Performance-driven Wire Sizing for Analog Integrated Circuits

YAGUANG LI and YISHUANG LIN, Texas A&M University, USA
MEGHNA MADHUSUDAN, ARVIND SHARMA, SACHIN SAPATNEKAR, and
RAMESH HARJANI, University of Minnesota, USA
JIANG HU, Texas A&M University, USA

Analog IC performance has a strong dependence on interconnect RC parasitics, which are significantly affected by wire sizes in recent technologies, where minimum-width wires have high resistance. However, performance-driven wire sizing for analog ICs has received very little research attention. In order to fill this void, we develop several techniques to facilitate an end-to-end automatic wire sizing approach. They include a circuit performance model based on customized graph neural network (GNN) and two optimization techniques: one using Bayesian optimization accelerated by the GNN model, and the other based on TensorFlow training. Experimental results show that our technique can achieve 11% circuit performance improvement or 8.7× speedup compared to a conventional Bayesian optimization method.

CCS Concepts: • **Hardware** → **Analog and mixed-signal circuit optimization**; **Software tools for EDA**; Wire routing;

Additional Key Words and Phrases: Machine learning, analog circuit design automation, wire sizing

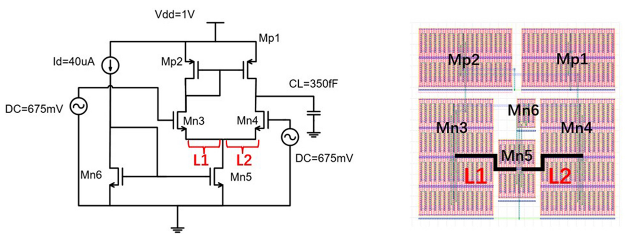
ACM Reference format:
Yaguang Li, Yishuang Lin, Meghna Madhusudan, Arvind Sharma, Sachin Sapatnekar, Ramesh Harjani, and Jiang Hu. 2022. Performance-driven Wire Sizing for Analog Integrated Circuits. *ACM Trans. Des. Autom. Electron. Syst.* 28, 2, Article 19 (December 2022), 23 pages.
https://doi.org/10.1145/3559542

1 INTRODUCTION

Analog IC performance is sensitive to layout RC parasitic, and this is why performance degradation is easily seen from a schematic design to its post-layout simulation. It is observed [27] that such layout-induced performance degradation becomes increasingly significant at advanced technology nodes. In manual layout designs, such degradation is addressed by designers through simulation-based diagnosis and layout iterations. Automatic analog layout tools attempt to mitigate the degradation by enforcing geometric [24, 30, 46] or parasitic constraints [11] during placement and routing [8]. Since analog circuit behavior is very complex, such simple constraints are either inadequate or overly tight so that satisfying performance specification remains a challenge.

Authors' addresses: Y. Li, Y. Lin, and J. Hu, Texas A&M University: Wisenbaker Engineering Building 3128, 188 Bizzell St, College Station, TX 77843; emails: {liyg, lionlin, jianghu}@tamu.edu; M. Madhusudan, A. Sharma, S. Sapatnekar, and R. Harjani, University of Minnesota: 4-174 Keller Hall, 200 Union St. SE, Minneapolis, MN 55455; emails: {madhu028, aksharma, sachin, harjani}@umn.edu.

1084-4309/2022/12-ART19 $15.00
https://doi.org/10.1145/3559542



	Schematic	Wire Width of L1 & L2		
		1X	2X	3X
Gain (dB)	26.2	23.0	24.5	25.0
UGF (MHz)	1223.0	688.5	881.4	972.0
BW (MHz)	60.0	48.1	52.6	54.5
PM (°)	90.45	92.2	91.4	91.1
Power (μW)	179.6	178.0	178.8	179.1

Fig. 1. Impact of wire widths on the performance of an OTA design. UGF: Unity Gain Frequency; BW: Bandwidth; PM: Phase Margin.

Characteristic	Specification	No sizing	Wire sizing	Transistor sizing	WS + TS
Gain (dB)	40	34.75	37.23	34.17	35.37
UGF (GHz)	4.5	1.19	1.52	3.12	4.18
BW (MHz)	65	18.48	16.74	37.02	41.70
PM (°)	45	76.09	76.09	55.87	46.23
FOM	1.00	0.52	0.58	0.69	0.79

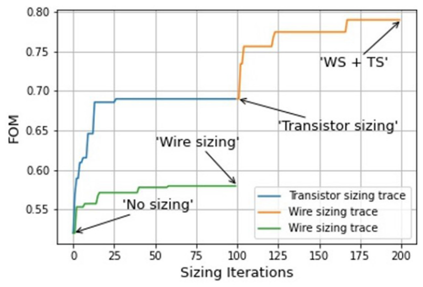


Fig. 2. Effect of wire sizing along with transistor sizing for a two-stage OTA. FOM: Figure of Merit, ideally 1.

This is a main reason there are still performance gaps between manual designs and automatic designs.

Although the first-order effects of RC parasitics are determined by the wirelength that results from placement and routing, changing wire width, a.k.a. wire sizing, can exert a significant impact to circuit performance, particularly in modern technologies (FinFET and beyond), where the design is wire resistance constrained. This impact is illustrated through an example of an **Operational Transconductance Amplifier (OTA)** in Figure 1, built in a 7nm technology. The picture at the left shows the schematic after transistor sizing, and the middle shows its layout. When the wire widths of L1 and L2 change from 1X to 3X of the minimum wire width, the **Unity Gain Frequency (UGF)** increases from 688.5MHz to 972.0MHz, and the gain increases from 23dB to 25db. This is because wire sizing reduces the wire resistance in series with the transistors, due to which the effective transconductance (Gm) of the differential pair (Mn3/Mn4) increases as the resistances of L1 and L2 decrease, and thereby improves the UGF. The effect of wire sizing is still significant even when transistor sizing has been performed, because the performance bottleneck is caused by the large wire resistance. In Figure 2, we compare the post-layout performance of four solutions: (1) no sizing, (2) wire sizing only, (3) transistor sizing only, and (4) **transistor sizing + wire sizing (TS+WS)**. Both the transistor sizing and wire sizing are achieved through Bayesian optimization. The overall performance is reflected by a composite **Figure of Merit (FOM)** with 1.00 being its ideal value. The curves on the right show that wire sizing alone improves FOM by 11.5%. If the wire sizing is performed after transistor sizing, it can improve FOM by 19%. Although transistor sizing is a more powerful technique, wire sizing brings additional significant benefit on top of it.

In analog design automation, wire sizing has been studied for addressing electromigration [22, 23, 38] and IR-drop [42]. In digital designs, performance (timing/power)-driven wire sizing was once a very active research subject [36]. These approaches are largely facilitated by the availability of analytical models, for example, the Black's equation [38] for electromigration and the Elmore delay model [9, 13, 36] for digital circuit timing. There has been little research on

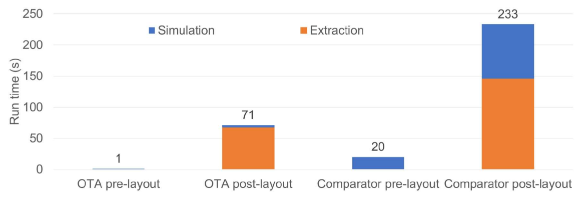


Fig. 3. Performance evaluation runtime for an OTA design and a comparator design.

performance-driven wire sizing for analog ICs largely due to the lack of a fast yet credible performance model. Usually, analog circuit performance is evaluated through circuit simulation, which is too time consuming for frequent use in optimizations. This is particularly true for layout designs, where the number of circuit elements is substantially more than schematic designs.

A similar but better-studied problem is analog transistor sizing. A variety of optimization techniques for analog transistor sizing have been proposed [3, 4, 14, 15, 25, 28, 29, 31–33, 40, 43, 50], including simulated annealing [31], evolutionary algorithms [25], gradient-based local search [33], and Bayesian optimization [29]. Recently, a reinforcement learning approach [40] was also explored. Most of these previous techniques rely on time-consuming circuit simulations. To accelerate the optimizations, surrogate performance models have been developed, including polynomial models [28, 43], **Support Vector Machine (SVM)** [14], and Gaussian-process-based models [32]. However, wire sizing faces additional difficulties compared to transistor sizing. The key difference is that transistor sizing is often performed for schematic designs, while wire sizing must consider actual routing and wire parasitics. Post-layout circuit performance evaluation is typically much slower than schematic level due to the extra time on parasitic extraction and significantly increased circuit elements after layout. The examples in Figure 3 show that the runtime difference can be as much as one order of magnitude. Hence, the budget for simulation-based circuit performance evaluation in wire sizing is usually much tighter than transistor sizing.

