simplified architecture shown in Fig. 1 consists of N identical stacked encoder blocks, followed by N identical stacked decoder blocks. The encoder and decoder is fed by an input embedding block, which converts a discrete input sequence to a continuous representation for neural processing. Additionally, a positional encoding block encodes the relative or absolute positional details of each element in the sequence using sine-cosine encoding functions at different frequencies. This allows the model to comprehend the position of each element in the sequence, thus understanding its context. Each encoder block comprises a multi-head self-attention block and a position-wise feed-forward network (FFN); each decoder block, which has a similar structure to the encoder, consists of an additional multi-head cross-attention block, stacked between the multi-head self-attention and feed-forward blocks. The attention block tracks the correlation between elements in the sequence and builds a contextual representation of interdependencies using a scaled dot-product between the query (Q), key (K), and value (V) vectors:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V, \tag{1}$$

where d_k is the dimension of the query and key vectors. The FFN consists of two fully connected networks with an activation function and dropout after each network to avoid overfitting. The model features residual connections across the attention blocks and FFN to mitigate vanishing gradients and facilitate information flow.

B. Driving-point signal flow graphs

The input data sequence to the transformer must encode information that relates the parameters of a circuit to its performance metrics. Our method for representing circuit performance is based on the signal flow graph (SFG). The classical SFG proposed by Mason [16] provides a graph representation of linear time-invariant (LTI) systems, and maps on well to the analysis of linear analog circuits such as amplifiers. In our work, we employ the driving-point signal flow graph (DP-SFG) [14], [15]. The vertices of this graph are the set of excitations (voltage and current sources) in the circuit and internal states (e.g., voltages) in the circuit. An edge connects vertices with an electrical relationship, and the edge weight is the gain; for example, if a vertex z has two incoming edges from vertices x and y, with gains a and b, respectively, then $z = ax + by$, using the principle of superposition in LTI systems. To effectively use superposition to assess the impact of each node on every other node, the DP-SFG introduces auxiliary voltages at internal nodes of the circuit that are not connected to excitations. These auxiliary sources are structured to not to alter any currents or voltages in the original circuit, and simplifies the SFG formulation for circuit analysis.

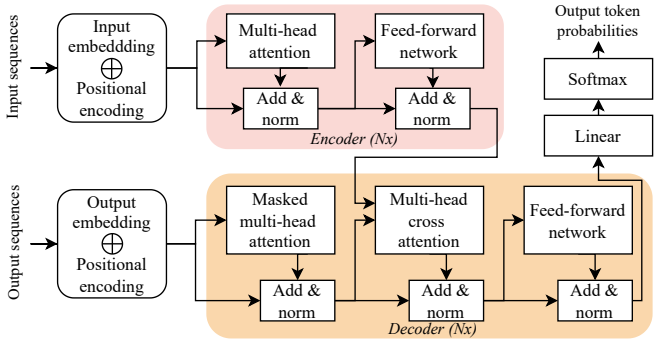


Figure 1: Architecture of a transformer.

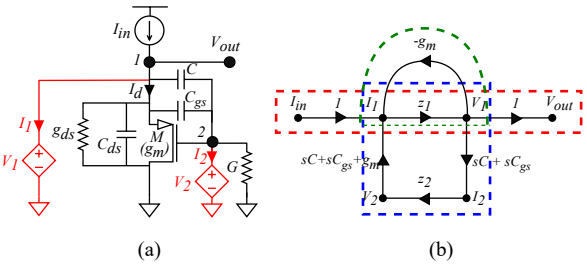


Figure 2: (a) Schematic and (b) DP-SFG for an active inductor.

Fig. 2(a) shows a circuit of an active inductor, which is an inductorless circuit that replicates the behavior of an inductor over a certain range of frequencies. Fig. 2(b) shows the equivalent DP-SFG. In Section III-B, we provide a detailed explanation that shows how a circuit may be mapped to its equivalent DP-SFG.

III. Proposed Methodology

A. Overview of the solution

We leverage transformer models to capture the complex relationships between device attributes and circuit performance. We conceptualize the transistor sizing problem as a language translation task, where the input sequence consists of a DP-SFG representation for an OTA circuit, together with performance specifications. The transformer model generates an enhanced DP-SFG representation with the device characteristics necessary to meet the specifications.