There are several analog transistor sizing methods that consider the effect of layout parasitics [5, 17, 19, 21, 26, 34]. Layout parasitics are estimated through templates in [5] and schematic-level RC annotation is utilized in [19]. Both of the approaches [5, 19] are difficult to cover a wide range of scenarios. In [17, 21], layout parasitics are considered for transistor sizing. Constraint-based layout tools are embedded in the sizing loop [17], while [21] employs a floorplaner. However, both [17] and [21] still require the use of expensive parasitic extraction and post-layout simulation. A linear approximation technique is proposed in [34] but tends to be inaccurate. In [26], an RC prediction technique is developed, yet expensive simulations are still needed. While these techniques [5, 17, 19, 21, 26, 34] are valuable for considering the layout effect in transistor sizing, the considerations are either too simplified or incomplete for wire sizing, which is carried out in a late step of layout design. There are parasitic-aware analog layout techniques [16, 35, 48]. In [48], capacitance sensitivity is considered during placement, while signal coupling is reduced in routing [16]. The work of [35] prioritizes resistance-sensitive nets during routing. In [10], routing wire resistance is minimized through balancing the number of layer changes and routing wire length. The work of [47] proposes wire detouring techniques to deal with parasitic mismatching. However, none of these works directly address circuit performance.

In this work, both performance modeling and optimization techniques are studied for performance-driven analog wire sizing, which is the first work on this subject. The proposed wire sizing techniques can be incorporated with either automatic or manual analog layout design where transistor sizing has already been performed. Our approach is a general framework that is applicable to a variety of different types of analog IC designs. Different performance metrics are combined into an FOM as in other works [40]. The effect of wire sizing and the effectiveness of our techniques are demonstrated on four different types of circuits: OTA, comparator, **Voltage Controlled Oscillator (VCO)**, and **Switched Capacitor Filter (SCF)**. The contributions of this work are summarized as follows:

- A customized **Graph Neural Network (GNN)**-based analog circuit performance model, called **Wire Attention Graph Network (WAGN)**, is developed. WAGN can improve the true-positive rate from 77.1% to 89.5% compared to a recent previous work [20] with a similar false-positive rate. It also outperforms conventional surrogate models used in transistor sizing, such as SVM.
- Two wire size optimization techniques are investigated. One is **Bayesian optimization guided by WAGN (BO-WAGN)**. The other is **TensorFlow-based optimization (TF)**. BO-WAGN is slightly faster than TF, while the implementation effort of TF is signficantly lower than BO-WAGN.
- With consideration of training cost including training data generation time and WAGN model training time, BO-WAGN with model knowledge transfer achieves either 11% circuit performance improvement with similar runtime or 8.7× speedup with similar solution quality compared to a conventional Bayesian optimization method. Our other techniques obtain similar results. Compared to automated layout without wire sizing, our techniques can improve circuit performance by 8% to 21%.
- The proposed wire sizing techniques are integrated with an open-source analog router to ensure routing completion. Our approach is complementary to constraints-based analog automation methodologies.

2 PROBLEM FORMULATION

Given a global routing solution, wire sizing is to select wire width for all nets so that the circuit performance is optimized. In modern process technologies (FinFET and beyond), lithography is performed with multiple patterning [2, 12], where discretization of device dimensions and interconnection widths are increasingly important. Therefore, we use discrete wire widths as the sizing variables. Let $\boldsymbol{s} = [s_1, s_2, \dots, s_C] \in \mathbb{Z}^C$ denote integer wire width variables for C nets.[1] The wire sizing problem can be formulated as

$$\begin{aligned} & \max_{\boldsymbol{s}} \; FOM(\boldsymbol{s}) \\ s.t. \quad & s_L \le s_i \le s_U, i = 1, 2, \dots, C \\ & s_i \in \mathbb{Z}, i = 1, 2, \dots, C, \end{aligned} \tag{1}$$

where s_L and s_U are the lower and upper bounds for wire sizes, respectively. To account for multiple performance metrics, a composite FOM is defined to assess the overall circuit performance:

$$FOM = \sum_{i=1}^{M} w_i \cdot \tilde{z}_i, \tag{2}$$

where $\tilde{z}_i \in [0, 1]$ represents relative performance, and w_i indicates weighting factors satisfying $\sum_{i=1}^{M} w_i = 1$. The relative performance $\tilde{z}_i$ is defined as

$$\tilde{z}_i = \begin{cases} \min\left(\dfrac{z_i}{\phi_i}, 1\right), & \text{for } z_i \in \Pi^+ \\ \min\left(\dfrac{\phi_i}{z_i}, 1\right), & \text{for } z_i \in \Pi^-, \end{cases} \tag{3}$$

[1] $\mathbb{Z}$ represents a set of integer scalar and $\mathbb{Z}^C$ denotes a set of C-dimensional integer vectors.

where z_i denotes the raw performance value obtained through circuit simulations, and ϕ_i implies user-defined specification. Π^+ (Π^-) is the set of performance metrics that are preferred to be greater (less) than ϕ_i, such as gain and bandwidth (delay and offset). With the transformation in Equation Equation (3), the relative performance $\tilde{z}_i$ is desired to be the greater, the better, for both Π^+ and Π^-.

3 OVERVIEW OF THE PROPOSED APPROACH

The proposed wire sizing is performed after global routing and before detailed routing in an automatic analog layout flow. It covers a circuit performance model and two optimization techniques:

- Circuit performance model: A GNN model is customized, which is called WAGN, for fast FOM prediction. Its input is a circuit graph with associated features, and the output is FOM classification. WAGN is built upon a **Pooling with Edge Attention (PEA)** network [20], which is also a customized GNN but developed for performance-driven analog placement. Both of them consist of multiple attention-pooling layers and a **Multi-Layer Perceptron (MLP)** network. However, there are two significant differences. Unlike PEA, which only includes a circuit netlist and a placement solution in its features, WAGN additionally considers a global routing solution as features. Moreover, a multi-kernel-based attention scheme is proposed in WAGN for the attention-pooling layer, which is an enhancement over the linear-function-based attention scheme in PEA. Details for WAGN are elaborated in Section 4.
- Wire size optimization guided by WAGN.
  - BO-WAGN. Previous works on BO-based transistor sizing are mostly guided by circuit simulations, which are notoriously slow. As the extraction/simulation cost for wire sizing is even higher, WAGN is applied for the BO-based wire sizing. Compared to conventional Bayesian optimization, it can either significantly accelerate the computation without sacrificing solution quality or attain significantly better solution quality with similar runtime cost.
  - TF. By treating wire sizes as trainable parameters instead of input features, the infrastructure of WAGN is reused for wire size optimization. The optimization is conducted using TensorFlow training with multi-start to avoid local optimal.

Each wire sizing solution is ensured to be discrete and realized in detailed routing to conform with design rules. As analog circuits are typically not as congested as digital circuits, routability is rarely an issue even with wire sizing. For the same reason, wire width increase rarely enlarges the chip area for analog circuits. It should be noted that two nets with a symmetry constraint are always assigned with the same width.

4 GNN-BASED PERFORMANCE MODEL

4.1 Notations and Background on GNN

Our analog circuit performance model is a GNN [45], which deals with problems that can be abstracted to graphs. A graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ composed by nodes $\mathcal{V}$ and edges $\mathcal{E}$ can be represented by adjacency matrix $\boldsymbol{A}$, node feature matrix $\boldsymbol{X}$, and edge feature tensor $\boldsymbol{E}$. $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ is an n by n matrix, where $n = |\mathcal{V}|$ is the number of nodes and its element $\boldsymbol{a}_{ij}$ indicates whether an edge e_{ij} exists between node v_i and node v_j. $\boldsymbol{X} \in \mathbb{R}^{n \times d}$ is an n by d matrix, and its row $\boldsymbol{X}_i \in \mathbb{R}^d$ is a d-dimension feature vector for node v_i. $\boldsymbol{E} \in \mathbb{R}^{n \times n \times p}$ is an n by n by p tensor, and its element $\boldsymbol{E}_{ij} \in \mathbb{R}^p$ is a p-dimension feature vector for edge e_{ij}. A GNN takes $\boldsymbol{A}$, $\boldsymbol{X}$, and $\boldsymbol{E}$ as inputs and outputs the class of the entire graph or the class of every node in the graph.

Attention-based Graph Convolution. A central concept in GNN is graph convolution, in which a node feature is updated by aggregating features from its neighboring nodes. Before an aggregation, usually a transformation is performed as $\boldsymbol{XW}$, where $\boldsymbol{W}$ is a trainable matrix. Then, the aggregation is formulated as $\Phi_{\boldsymbol{A}}\boldsymbol{XW}$, where $\Phi_{\boldsymbol{A}} \in \mathbb{R}^{n\times n}$ is a matrix depending on $\boldsymbol{A}$. The form of $\Phi_{\boldsymbol{A}}$ varies for different GNN techniques. A **node embedding** is generated as $\boldsymbol{Z}^{(1)} = \sigma(\Phi_{\boldsymbol{A}}\boldsymbol{XW})$, where $\sigma(\cdot)$ is an activation function. With repeated iterations of such procedure, multiple layers of node embeddings are generated as $\boldsymbol{Z}^{(0)} = \boldsymbol{X}^{(1)} = \boldsymbol{X}, \boldsymbol{Z}^{(1)} = \boldsymbol{X}^{(2)}, \boldsymbol{Z}^{(2)} = \boldsymbol{X}^{(3)}$.... More generally, a graph convolution operation is described as

$$\boldsymbol{Z}^{(l)} = \sigma\left(\Phi_{\boldsymbol{A}}^{(l)}\boldsymbol{X}^{(l)}\boldsymbol{W}^{(l)}\right), \tag{4}$$

where l is the index of layers. As feature dimension d may vary from layer to layer, the weight matrix is denoted as $\boldsymbol{W}^{(l)} \in \mathbb{R}^{d_l \times d_{l+1}}$.

The popular **Graph Attention Network (GAT)** [39] combines $\Phi_{\boldsymbol{A}}$ with the **attention** mechanism, where the attention coefficient α_{ij} from node v_j to v_i is defined by

$$\begin{aligned} \alpha_{ij} &= \text{softmax}_{\text{row}}(\tau_{ij}) = \frac{e^{\tau_{ij}}}{\sum_{k\in\mathcal{NB}_i} e^{\tau_{ik}}} \\ \tau_{ij} &= \text{LeakyReLU}\left(\boldsymbol{a}^{(l)} \cdot [(\boldsymbol{W}^{(l)^{\mathrm{T}}}\boldsymbol{X}_i^{(l)^{\mathrm{T}}})||(\boldsymbol{W}^{(l)^{\mathrm{T}}}\boldsymbol{X}_j^{(l)^{\mathrm{T}}})]\right), \end{aligned} \tag{5}$$

where $\boldsymbol{a}^{(l)} \in \mathbb{R}^{2d_{l+1}}$ is a trainable weight vector, $\boldsymbol{X}_i^{(l)}$ is the feature vector of node v_i, $\mathcal{NB}_i$ is the set of neighboring nodes of v_i, $\cdot^T$ means vector transposition, and $||$ is the vector concatenation operation. The row-wise softmax here means the index k in the denominator enumerates columns for row i. LeakyReLU$(\cdot)$ is a nonlinear function defined by

$$\text{LeakyReLU}(x) = \begin{cases} x & x \geq 0 \\ cx & x < 0, \end{cases} \tag{6}$$

where $c \in [0, 1)$ is a parameter. The attention-based **graph convolution** is described by

$$\boldsymbol{Z}^{(l)} = \sigma(\boldsymbol{\alpha}\boldsymbol{X}^{(l)}\boldsymbol{W}^{(l)}), \tag{7}$$

where $\boldsymbol{\alpha} \in \mathbb{R}^{n\times n}$ is a matrix with $\alpha_{ij}, i, j = 1, 2, \ldots, n$ as its entries.

Graph Pooling. Graph pooling, such as DiffPool [49], is to iteratively coarsen a graph through clustering such that a global view is obtained. At layer l, the number of nodes is changed from n_l to n_{l+1}, where $n_{l+1} < n_l$ and $n_0 = n$. The graph pooling operation requires an assignment matrix $\boldsymbol{S}^{(l)} \in \mathbb{R}^{n_l\times n_{l+1}}$, where each row corresponds to a node at layer l and each column indicates a cluster (new node) for layer $l+1$. This is soft clustering that the element $S_{ij}^{(l)}$ is the probability of assigning node $v_i^{(l)}$ into cluster $v_j^{(l+1)}$. The assignment matrix of layer l is defined as

$$\boldsymbol{S}^{(l)} = \text{softmax}_{\text{row}}\left[\sigma(\tilde{\boldsymbol{D}}^{(l)^{-\frac{1}{2}}}\tilde{\boldsymbol{A}}^{(l)}\tilde{\boldsymbol{D}}^{(l)^{-\frac{1}{2}}}\boldsymbol{X}^{(l)}\boldsymbol{W}_{pool}^{(l)})\right], \tag{8}$$

where $\boldsymbol{W}_{pool}^{(l)} \in \mathbb{R}^{d_l\times n_{l+1}}$ is a trainable weight matrix. In addition, $\tilde{\boldsymbol{A}}^{(l)} = \boldsymbol{A}^{(l)} + \boldsymbol{I}$, where $\boldsymbol{I}$ indicates identity matrix, and $\tilde{\boldsymbol{D}}^{(l)} \in \mathbb{R}^{n_l\times n_l}$ is a diagonal matrix, where $\tilde{D}_{ii}^{(l)} = \sum_{j=1}^{n_l}\tilde{A}_{ij}^{(l)}$. The **pooling** operation is to aggregate embedding $\boldsymbol{Z}^{(l)}$ into the next layer by

$$\boldsymbol{X}^{(l+1)} = \boldsymbol{S}^{(l)^{\mathrm{T}}}\boldsymbol{Z}^{(l)} \tag{9}$$

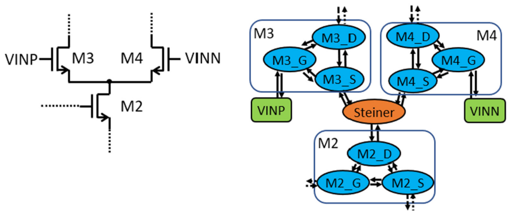


Fig. 4. A differential pair and its graph encoding.

and then perform soft clustering by

$$A^{(l+1)} = S^{(l)^{\mathrm{T}}} A^{(l)} S^{(l)}. \tag{10}$$

The pooling operation is often applied along with graph convolution at each layer.

4.2 Circuit Graph and Features

A circuit netlist and its global routing solution can be encoded into a directed graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$, where device pins, IO pins, and Steiner nodes in routing constitute graph nodes $\mathcal{V}$ and connections between nodes are indicated by graph edges $\mathcal{E}$.

The features directly related to the ith node are encoded in vector $X_i \in \mathbb{R}^d$. They include:

- Node type: PMOS, NMOS, capacitor, current source, GND, Steiner node, and so forth
- Functional module the node/pin belongs to, such as bias current mirror, differential pair, and active load
- Device dimension: width/length/number of fin of transistors
- Pin location

In the edge feature matrix, $E_{ij} \in \mathbb{R}^p, i, j = 1, 2, \ldots, n$, represents the features of the edge from node j to node i. The features related to physical routes and edge properties are encoded in edge feature vector E_{ij}. The p features include:

- Horizontal distance, vertical distance, number of vias, and wire width options between node i and node j.
- Pin length of node i and node j (the length of pins can influence the actual routing length). Pins have rectangle shapes on metal layers. The length refers to the shape length.
- Type of node i and node j, such as transistor source, drain, gate, Steiner point, and so forth. The types of node i and node j may affect the current direction of the associated edge in a circuit.