Fig. 3 illustrates the workflow of our framework, with four stages. Stage I performs preprocessing, generating the DP-SFG from the circuit netlist. The DP-SFG and the designer-specified performance constraints are then tokenized into a combined sequence. Next, in Stage II, a transformer model processes these tokens to predict circuit parameters that meet performance specifications; these are then translated to individual device widths in Stage III using the precomputed LUTs and a g_m/I_d methodology. Finally, in Stage IV,

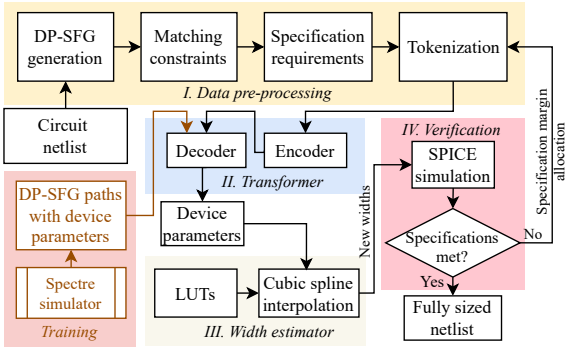


Figure 3: Overall sizing flow using our transformer-based method.

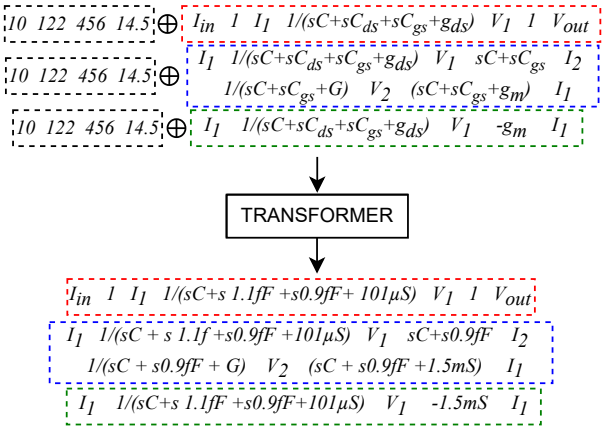


Figure 4: **Input:** DP-SFG paths with desired performance specifications, **Output:** Predicted sequences with device parameter values.

the performance of the predicted sized circuit is verified using SPICE simulation. In a vast majority of cases, we will show that the performance criteria are satisfied; if not, the designer is brought into the loop to provide tighter specifications, and procedure is reinvoked so that the original specifications are met. The remainder of this section discusses the detailed implementation of each stage.

B. Circuit-to-sequence mapping using the DP-SFG

The procedure for creating the DP-SFG formalizes the approach in [15], [17]. We use the running example of an active inductor circuit, whose DP-SFG is shown in Fig. 2(b) to illustrate the method.

Step 0: Initial bookkeeping and node initialization The algorithm begins with initializing the vertex (or node) set V, and initializing data structures for fast access to connectivity information between circuit components (RCs, transistors, etc.) from the netlist.

Step 1: Insertion of auxiliary nodes. For nodes that are not connected to voltage sources, we create auxiliary voltage sources. These sources are described by $V = zI$, where z is the driving point impedance (DPI) at the node, i.e., the the inverse sum of all conductances connected to the node. In Fig. 2, auxiliary sources are added at nodes 1 and 2. The sources replicate node voltages without introducing additional current into the circuit. This establishes relationships $V_1 = z_1 I_1$ and $V_2 = z_2 I_2$, where

$$z_1 = \frac{1}{sC + sC_{ds} + sC_{gs} + g_{ds}}, \quad z_2 = \frac{1}{sC + sC_{gs} + G} \tag{2}$$

Step 2: Adding branches due to passive components. Next, we add the edges associated with passive components. We consider how each terminal connects to auxiliary sources. If a terminal connects to an auxiliary source, we connect it to the auxiliary node of the other terminal using its admittance as the edge weight. If neither terminal connects to an auxiliary source, we connect them with an edge using the admittance of the component. In Fig. 2, the terminals of capacitor C are both connected to auxiliary sources carrying currents $I_1 = s(C + C_{gs})V_2$ and $I_2 = s(C + C_{gs})V_1$ through the associated edges.

Step 3: Adding branches due to transistor transconductances. The voltage at each terminal of a transistor influences its drain current. Based on these terminal voltages, we establish connections that directly or indirectly affect the auxiliary node currents. In the example of Fig. 2, if V_1 increases, the drain current I_d increases, and hence current flowing to I_1 decreases in the opposite direction. This is reflected by setting the weight on the branch from V_1 to I_1 to $-g_m$. Similarly, the branch V_2 to I_1 has weight $+g_m$, reflecting the positive dependence of drain current I_d with V_2.