Figure 4 shows an example of encoding a differential pair into a circuit graph. The graph is directed and an edge direction indicates causality in analog circuit behaviors. For example, the voltages at $M3\_S$ and $M4\_S$ control the voltage at $M2\_D$ but not the other way around. Intuitively, as circuit performance is affected by both node features (transistor's size/dimension, etc.) and edge features (distance between two transistors' pins, etc.), we apply both of them into our WAGN's attention mechanism. In this way, information of neighboring nodes are aggregated according to their node features and connection relationships.

4.3 Wire Attention Graph Network (WAGN)

Figure 5 shows the architecture of WAGN, where multiple serially connected **attention-pooling layers** are followed by an MLP network. Our WAGN network shares the same structure with the PEA network but has a significant difference in its attention-pooling layers. Each **attention-pooling layer** consists of four steps.

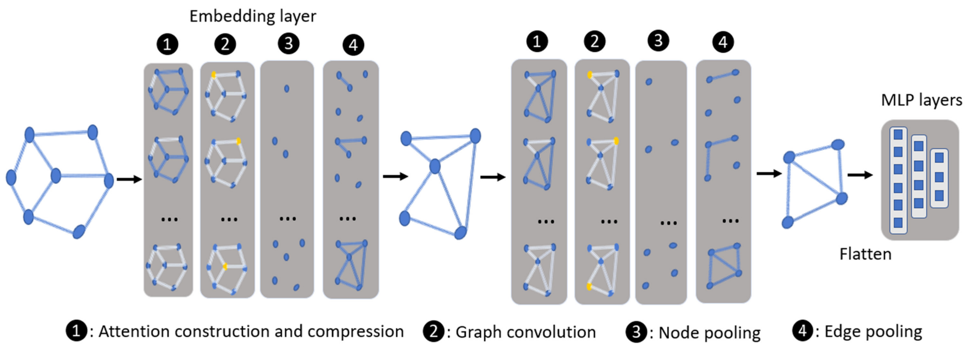


Fig. 5. WAGN architecture.

Step 1: Attention construction and compression. In [20], attention construction and compression are realized by:

- Raw attention construction defined as

$$\hat{\alpha}_{ijk}^{(l)} = \tau_{ij} E_{ijk}^{(l)}, \tag{11}$$

  where **attention coefficient** τ_{ij} is a function of node features $\boldsymbol{X}_i^{(l)}$ and $\boldsymbol{X}_j^{(l)}$, $E_{ijk}^{(l)}$ is the kth channel of edge feature $\boldsymbol{E}_{ij}^{(l)}$, and l is the layer index.
- Bidirection normalization to obtain 3D attention, which is defined as

$$\boldsymbol{\alpha}^{(l)} = \begin{cases} \tilde{\alpha}_{ijk}^{(l)} = \text{softmax}_{\text{row}}(\hat{\alpha}_{ijk}^{(l)}) \\ \alpha_{ijk}^{(l)} = \sum_{m=1}^{n_l} \frac{\tilde{\alpha}_{imk}^{(l)} \tilde{\alpha}_{jmk}^{(l)}}{\sum_{u=1}^{n_l} \tilde{\alpha}_{umk}^{(l)}}. \end{cases} \tag{12}$$

- Compressing the attention from 3D to 2D as

$$e_{ij}^{(l)} = g\left(\boldsymbol{\alpha}_{ij}^{(l)}; \boldsymbol{b}^{(l)}\right) = \sum_{k=1}^{p_l} \alpha_{ijk}^{(l)} b_k^{(l)}, \tag{13}$$

  where $\boldsymbol{b}^{(l)} \in \mathbb{R}^{p_l}$ is a trainable vector. In this way, vector $\boldsymbol{\alpha}_{ij}^{(l)}$ is transformed into a scalar and the 3D attention $\boldsymbol{\alpha}^{(l)}$ is compressed into a 2D matrix. This step means to reduce both runtime and memory use.

A key enhancement by WAGN is the new treatment of attention coefficient τ_{ij}, which plays a critical role in each attention-pooling layer. In PEA [20], τ_{ij} is a LeakyReLU function of a single linear kernel of node features $\boldsymbol{X}_i^{(l)}$ and $\boldsymbol{X}_j^{(l)}$. In WAGN, we propose using multiple kernel functions for the attention coefficient to capture the nonlinear connection strength between node i and j. On average, this new attention coefficient benefits WAGN with 3.0% accuracy improvement over PEA (details are demonstrated in Table 3) and more efficient knowledge transfer (demonstrated in Table 5). The new attention coefficient is defined as

$$\tau_{ij} = \text{LeakyReLU}\left(\sum_{r=1}^{3} B_r({\boldsymbol{W}^{(l)}}^{\mathrm{T}} {\boldsymbol{X}_i^{(l)}}^{\mathrm{T}}, {\boldsymbol{W}^{(l)}}^{\mathrm{T}} {\boldsymbol{X}_j^{(l)}}^{\mathrm{T}})\right), \tag{14}$$

where $B_r(\boldsymbol{x}, \boldsymbol{x}')$ is the rth kernel function and $\boldsymbol{x}$ and $\boldsymbol{x}'$ are feature vectors after linear transformations. Here we employ three kernel functions:

$$\begin{aligned} B_1(\boldsymbol{x}, \boldsymbol{x}') &= \exp(-\gamma||\boldsymbol{x} - \boldsymbol{x}'||_2^2), \\ B_2(\boldsymbol{x}, \boldsymbol{x}') &= (\boldsymbol{x}^{\mathrm{T}}\boldsymbol{x}' + c)^d, \\ B_3(\boldsymbol{x}, \boldsymbol{x}') &= \mathrm{MLP}(\boldsymbol{x}||\boldsymbol{x}'). \end{aligned} \tag{15}$$

- $B_1(\boldsymbol{x}, \boldsymbol{x}')$ is a Gaussian kernel function where γ is a hyper-parameter. This kernel can describe the difference of the input vectors.
- $B_2(\boldsymbol{x}, \boldsymbol{x}')$ is a polynomial kernel function where c is a trainable variable and d is a hyper-parameter. The inner product of two input vectors can describe their similarity and interaction.
- $B_3(\boldsymbol{x}, \boldsymbol{x}')$ is a multi-layer perceptron function. The multi-layer perceptron function is used here to capture the relationship beyond difference and similarity.

Our multi-kernel-based attention coefficient is a new contribution to GNN.

The rest of the operations in WAGN, including graph convolution, node pooling, and edge pooling, are the same as PEA. They are briefly covered here for the completeness of the description.

Step 2: Graph convolution. The graph convolution is performed as

$$\boldsymbol{Z}^{(l)} = \sigma\left(g(\boldsymbol{\alpha}^{(l)}; \boldsymbol{b}^{(l)})\boldsymbol{X}^{(l)}\boldsymbol{W}^{(l)}\right), \tag{16}$$

where $\sigma(\cdot)$ is an activation function. Node embedding $\boldsymbol{Z}^{(l)}$ is aggregated from node feature $\boldsymbol{X}^{(l)}$ with a 2D attention matrix $g(\boldsymbol{\alpha}^{(l)}; \boldsymbol{b}^{(l)})$.

Steps 3 and 4: Node and edge pooling. Through a trainable assignment matrix $\boldsymbol{S}^{(l)} \in \mathbb{R}^{n_l \times n_{l+1}}$ for layer l , new node feature matrix $\boldsymbol{X}^{(l+1)}$, adjacency matrix $\boldsymbol{A}^{(l+1)}$, and edge feature tensor $\boldsymbol{E}^{(l+1)}$ for layer $l+1$ are computed with details in Section 4.1 or [20].

By flattening the adjacency/node/edge matrix of the circuit graph in the last attention-pooling layer, a 1D vector is obtained and fed to several MLP layers. The MLP output y is the probability that the overall performance (FOM) is below a user-specified performance threshold T:

$$y = P(FOM < T), \tag{17}$$

i.e., a soft classification if the performance is unsatisfactory.

4.4 Handling Different Topologies and Model Complexity

The same type of circuit can be implemented with different topologies. For example, OTA can be realized as either cascode OTA or current mirror OTA with the same functionality. Even for the same netlist topology, global routing may result in different Steiner tree topologies for multi-pin nets. The correspondingly different graph structures can be handled by a single WAGN model. This is in contrast to the Gaussian process model in [29], which is restricted to a single topology.