At the end of Step 3, we obtain the final DP-SFG in Fig. 2(b) for the active inductor circuit. We will encode such a DP-SFG into sequential data that encapsulates the functionality of the circuit as well as the parameters of circuit components, including parasitics. This sequence representation is provided as an input to the transformer and is eventually used to size transistors in the circuit. We utilize the NetworkX Python package to process the final DP-SFG. This approach utilizes Johnson's algorithm ($O(V^2 \log V + VE)$ complexity) to identify all cycles, and the depth-first search algorithm ($O(V+E)$ complexity) to find all forward paths, where V represents the number of nodes and E represents the number of edges in the graph. For our running example, Fig. 4 shows the sequences obtained from the DPSFG. Specifically, the path outlined by red dotted rectangle represents the forward path between input and output nodes, while the blue and green outlined paths denote the cycles.

C. Implementation of the transformer

Transformer inputs. The transformer model comprehends the interdependencies between device parameters and circuit performance metrics. We frame the paths by concatenating the nodes and edge weights from the DPSFG for every forward path and loop. The transformer takes in the list of paths extracted from the DP-SFG, each augmented with the desired set of specifications outlined by the black dotted rectangle in Fig. 4. The transformer acts on the sequences and predicts the device parameters g_m, g_{ds}, C_{ds}, and C_{gs} which will satisfy the desired specifications.

Overall transformer architecture. We implement the transformer in Python, leveraging the PyTorch library. The architecture of the transformer model closely resembles the one proposed by [13], with minor modifications. We use a 720-dimensional input embedding with 12 heads of parallel attention layers, while keeping the rest of the parameters unchanged.

Tokenization and byte-pair encoding. Tokenization is a crucial step for optimizing transformer efficacy. It breaks down a long sequence into smaller entities called tokens. Unlike in traditional NLP, where individual words and sub-words are treated as tokens, we use specific groups of individual characters such as the key device parameters g_m, g_{ds}, C_{ds}, C_{gs}, edge weights, and the names of the transistors, as tokens that convey necessary information about the circuit.

For our problem, character-level tokenization (CLT), where each character is treated as a single token, is found to lead to long sequence lengths, i.e., a large number of tokens within a single sequence, resulting in computational inefficiency. To overcome this problem, we employ the byte-pair encoding (BPE) approach [18]. This approach iteratively combines the most frequently occurring tokens (bytes) into a single token, and dynamically adapts the vocabulary of the training data to capturing a common group of characters conveying essential information. By applying BPE, we achieve a 3.77× compression of the sequence lengths compared to the use of CLT, leading to substantial savings in training time and memory requirements.

To demonstrate tokenization and for an actual DP-SFG path, we choose a partial path of a five-transistor operational transconductance amplifier (5T-OTA). The results of CLT and BPE are:

CLT: 32 gmP1 -16 1/(gdsM0+sCdsM0+sCdsP1+gmP1)

BPE: 32 2.5mSP1 -16 1/(567uSM0+s1.7aFM0+s541aFP1+2.5mSP1

The CLT sequence colors each neighboring character differently, denoting the tokenization of each unique character. However, CLT cannot easily comprehend the relation to device parameters or device names. The BPE approach overcomes this by iteratively combining frequently-occurring tokens. For example, BPE combines tokens such as gmP1, gdsM0, and CdsM0 to represent g_m of transistor P1, and

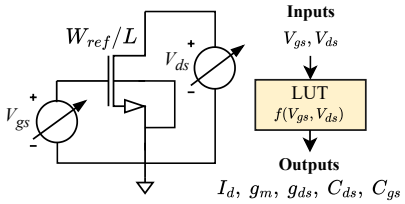


Figure 5: LUT generation and characterization.

g_{ds} and C_{ds} of transistor M0, respectively. Similarly, character-level tokens for the units of circuit parameters (e.g., "mS" or "aF") are combined into tokens of multiple characters. However, all purely numeric strings are left uncombined, as shown in the BPE sequence. For instance, for the value 2.5mS, which corresponds to the g_m of transistor P1, the tokens representing 2.5 are maintained as character-level tokens, enabling the transformer to predict each digit relative to performance metrics independently, but the two character-level tokens for "mS" are combined into a single token. This restricted BPE representation thus enables the transformer model to better comprehend circuit relationships, as compared with CLT.

Loss Function. To enhance the learning of device parameter prediction from specified inputs, we utilize a weighted cross-entropy loss function for the transformer. Each token is treated as a separate class, with the loss function assigning greater importance to classes critical for accurate predictions. We focus on tokens representing numerical values of device characteristics (e.g., g_m, g_{ds}, C_{ds}, and C_{gs}), ensuring they receive more attention during training. This approach allows the transformer to grasp the significance of these characteristics and their impact on performance. Our experiments compared unweighted and weighted loss functions with varying weights, revealing that applying a 20% increased weight on the numerical tokens yielded optimal performance.

D. Translating circuit parameters to device widths

After the trained transformer predicts the values of circuit parameters, they must be transformed to device widths. In this section, we describe a methodology for this purpose.