Given a WAGN model with L layers, each layer is characterized by trainable weights $\boldsymbol{W}^{(l)}, \boldsymbol{W}^{(l)}_{pool}$, $\boldsymbol{W}^{(l)}_{edge}$, and $\boldsymbol{b}^{(l)}$, whose sizes depend on node feature size $d_{(l)}$, edge channel size $p_{(l)}$, and number of nodes n_l. Thus, we can identify a graph and a convolution/pooling layer in WAGN by three parameters $\{d_l, p_l, n_l\}$. Please note the parameters $\{d_l, p_l, n_l\}$ in one layer are independent of the input graph $\{d_0, p_0, n_0\}$. As such, a single WAGN model can handle graphs of different sizes and structures.

The model complexity depends on the configuration of attention-pooling layers. Given the fact that $n_l \geq n_{l+1}$, $d_l \geq d_{l+1}$, and $p_l \geq p_{l+1}$, in the l^{th} attention-pooling layer, the computation complexity is $O(n_l^2(n_l + p_l + d_l^2))$ for attention construction and compression, $O(n_l^2 d_l + n_l d_l^2)$ for graph convolution, $O(n_l^2(n_l + d_l))$ for node pooling, and $O(n_l^2(n_l p_l + p_l^2))$ for edge pooling. The overall computation complexity for the l^{th} attention-pooling layer is $O(n_l^2(n_l p_l + d_l^2 + p_l^2))$. The backward propagation and forward evaluation have the same computation complexity. The training takes much longer time than inference, as training requires multiple iterations of backward propagations, while inference is only one-pass forward evaluation.

4.5 Training of WAGN Model

Given N data samples obtained from post-layout simulations, the ith sample includes circuit graph $\mathcal{G}_i$ and label y_i, which is 0 for satisfactory performance and 1 for unsatisfactory performance. Let $\boldsymbol{\theta}$ denote trainable parameters of a WAGN model, including $\boldsymbol{W}^{(l)}, \boldsymbol{b}^{(l)}$, and others; the model can be trained by minimizing the cross-entropy for all data samples, defined as follows:

$$Loss = -\frac{1}{N}\sum_{i=1}^{N}(y_i log(\hat{y}_i) + (1 - y_i)log(1 - \hat{y}_i)) + \lambda||\boldsymbol{\theta}||_2^2, \tag{18}$$

where $\hat{y}_i$ is the model output. L_2-regularization is used here to avoid overfit and λ is the regularization parameter. After training, the model is denoted as

$$\hat{y}_i = \text{WAGN}_{\boldsymbol{\theta}_\star}(\mathcal{G}_i), \tag{19}$$

where $\text{WAGN}_{\boldsymbol{\theta}_\star}(\cdot)$ represents a WAGN model with trained parameter $\boldsymbol{\theta}_\star$ and $\hat{y}_i$ indicates the probability that sample $\mathcal{G}_i$ has unsatisfactory performance.

5 PERFORMANCE-DRIVEN WIRE SIZING OPTIMIZATION

5.1 Optimization Strategy

With a trained WAGN model $\text{WAGN}_{\boldsymbol{\theta}_\star}$, the performance-driven wire sizing problem becomes

$$\begin{aligned} &\min_{\boldsymbol{s}} \ \text{WAGN}_{\boldsymbol{\theta}_\star}(\boldsymbol{s}) \\ s.t. \quad & s_L \leq s_i \leq s_U, i = 1, 2, \ldots, C \\ & s_i \in \mathbb{Z}, i = 1, 2, \ldots, C. \end{aligned} \tag{20}$$

The decision variables are integer wire width $\boldsymbol{s}$ for all nets, which are a part of input features for the WAGN model. In guiding solution search, WAGN and circuit simulation have different tradeoffs: WAGN is fast but relatively inaccurate, while circuit simulation is accurate but slow. To integrate the strength of both, we propose a two-stage optimization strategy as follows:

- **Stage 1:** The integer constraints are relaxed and two continuous optimization methods are described in Sections 5.2 and 5.3. The fractional results are discretized through partial enumeration. If a fractional wire width is very close to its nearest integer, i.e., the difference is below a threshold, it is directly rounded to the nearest integer. For the other nets, we enumerate all combinations of rounding each up and down and evaluate the rounded solutions according to $\text{WAGN}_{\boldsymbol{\theta}_\star}(\boldsymbol{s})$. The enumeration here has limited impact on the optimization runtime for two reasons: (1) an analog circuit is typically not large compared with a digital circuit, and (2) only a portion of the nets are enumerated.
- **Stage 2:** The top few (by default 10) best solutions from stage 1 are simulated and the optimal one according to Equation (1) is selected to be the final solution.

5.2 TensorFlow Training-based Optimization

Wire size optimization Equation (20) after relaxing the integer constraints is a nonlinear programming problem. One observation is that neural network training is a process of minimizing its loss

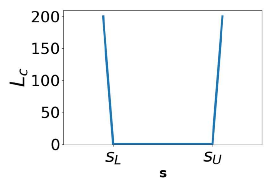


Fig. 6. Penalty function.

function in the same way as solving a nonlinear programming problem. This observation implies that the infrastructure for training $\text{WAGN}_{\theta_\star}(\boldsymbol{s})$ can be reused for continuous wire sizing. In this work, TensorFlow [1] is adopted for both training model $\text{WAGN}_{\theta_\star}(\boldsymbol{s})$ and wire sizing. The loss function for wire sizing in Tensorflow is defined as

$$\begin{aligned} L_{sizing} &= \text{WAGN}_{\theta_\star}(\boldsymbol{s}) + L_c \\ L_c &= \phi \sum_{j=1}^{C} \Big(\text{ReLU}(s_L - s_j) + \text{ReLU}(s_j - s_U)\Big) \\ \phi &\geqslant 0, \end{aligned} \tag{21}$$

where $\boldsymbol{\theta}_\star$ indicates WAGN parameters after training by Equation (18) and are fixed during wire sizing. Instead of $\boldsymbol{\theta}_\star$, $\boldsymbol{s}$ is treated as trainable parameters in minimizing L_{sizing}. Constraints $s_L \leq s_i \leq s_U, i = 1, 2, \ldots, C$ are relaxed and handled by penalty function L_c, where ϕ is an empirical penalty coefficient. Penalty function L_c is plotted in Figure 6. One can see that a huge penalty is applied once the value of s is outside of $[s_L, s_U]$. When ϕ is sufficiently large, optimized wire width solutions can be ensured in $[s_L, s_U]$. The TensorFlow training minimizes L_{sizing} and then obtains a feasible solution with the minimum $\text{WAGN}_{\theta_\star}$ value, i.e., the minimum probability of violating performance specifications.

TensorFlow minimizes Equation (21) through the gradient descent method, which is an iterative algorithm depending on the initial solution. To reduce the chance of being trapped into local minimum, we suggest a multi-start approach. That is, multiple sequences of iterations are performed with different initial solutions. The top few best solutions among all sequences are simulated and the one with the maximum *FOM* is chosen as the final continuous sizing solution.

5.3 WAGN-guided Bayesian Optimization

Bayesian optimization is for solving problems with a black box model as its objective function [37]. Through initial sampling to this black box model, a probabilistic surrogate model is constructed. In later iterations, new samples are decided by an acquisition function and the surrogate model is continuously trained. At the end, the best solution found during the iterations is returned.

The sampling in Bayesian-optimization-based transistor sizing [29] is through circuit simulation, which becomes overly expensive for evaluating wire sizing solutions. A neural network is employed for Bayesian optimization in [50] to act as a surrogate model. However, its sampling is still obtained through circuit simulations. In this work, we propose a hybrid use of WAGN and circuit simulation in Bayesian-optimization-based wire sizing and the framework is outlined in Algorithm 1. In this framework, the initial sampling (step 1) and the main sampling iterations (step 5) are through our WAGN network, which is much faster than the simulation-based sampling in [29]. Only a few top solutions obtained by step 8 are simulated to obtain the max-FOM solution with high accuracy.

In our implementation, the surrogate model is a Gaussian process with Radial Basis Function kernel. The acquisition function is decided according to Expected Improvement [18]. The

ALGORITHM 1: WAGN-guided Bayesian Optimization

```
1: Sample $N_s$ solutions and evaluate them by WAGN
2: Construct a probabilistic surrogate model from the $N_s$ samples
3: for t = 1, 2, 3, ..., $N_{iter}$ do
4:     Find $s_*$ that optimizes the acquisition function [29]
5:     Sample $y_* = \text{WAGN}_{\theta_\star}(s_*)$
6:     Update probabilistic surrogate model according to $y_*$
7: end for
8: Collect top few best solutions $S_{top}$
9: Simulate $FOM$ for all solutions in $S_{top}$
10: return the solution with the max simulated $FOM$
```

Table 1. Circuit Specifications

Design	Specifications
CC-OTA	Gain: 37.0dB, UGF:1522.9MHz, BW:21.82MHz, PM:82.10°
CM-OTA	Gain: 32.57dB, UGF:531.0MHz, BW:12.4MHz, PM: 82.82°
5T-OTA	Gain: 32.43dB, UGF:1105MHz, BW:26.45MHz, PM:86.47°
TS-OTA1	Gain: 37dB, UGF: 400MHz, BW:6.0MHz, PM:60°
TS-OTA2	Gain: 48dB, UGF: 550MHz, BW:2.5MHz, PM:60°
Comp1	Evaluation delay: 22.27ps, Precharge delay: 26.48ps, Power: 108μW, Offset: 2.60mV
Comp2	Evaluation delay: 37.28ps, Precharge delay: 17.65ps, Power: 297.5μW, Offset:2.40 mV
Comp3	Evaluation delay: 71.32ps, Precharge delay: 9.73ps, Power: 91.13μW, Offset: 2.00mV
VCO1/VCO2	Power: 31.7mW, Max Frequency: 1GHz, Min Frequency:0.38GHz
SCF1/SCF2	Gain: 17.5dB, UGF:3.3MHz, BW:1.2MHz

L-BFGS-B algorithm [6], which can handle constraints and thereby ensure that all wire width values are within $[s_L, s_U]$, is used in step 4. More details of Bayesian optimization can be found in [7]. Let $N = N_s + N_{iter}$, where N_s is the number of initial samples (line 1 of Algorithm 1) and N_{iter} is the number of iterations (line 3). The computational complexity of training the surrogate model is $O(N^3)$ (line 6) and the complexity of the surrogate model inference is $O(N^2)$ (line 4). A derivation of these complexities can be found in [50]. The computational cost of sampling (lines 1 and 5) is the WAGN model inference, and its complexity has been discussed in Section 4.4.

6 EXPERIMENTS

Our experiments are conducted on a Linux machine (64-core) using Xeon (R) E5-2680 V2 processor with 2.8GHz frequency and 256G memory. The machine learning models are implemented in Python. Our analog IC placer and router based on [41] are programmed in C++ and support symmetry, matching, and common centroid constraints. SPICE[2] simulations are performed for schematic and layout solutions. Parasitic extractions are performed using Calibre. Both the simulation time and extraction time are counted in the data sampling process. The layout generation and their post-layout simulations are conducted in parallel on the 64-core machine.

The test cases include five different OTA designs (5-transistor OTA, cascode OTA, current mirror OTA, two two-stage OTAs), three comparator designs, two VCO designs, and two SCF designs. The ASAP 7nm (ASAP7) process technology [12] and GlobalFoundries 12nm (GF12) technology are employed in the test cases. Their schematics are depicted in Figure 7 and circuit specifications are provided in Table 1. The specifications are obtained according to schematic designs so that the goal of wire sizing is to approach the schematic design performance as much as possible. For

[2]Both Hspice and Spectre are used. Spectre is used for SCF circuits for the **Periodic AC (PAC)** and **Periodic Steady-state (PSS)** analysis. Hspice is used for the simulations of the rest of the circuits.

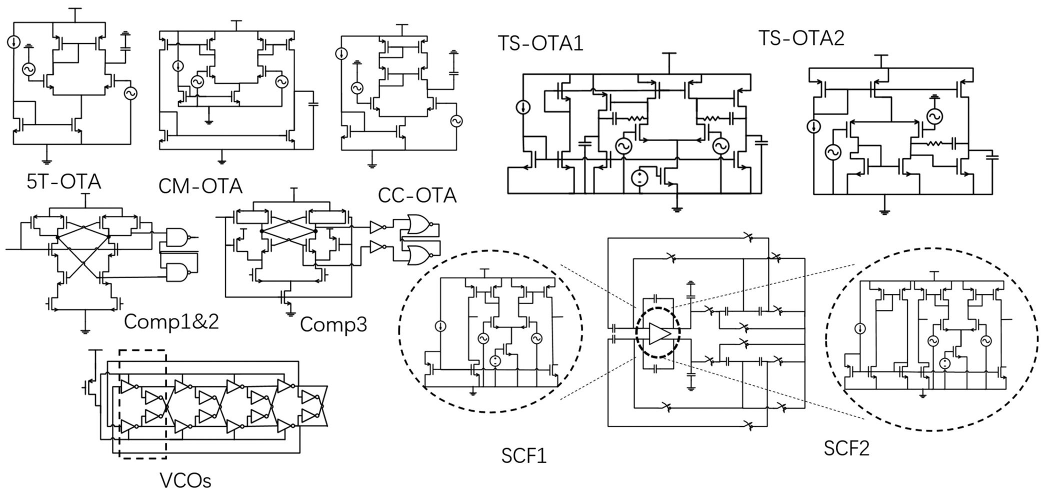


Fig. 7. Analog circuit test cases. Comp1 and Comp2 share the same topology but with different transistor sizes. SCF1 and SCF2 share the same switch structures with different opamps. VCO1 has four repeating oscillator structures, while VCO2 has six repeating structures.

Table 2. WAGN Network Configuration

Feature Extractor	# Nodes	# Node Features	# Edge Features	Predictor	# Neurons
				MLP layers	32
WAGN layers	12	11	31		16
	12	11	31		8
	6	5	12		4
	6	5	12		1

each schematic design, 2,000 layouts with different placement, routing, and wire sizes are generated by varying tool parameters and layout constraints. Among the data, 80% of the samples are for training and the other 20% are for testing so that no training data is seen in testing. During the WAGN model construction, cross-validation is performed to tune model hyperparameters. We implement the proposed WAGN model with four attention-pooling layers and five MLP layers. Table 2 summarizes the configuration of the WAGN network.

We consider discrete wire sizing in the context of gridded routing, where each net has wire width options of 1×, 2×, and 3×. According to our experience, the upper bound of 3× is high enough for sufficient performance improvement and low enough for reducing routability problems. Indeed, 99% of our wire sizing solutions are routable. In the rare event that a net is not routable with increased wire width, it can be captured by our flow and restored to its original wire width. The overall impact of wire sizing to routability is very small. Although only three width options are considered, the solution space is still huge; e.g., a circuit with only 20 nets has over 3 billion wire sizing solutions, which is immensely more than the 2,000 training samples.

The wire widths decided by a sizing solution are realized in detailed routing through parallel routes. An example with different wire width implementations is depicted in Figure 8. There are 3(2) parallel routes from Pin B to Pin A(C) to realize the 3(2)× wire widths, and design rules are well followed. The path lengths for the parallel routes might be slightly different, but the impact on performance is negligible.

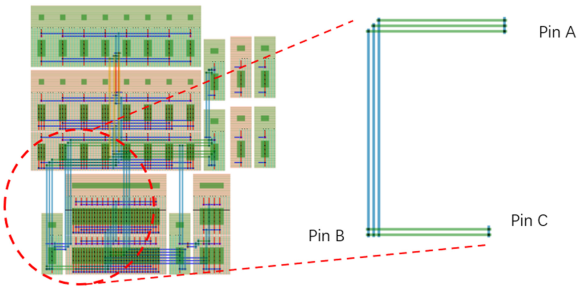


Fig. 8. Wire sizing through parallel routes.