1) ***Device characterization****:* In older technologies, the square-law model for MOS transistors could be used to perform a translation between circuit parameters and transistor widths, but square-law behavior is inadequate for capturing the complexities of modern MOS transistor models. In this work, we use a precomputed lookup table (LUT) that rapidly performs the mapping to device sizes while incorporating the complexities of advanced MOS models.

The LUT is indexed by the V_{gs}, V_{ds}, and length L of the transistor, and provides five outputs: the drain current (I_d), transconductance (g_m), drain-source conductance (g_{ds}), drain-source capacitance (C_{ds}), and gate-source capacitance (C_{gs}) all computed per unit transistor width. The entries of the LUT are computed by performing a nested DC sweep simulation across the input indices for the MOSFET with a specific reference width, W_{ref}, as shown in Fig. 5, and for each input combination, the five outputs are recorded. Since the five quantities all vary linearly with the width of the transistor, we store their corresponding values per unit width. The LUT stores the vector-valued function

$$[I_d \ \ g_m \ \ g_{ds} \ \ C_{ds} \ \ C_{gs}] = f(V_{gs}, V_{ds}) \tag{3}$$

We have constructed a lookup table for a 65nm technology with a reference transistor width of 700nm, with V_{gs} and V_{ds} values ranging from 0–1.2V with a 60mV step. Given the relatively coarse granularity of data points in the LUT, we have implemented cubic spline interpolation to enhance accuracy at intermediate values. These LUT granularity, together with interpolation, ensures that it provides accurate predictions, and yet has a reasonable size.

Algorithm 1 Width Estimation

1: **Inputs:** Transformer-predicted g_m^p, g_{ds}^p, C_{ds}^p, C_{gs}^p and current I_d^{in} for a MOSFET; LUT for the device type (PMOS/NMOS); tolerances α and ϵ
2: **Outputs:** Optimal width, W, of the MOSFET
3: $V_{ds,curr} \leftarrow V_{dd}/2$, $cost_{curr} \leftarrow \infty$, $\Delta \leftarrow \infty$
4: $g_m\_I_d \leftarrow g_m^p/I_d$ *// Compute the $g_m\_I_d$ operating point*
5: **while** $|\Delta| > \epsilon$ **do** *// until current cost $\approx$ previous cost*
6: mincost$_{prev} \leftarrow$ mincost$_{curr}$, $V_{ds,prev} \leftarrow V_{ds,curr}$
7: Find LUT entry $[I_d \ \ g_m \ \ g_{ds} \ \ C_{ds} \ \ C_{gs}] = f(V_{gs}, V_{ds})$ at which $g_m/I_d = g_m\_I_d$. Report V_{gs}^p for this entry.
8: At V_{gs}^p, the LUT for $[I_d \ \ g_m \ \ g_{ds} \ \ C_{ds} \ \ C_{gs}]$ is $f(V_{ds})$.
9: *// Find $w_1 \cdots w_5$ as functions of V_{ds}*
10: $w_1 \leftarrow g_m^p/g_m$, $w_2 \leftarrow g_{ds}^p/g_{ds}$, $w_3 \leftarrow C_{ds}^p/c_{ds}$, $w_4 \leftarrow C_{gs}^p/c_{gs}$, $w_5 \leftarrow I_d^{in}/I_d$
// Find V_{ds} at which w_is are closest
11: cost$(V_{ds}) \leftarrow \sum_{n=1}^{3} \sum_{m=n+1}^{4} |w_n - w_m|$
12: mincost$_{curr}$ = min$_{V_{ds}}$ (cost)
13: $\Delta \leftarrow$ mincost$_{prev}$ $-$ mincost$_{curr}$
// Updating the initial guess V_{ds} value
14: $V_{ds,curr} \leftarrow V_{ds,curr} + \text{sgn}(\Delta).\alpha.V_{ds,prev}$
15: **end while**
16: $W \leftarrow \ w_1(V_{ds})$

Our methodology uses this LUT, together with the g_m/I_d methodology [19], [20], to translate circuit parameters predicted by the transformer to transistor widths. The cornerstone of this methodology relies on the inherent width independence of the ratio g_m/I_d to estimate the unknown device width: this makes it feasible to use an LUT characterized for a reference width W_{ref}.