Table 3. Comparison among Different Circuit Performance Models on Average of the 12 Test Cases (Details Are Provided in Table 12)

	Accuracy	Precision	TPR	FPR	AUROC
WAGN	94.0%	83.3%	89.5%	4.7%	0.971
PEA	91.0%	81.3%	77.1%	4.9%	0.932
SVM	85.0%	76.6%	58.7%	5.4%	0.892
RF	83.9%	76.6%	47.4%	4.5%	0.881

TPR: True-Positive Rate; FPR: False-Positive Rate; AUROC: Area under Receiver Operating Characteristic curve.

Table 4. Source-target Topology Pairs for Transfer Learning

Source	CC-OTA	CC-OTA	CM-OTA	TS-OTA1	TS-OTA2	Comp1	Comp2	Comp1	VCO1	VCO2	SCF1	SCF2
Target	CM-OTA	5T-OTA	5T-OTA	TS-OTA2	TS-OTA1	Comp2	Comp1	Comp3	VCO2	VCO1	SCF2	SCF1

6.1 Results on ML Performance Models

A classification model is evaluated by the following metrics based on **True Positive (TP)**, **True Negative (TN)**, **False Positive (FP)**, and **False Negative (FN)**.

- **Recall**, a.k.a. **True-positive Rate (TPR)**: $\frac{TP}{TP+FN}$
- **False-positive Rate (FPR)**: $\frac{FP}{FP+TN}$
- **Accuracy**: $\frac{TP+TN}{TP+TN+FP+FN}$
- **Precision**: $\frac{TP}{TP+FP}$

With different thresholds in the classification, there is a tradeoff between TPR and FPR. **Receiver Operating Characteristic (ROC)** curve shows the TPR-FPR tradeoff. **Area under the ROC curve (AUROC)** is a metric for assessing the overall performance of the entire tradeoff. AUROC is 1 if the model is perfect and 0.5 if the model performs random guesses. The proposed WAGN model is evaluated by comparisons with PEA [20], **Random Forest (RF),** and SVM.

Table 3 compares different models where training and testing data are for the same schematic topology. One can see that GNN techniques, including both WAGN and PEA, are superior to RF and SVM. Compared to PEA [20], our WAGN model improves TPR from 77.1% to 89.5% with around 5% FPR.

We also evaluate the knowledge transfer capability of different models. Here, "transfer" means that a model is trained with 80% data of a **source (S)** topology and applied to a **target (T)** topology with fine-tune training by 10% data from T. As such, the knowledge learned from S is applied (transferred) to T. Twelve S-T transfer pairs listed in Table 4 are tested. The transfer learning results are shown in Table 5, where "fine-tune-only" means the training is based on 10% of T data

Table 5. Transfer Learning Results Averaged from the 12 Circuits

	Accuracy	Precision	TPR	FPR	AUROC
WAGN transfer	90.0%	81.2%	73.9%	4.9%	0.916
WAGN fine-tune-only	86.5%	77.4%	61.0%	5.3%	0.872
PEA transfer	87.4%	79.6%	63.7%	4.9%	0.895
PEA fine-tune-only	86.3%	78.1%	59.1%	5.0%	0.886

Detailed results are provided in Table 13. TPR: True-Positive Rate; FPR: False-Positive Rate; AUROC: Area under Receiver Operating Characteristic curve.

only. Please note that “fine-tune-only” is different from WAGN/PEA, where the model is trained with 80% of T data and its inference is performed on T topology. For WAGN, the transfer leads to 12.9% improvement on TPR for a similar false-positive rate. By contrast, the TPR improvement from PEA transfer learning is only 4.6%. Please note that the knowledge transfer is restricted to be between different topologies of the same type of circuit, e.g., different topologies of OTAs. Knowledge transfer among different types of circuits is much more difficult as they often have different performance metrics, e.g., gain for OTA and linearity for ADC.

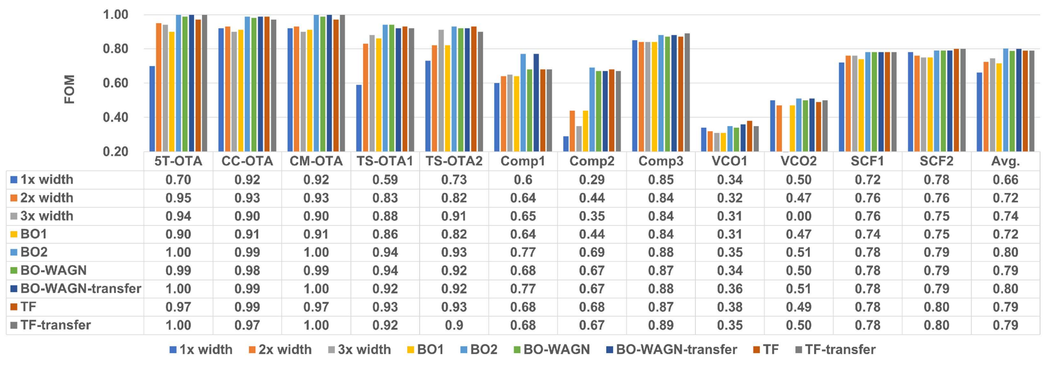
6.2 Results on Wire Sizing

There is no previous work on performance-driven analog wire sizing. Thus, we use a uniform sizing strategy and conventional Bayesian optimization, which is a recent approach to transistor sizing [29], as the main baselines for comparison. Overall, the following methods are compared:

- 1× (2×, 3×) width. All nets of a circuit have the same 1× (2×, 3×) wire width.
- **BO1.** A baseline approach of conventional Bayesian optimization [29], which is guided by circuit simulation. The number of circuit simulations is 14, which is derived in a way such that BO1 CPU runtime is similar to our techniques.
- **BO2.** This is almost the same as BO1 except that it allows 110 circuit simulation runs, which is a typical number of total samplings in conventional use.
- **BO-WAGN.** Wire sizes are optimized by our Bayesian optimization, which is guided by a hybrid of WAGN and 10 circuit simulations.
- **BO-WAGN-transfer.** Wire sizes are optimized by our Bayesian optimization, which is guided by a hybrid of WAGN with transfer learning (Section 6.1) and 10 circuit simulations.
- **TF.** Wire sizes are optimized by our TensorFlow-based sizing optimization, where WAGN and 10 circuit simulations are employed for each design.
- **TF-transfer.** Wire sizes are optimized by TensorFlow-based sizing, where WAGN with transfer learning (Section 6.1) and 10 circuit simulations are employed for each design.

Please note 3× width is realized by three parallel routes for a net, and possibly with different wire length resulting from detailed routing. Therefore, the wire area, which is the product of wire length and wire width, of 3× width can be more than 3× of the wire area of 1× width.

A comparison among different methods on post-layout circuit performance in terms of FOM is plotted in Figure 9. The best of our approach, BO-WAGN-transfer, leads to 8% to 21% average FOM improvement compared to the uniform sizing strategy (1×, 2×, 3× width). The reason BO-WAGN-transfer is better than BO-WAGN is mostly because of circuit Comp1. For Comp1, BO-WAGN-transfer finds a solution with FOM 0.77 (BO2 also finds this solution). However, BO-WAGN only finds a solution with FOM 0.68. For other circuits, BO-WAGN-transfer and BO-WAGN achieve the same or similar solutions. The others of our techniques obtain similar FOMs as BO-WAGN-transfer, with only 1% difference on average. These results confirm that uniform wire sizing is insufficient for performance improvement and our sizing techniques are indeed effective.