2) ***Width estimation****:* The width estimation process uses the recorded LUT and transformer-predicted MOSFET parameters to compute the optimal width. The pseudocode for the algorithm employed is presented in Algorithm 1. After initialization on line 3, the input is converted to the desired g_m/I_d ratio. Lines 6–14 iterate over the LUT to find the W that matches the transformer-supplied parameters. Specifically, line 7 finds the value of V_{gs} at which the g_m/I_d ratio is met. For this value, lines 11–13 determine candidate values of W, $w_1, \cdots, w_5$, by ratioing I_d^{in}, g_m^p, g_{ds}^p, C_{ds}^p, and C_{gs}^p, respectively, with the corresponding LUT outputs. We iterate over V_{ds} until $w_1, \cdots, w_5$ are as close as possible. Line 14 takes a step in this direction using the empirically chosen factor $\alpha = 10^{-4}$. The iterations continue until the candidate width values converge.

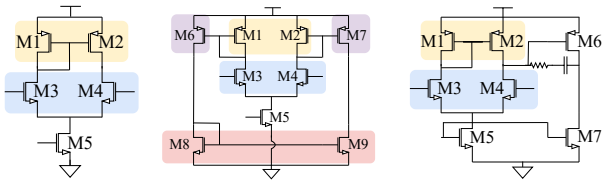
E. SPICE verification and margin allocation

Finally, we perform just one SPICE simulation to verify compliance with all specifications. If any specification deviates from the requirements, the model modifies the specifications and repeats the inference step to obtain a new set of device sizes. For example, if the gain of the sized OTA is 10% below the desired value, the model iteratively tightens the specifications to accommodate this 10% difference in the gain requirement until all specifications are satisfied.

IV. Experimental Setup and Results

A. Data generation and preprocessing

To demonstrate the efficacy of our framework, we employ three distinct OTA topologies: five-transistor OTA (5T-OTA), current-mirror OTA (CM-OTA), and two-stage OTA (2S-OTA), each implemented using a 65nm technology node. In Fig. 6, we show the OTA schematics along with matching constraints under consideration. For



(a) (b) (c)

Figure 6: Schematic of (a) 5T-OTA, (b) CM-OTA, and (c) 2S-OTA.

Table I: Dataset information.

Topology	Gain (dB) *min-max*	3dB bandwidth (MHz) *min-max*	UGF (MHz) *min-max*	#forward paths	#cycles
5T-OTA	18 – 23	7 – 54	80 – 871	9	4
CM-OTA	19 – 25	17.5 – 86	57 – 1185	26	5
2S-OTA	28 – 54	0.01 – 0.32	1.8 – 370	2	11

clarity in our demonstration, we focus on three performance metrics: gain, 3dB-bandwidth (BW), and unity-gain frequency (UGF), and aim to meet the given performance specifications.

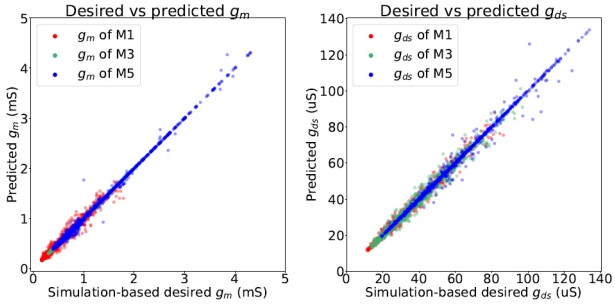
Table I shows the range of different specifications for the OTAs considered for our training set. We assume that the length of all the devices in a circuit is set to 180nm with a load capacitor C_L of 500fF for all the topologies. To ensure reliable model analysis, we start with precise data generation for each OTA topology using OCEAN scripting. This involves the following steps:

- Generating multiple designs with varying transistor sizes by nested sweeps of widths ranging from 0.7μm to 50μm.
- Enforcing matching constraints for active load current mirror (CM), and differential pair (DP).
- Sweeping the DC voltage to determine the input common-mode range (ICMR) of the designs.
- Ensuring that the CMs operate in the strong inversion region while the DPs function in the weak inversion region.
- Filtering out designs that falls out of the predefined specification range for the dataset outlined in Table I.
- Capturing the device parameters – specifically, g_m, g_{ds}, C_{ds}, and C_{gs} – for the final legal designs.

Next, we focus on generating appropriate DP-SFG paths for each circuit topology. Table I shows the number of sequential paths for each topology. The DP-SFGs are small and the cost of path enumeration is small; for more complex examples, if the number of paths grows large, it is possible to devise other string representations of the DP-SFG. Finally, in the preprocessing stage, we generate two sets of sequential data, one each for the encoder and the decoder.

- The sequential data at the encoder comprises DP-SFG paths that maintain consistency across all designs within a specific topology. It also includes performance metrics for each design, encompassing gain, BW, and UGF parameters, associated with each unique set of transistor sizes.
- The sequential data at the decoder covers the same DP-SFG paths, but with device parameters replaced by values obtained during data generation. These values are unique to each design, aligning with the performance metrics in the encoder sequence.

We train a single transformer model that works across all three OTA topologies. By considering all performance criteria and all DP-SFG paths, we convey complete information about each circuit to the transformer. Our dataset comprises 17,000 designs for 5T-OTA, 25,000 designs for CM-OTA, and 8,000 designs for 2S-OTA, each with a different set of transistor sizes. This diverse dataset trains the model across multiple design specification requirements.



(a) (b)

Figure 7: For 5T-OTA: Scatter plots showing comparisons between predicted and simulation-based device parameters (a) g_m and (b) g_{ds}.

B. Training and validation

For our experiments, we employ an Nvidia L40S GPU equipped with 45GB of memory. The dataset is split into an 80:20 ratio for training and validation across each OTA topology. We train a single model using datasets from all three topologies for 40 epochs, employing an adaptive learning rate strategy with the Adam optimizer, beginning with an initial learning rate of 10^{-4}. Subsequently, our framework is validated against unseen performance specifications across all three OTA topologies. For a given topology and performance specifications, the validation phase rapidly predicts a sequence of tokens corresponding to circuit parameters.The transformer takes in the encoder sequences list and predicts output sequences containing the device parameter values that satisfy the specifications. This is followed by the LUT-based estimator that translates the predicted device parameters to transistor widths.

C. Performance of the framework

We conduct comprehensive performance evaluations to assess the effectiveness of the transformer model and LUT-based width estimator for each OTA topology. Our method sizes 100 unique designs per topology, each with distinct performance specifications not included in the training set. For each specification, the transformer model predicts the key device parameters, which are then converted to transistor sizes using the LUT-based method. The performance of the final design is validated through Spectre simulation of the sized OTA circuit for each topology. Additionally, we ensure the optimized devices operate in the desired region of operation.

5T-OTA The 5T-OTA topology includes a matched active current-mirror load (M1/M2), a differential pair (M3/M4), and a tail transistor (M5), all requiring precise sizing to meet performance targets. We assess the prediction accuracy of the transformer by correlating predicted device parameters with SPICE-based validation results. Fig. 7

Table II: Correlation coefficient of device parameters between validation data and model outputs for the 5T-OTA.

MOS devices	Transistor information	Correlation coefficient			
		g_m	g_{ds}	C_{ds}	C_{gs}
M1/M2	Active load	0.982	0.993	0.962	0.964
M3/M4	DP	0.999	0.991	0.997	0.998
M5	Tail MOS	0.999	0.997	0.997	0.997

Table III: Comparison of optimized design performance with target specifications for the 5T-OTA

Gain (dB)		UGF (MHz)		3dB bandwidth (MHz)	
Target	Optimized	Target	Optimized	Target	Optimized
20.13	20.6	118.78	144.64	11.38	13.33
21.23	21.37	181.25	185.38	15.31	15.49
22.78	22.79	281.75	288.54	20.18	20.48

Table IV: Correlation coefficient of device parameters between validation data and model outputs for the CM-OTA.

MOS devices	Transistor information	Correlation coefficient			
		g_m	g_{ds}	C_{ds}	C_{gs}
M1/M2	Matched CM load	0.811	0.838	0.871	0.875
M3/M4	DP	0.798	0.683	0.878	0.883
M5	Tail MOS	0.817	0.867	0.601	0.760
M6/M7	Matched CM load	0.893	0.803	0.881	0.895
M8/M9	Matched CM load	0.912	0.914	0.891	0.892

Table V: Comparison of optimized design performance with target specifications for the CM-OTA

Gain (dB)		UGF (MHz)		3dB bandwidth (MHz)	
Target	Optimized	Target	Optimized	Target	Optimized
20.83	21.99	345.9	475.74	30.84	37.65
21.55	23.25	247.98	408.11	20.15	27.48
23.8	24.3	1033.77	1478.5	71.47	104.24

shows a strong correlation between predicted g_m and g_{ds} values and their SPICE counterparts along the 45° line, and Table II summarizes the correlation coefficients of all the parameters, highlighting model accuracy. We show the results of applying the transformer model for three sets of unseen target specifications in Table III: the optimized circuit can be seen to meet all requirements.

CM-OTA The CM-OTA topology incorporates a differential input stage, succeeded by three current mirror loads. A total of nine devices require sizing in this configuration. The correlation coefficient between the device parameters predicted by the transformer model and the SPICE-based validation data are shown in Table IV and display high accuracy. Finally, Table V delineates the target specifications for three randomly selected designs from the validation set. As in the case of the 5T-OTA, the output of the transformer yields optimized circuits that meet all performance requirements.

2S-OTA The 2S-OTA topology includes a 5T-OTA in the first stage, followed by a common source amplifier comprising seven devices. Table VI provides a summary of the correlation coefficient between the device parameters predicted by the transformer model and those generated by SPICE, thereby affirming the accuracy of the model. Furthermore, Table VII presents the target specifications for three randomly selected designs from the validation set. Again, the transformer delivers optimized circuits that meet all specifications.

From the correlation coefficient analysis, we observe that in some cases the coefficients can be relatively lower (e.g., <0.8). These cases correspond to scenarios where the corresponding parameter does not impact the performance metrics significantly, while the other parameter has a more dominant influence. This behavior is attributed to the attention mechanism of the transformer which weights the importance of different parameters based on their level of influence

Table VI: Correlation coefficient of device parameters between validation data and model outputs for the 2S-OTA.

MOS devices	Transistor information	Correlation coefficient			
		g_m	g_{ds}	C_{ds}	C_{gs}
M1/M2	1st stage active load	0.942	0.936	0.876	0.879
M3/M4	1st stage DP	0.988	0.945	0.913	0.915
M5	1st stage tail MOS	0.928	0.989	0.918	0.922
M6	2nd stage tail MOS	0.856	0.881	0.843	0.798
M7	2nd stage CS	0.892	0.887	0.785	0.880

Table VII: Comparison of optimized design performance with target specifications for the 2S-OTA

Gain (dB)		UGF (MHz)		3dB bandwidth (kHz)	
Target	Optimized	Target	Optimized	Target	Optimized
43.6	45.61	13.33	13.4	90	140
47.17	47.93	11.09	11.77	80	90
55.19	46.04	9.42	10.11	60	91

Table VIII: Runtime analysis of training and inferencing stages.

OTA topology	One-time training duration	Single iteration		Multiple iterations		
		#designs optimized	Average time	#designs optimized	Average time	Average #iterations
5T-OTA	8.5h	95/100	37s	5/100	111s	3
CM-OTA	22h	98/100	46s	2/100	230s	5
2S-OTA	11h	90/100	36s	10/100	180s	5

on the performance specifications. As a result, the contributions of less impactful parameters may be overshadowed, leading to a lower correlation coefficient in those specific cases.

D. Runtime analysis

Table VIII provides a detailed runtime analysis, including both the one-time SPICE-based training duration and the average runtime per design optimization by the trained model. The reported runtime encompasses the entire process, from sequence inference by the trained transformer model (taking approximately 0.5s per sequence) to the LUT-based estimation and subsequent SPICE simulation verification. In cases where performance criteria are not fully met due to minor prediction inaccuracies, a "copilot" mode is activated, performing iterative refinements with progressively tighter specifications to introduce a design margin that compensates for errors. This ensures that all design specifications are ultimately satisfied, typically requiring only a few additional iterations, thereby balancing model accuracy with computational efficiency to achieve reliable design convergence. The runtime ranges from just above 30s to just under four minutes, significantly lower than competing methods.

E. Qualitative comparison with prior approaches.

Table IX compares our approach for OTA sizing with prior methods, including simulated annealing (SA) [4], particle swarm optimization (PSO) [5], graph convolutional network-based RL (GCN-RL) [11], weighted expected improvement-based Bayesian optimization (WEIBO) [21], and differential evolutionary (DE) algorithm [22].

We utilize various metrics for comparison. *SPICE simulation dependency* gauges the reliance on costly simulations for convergence: lower dependence indicates greater efficiency. *Sizing accuracy* measures how well the approach satisfies all design specifications. *Runtime* reflects the time required to reach a solution, and *memory utilization* pertains to the amount of memory resources consumed during the optimization process. Our approach, using a trained transformer model with precomputed LUTs, significantly reduces SPICE simulation dependency – achieving over 90% of sizing without simulations – while improving accuracy and reducing runtime from hours to seconds, positioning it as a highly efficient solution.

Table IX: Qualitative comparison with prior sizing approaches.

Sizing method	[4]	[5]	[11]	[21]	[22]	**Our approach**
Algorithm	SA	PSO	GCN + RL	WEIBO	DE	Transformer + LUT
SPICE simulation dependency	Very high	Very high	Low to moderate	High	Very high	**Very low***
Sizing accuracy	Variable	Moderate to high	High	High	High	High
Runtime	Very slow	Very slow	Moderate	Moderate	Slow	**Very fast**
Memory utilization	Moderate	Moderate	Moderate to high	High	Very high	**Moderate to low**

* >90% of sizing is performed without SPICE simulations, as shown in Table VIII.

V. Conclusion

We have introduced a fully automated rapid-sizing tool for OTA circuits that utilizes a transformer-based attention mechanism. Our framework successfully meets stringent design specifications across multiple unseen designs for three distinct OTA topologies, achieving a success rate exceeding 90% of the designs on the first attempt. Unlike much prior research, our method eliminates the need for extensive and costly SPICE simulations for each design, enhancing computational efficiency and accelerating the design process for OTA circuits.

REFERENCES

[1] R. Harjani, R. Rutenbar, and L. Carley, "OASYS: A framework for analog circuit synthesis," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 8, no. 12, pp. 1247–1266, Dec. 1989.

[2] J. R. Koza, F. H. Bennett, D. Andre, and M. A. Keane, "Automated design of both the topology and sizing of analog electrical circuits using genetic programming," in *Artificial Intelligence in Design '96*, J. S. Gero and F. Sudweeks, Eds. Dordrecht, Netherlands: Springer, 1996, pp. 151–170.

[3] W. Kruiskamp and D. Leenaerts, "DARWIN: CMOS opamp synthesis by means of a genetic algorithm," in *Proceedings of the ACM/IEEE Design Automation Conference*, 1995, pp. 433–438.

[4] G. Gielen, H. Walscharts, and W. Sansen, "Analog circuit design optimization based on symbolic simulation and simulated annealing," *IEEE Journal of Solid-State Circuits*, vol. 25, no. 3, pp. 707–713, Jun. 1990.

[5] R. A. Vural and T. Yildirim, "Analog circuit sizing via swarm intelligence," *AEU – International Journal of Electronics and Communications*, vol. 66, p. 732–740, Sep. 2012.

[6] 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, Jun. 2022.

[7] I. Abel, M. Neuner, and H. E. Graeb, "A hierarchical performance equation library for basic op-amp design," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 41, no. 7, pp. 1976–1989, 2022.

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

[9] A. F. Budak, P. Bhansali, B. Liu, N. Sun, D. Z. Pan, and C. V. Kashyap, "DNN-Opt: An RL inspired optimization for analog circuit sizing using deep neural networks," in *Proceedings of the ACM/IEEE Design Automation Conference*, 2021, pp. 1219–1224.

[10] K. Settaluri, A. Haj-Ali, Q. Huang, K. Hakhamaneshi, and B. Nikolic, "AutoCkt: Deep reinforcement learning of analog circuit designs," in *Proceedings of the Design, Automation & Test in Europe*, 2020, pp. 490–495.

[11] H. Wang, K. Wang, J. Yang, L. Shen, N. Sun, H.-S. Lee, and S. Han, "GCN-RL circuit designer: Transferable transistor sizing with graph neural networks and reinforcement learning," in *Proceedings of the ACM/IEEE Design Automation Conference*, 2020, pp. 1–6.

[12] M. Choi, Y. Choi, K. Lee, and S. Kang, "Reinforcement learning-based analog circuit optimizer using g_m/I_D for sizing," in *Proceedings of the ACM/IEEE Design Automation Conference*, 2023.

[13] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, vol. 30, Dec. 2017, pp. 5998–6008.

[14] A. Ochoa, "A systematic approach to the analysis of general and feedback circuits and systems using signal flow graphs and driving-point impedance," *IEEE Transactions on Circuits and Systems II*, vol. 45, no. 2, pp. 187–195, Feb. 1998.

[15] H. Schmid and A. Huber, "Analysis of switched-capacitor circuits using driving-point signal-flow graphs," *Analog Integrated Circuits and Signal Processing*, vol. 96, pp. 495–507, Sep. 2018.

[16] S. J. Mason, "Feedback theory-some properties of signal flow graphs," *Proceedings of the IRE*, vol. 41, no. 9, pp. 1144–1156, 1953.

[17] H. Schmid, "HT FHNW EIT: Analog and mixed-signal circuits and signal processing," https://tube.switch.ch/channels/d206c96c.

[18] R. Sennrich, B. Haddow, and A. Birch, "Neural machine translation of rare words with subword units," in *Annual Meeting of the Association for Computational Linguistics*, Aug. 2016, pp. 1715–1725.

[19] F. Silveira, D. Flandre, and P. Jespers, "A g_m/I_D based methodology for the design of CMOS analog circuits and its application to the synthesis of a silicon-on-insulator micropower OTA," *IEEE Journal of Solid-State Circuits*, vol. 31, no. 9, pp. 1314 – 1319, Oct. 1996.

[20] P. Jespers and B. Murmann, *Systematic Design of Analog CMOS Circuits: Using Pre-Computed Lookup Tables*. Cambridge, UK: Cambridge University Press, 2017.

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

[22] B. Liu, Y. Wang, Z. Yu, L. Liu, M. Li, Z. Wang, J. Lu, and F. V. Fernández, "Analog circuit optimization system based on hybrid evolutionary algorithms," *Integration*, vol. 42, no. 2, pp. 137–148, Apr 2009.