	5T-OTA	CC-OTA	CM-OTA	TS-OTA1	TS-OTA2	Comp1	Comp2	Comp3	VCO1	VCO2	SCF1	SCF2	Avg.
1x width	0.70	0.92	0.92	0.59	0.73	0.6	0.29	0.85	0.34	0.50	0.72	0.78	0.66
2x width	0.95	0.93	0.93	0.83	0.82	0.64	0.44	0.84	0.32	0.47	0.76	0.76	0.72
3x width	0.94	0.90	0.90	0.88	0.91	0.65	0.35	0.84	0.31	0.00	0.76	0.75	0.74
BO1	0.90	0.91	0.91	0.86	0.82	0.64	0.44	0.84	0.31	0.47	0.74	0.75	0.72
BO2	1.00	0.99	1.00	0.94	0.93	0.77	0.69	0.88	0.35	0.51	0.78	0.79	0.80
BO-WAGN	0.99	0.98	0.99	0.94	0.92	0.68	0.67	0.87	0.34	0.50	0.78	0.79	0.79
BO-WAGN-transfer	1.00	0.99	1.00	0.92	0.92	0.77	0.67	0.88	0.36	0.51	0.78	0.79	0.80
TF	0.97	0.99	0.97	0.93	0.93	0.68	0.68	0.87	0.38	0.49	0.78	0.80	0.79
TF-transfer	1.00	0.97	1.00	0.92	0.9	0.68	0.67	0.89	0.35	0.50	0.78	0.80	0.79

Fig. 9. Post-layout performance results from different wire sizing methods in terms of FOM. BO: Bayesian Optimization; TF: TensorFlow; VCO: Voltage Controlled Oscillator; SCF: Switched Capacitor Filter.

Table 6. Runtime Comparison of Different Methods

	BO1 (min)	BO2 (min)	BO-WAGN (min) WOTC	BO-WAGN (min) WTC	BO-WAGN-transfer (min)	TF (min) WOTC	TF (min) WTC	TF-transfer (min)
5T-OTA	17.1	116.8	11.2	39.5	14.6	17.1	45.4	20.0
CC-OTA	19.4	130.9	12.2	33.3	14.3	15.9	37.0	18.0
CM-OTA	18.1	124.9	11.8	28.9	13.5	12.6	29.7	14.3
TS-OTA1	68.6	510.9	48.10	208.51	59.18	54.7	221.61	72.54
TS-OTA2	48.0	363.3	33.01	134.97	44.11	35.1	141.86	46.70
Comp1	35.2	244.3	21.3	60.6	25.2	27.1	66.4	31.0
Comp2	53.8	386.2	32.4	70.0	36.2	25.8	63.4	29.6
Comp3	35.1	273.0	23.3	60.5	27.0	26.9	64.1	30.6
VCO1	128.4	934.7	88.3	262.5	105.7	132.5	306.7	150.0
VCO2	122.5	892.3	93.1	300.6	113.8	230.4	437.4	251.1
SCF1	129.3	962.0	99.0	278.4	116.9	97.3	276.3	115.2
SCF2	94.6	669.0	62.5	168.6	73.1	61.4	167.4	72.0
Avg.	64.21	467.4	44.7	137.2	53.6	61.4	154.77	70.93
Norm	**1.2×**	**8.7×**	**0.8×**	**2.6×**	**1×**	**1.1×**	**2.9×**	**1.3×**

WOTC: without considering training cost (training data generation + training time); WTC: with consideration of training cost. Training cost is also considered for both BO-WAGN-transfer and TF-transfer.

Runtime comparisons are provided in Table 6. All results are normalized with respect to that of BO-WAGN-transfer. The runtimes of BO-WAGN-transfer and TF-transfer cover the training cost of WAGN, which includes the time of training dataset generation (layout generation, parasitic extraction, and post-layout simulation) and WAGN model training. The runtime of BO-WAGN-transfer (TF-transfer) is similar to that of BO-WAGN (TF) when training cost is not considered. Thus, the runtime of BO-WAGN-transfer and TF-transfer without considering training cost is omitted in the table. Its dominating part is circuit simulations, which are performed in parallel easily. The simulations in BO1 and BO2 are difficult to be parallelized as a sample (or a new sizing solution) depends on the results of previous iterations. The parallel Bayesian optimization in [44] is for multi-objective optimization, which is different from the wire sizing here. The runtime of BO1 is similar to BO-WAGN-transfer, but its circuit performance (FOM) is 11% worse. BO2 can achieve similar circuit performance as our techniques but is 8.7× slower. According to [40], machine learning model transfer among different technology generations is not hard. Thus, our WAGN train cost

Table 7. Post-layout Results of CM-OTA

	1× Width	2× Width	3× Width	BO1	BO2	BO-WAGN	BO-WAGN-transfer	TF	TF-transfer
Gain (dB)	33.1	32.3	31.1	29.9	32.6	32.4	32.6	32.8	32.4
UGF (MHz)	451.0	470.5	511.4	517.0	516.5	524.7	507.2	520.0	513.7
BW (MHz)	10.2	11.4	14.5	16.9	12.3	12.9	13.2	12.1	12.5
PM (°)	78.5	77.7	77.6	76.9	77.4	77.0	76.4	77.5	77.3
FOM	**0.92**	**0.93**	**0.90**	**0.91**	**1.00**	**0.99**	**1.00**	**0.97**	**1.00**
Wire Area (μm^2)	0.24	0.63	0.86	0.52	0.57	0.44	0.51	0.66	0.67

Table 8. Post-layout Results of TS-OTA1

	1× Width	2× Width	3× Width	BO1	BO2	BO-WAGN	BO-WAGN-transfer	TF	TF-transfer
Gain (dB)	18.25	35.6	36.1	35.76	36.22	36.23	35.91	36.02	36.27
UGF (MHz)	88.0	283.0	324.6	302.8	364.9	364.2	350.6	357.2	357.7
BW (MHz)	10.79	4.46	4.92	4.66	5.43	5.43	5.35	5.51	5.21
PM (degree)	76.12	65.56	64.79	62.86	61.97	62.63	62.03	62.8	61.13
FOM	**0.59**	**0.83**	**0.88**	**0.86**	**0.94**	**0.94**	**0.92**	**0.93**	**0.92**
Wire Area (μm^2)	1.13	2.35	3.68	2.08	2.79	2.98	2.95	2.70	2.60

Table 9. Post-layout Results of TS-OTA2

	1× Width	2× Width	3× Width	BO1	BO2	BO-WAGN	BO-WAGN-transfer	TF	TF-transfer
Gain (dB)	36.70	42.56	46.4	41.86	47.43	46.79	46.77	47.32	46.81
UGF (MHz)	356.3	412.2	470.5	437.8	517.8	503.3	503.3	508.5	468.1
BW (MHz)	5.43	3.27	2.30	37.13	2.14	2.28	2.28	2.16	2.21
PM (°)	61.48	58.08	61.6	58.98	67.9	65.98	65.79	65.42	61.20
FOM	**0.73**	**0.82**	**0.91**	**0.82**	**0.93**	**0.92**	**0.92**	**0.93**	**0.90**
Wire Area (μm^2)	0.38	0.72	1.15	1.01	1.02	0.59	0.59	0.95	0.46

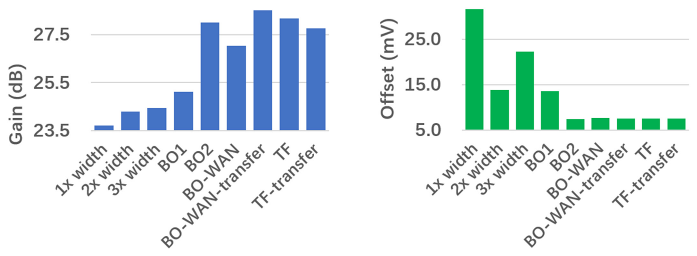
Table 10. Post-layout Results of Comp1

	1× Width	2× Width	3× Width	BO1	BO2	BO-WAGN	BO-WAGN-transfer	TF	TF-transfer
Evaluation delay (ps)	36.2	31.7	31.1	32.9	32.4	32.8	32.4	31.7	32.2
Precharge delay (ps)	45.6	43.5	40.7	44.1	41.1	41.4	41.1	43.5	41.1
Power (μW)	124.0	123.3	125.4	125.0	125.3	125.0	125.0	124.6	124.7
Offset (mV)	17.8	17.8	13.4	10.5	1.8	6.1	1.8	6.1	6.1
FOM	**0.60**	**0.64**	**0.65**	**0.64**	**0.77**	**0.68**	**0.77**	**0.68**	**0.68**
Wire Area (μm^2)	0.16	0.40	0.68	0.36	0.39	0.35	0.37	0.40	0.34

can be potentially further amortized across different technology generations. TF and BO-WAGN optimization methods can achieve about the same FOM improvement (Figure 9). While BO-WAGN is moderately faster than TF, the implementation effort of TF is lower than BO-WAGN as the same code infrastructure is used for both the model construction and optimization in TF.

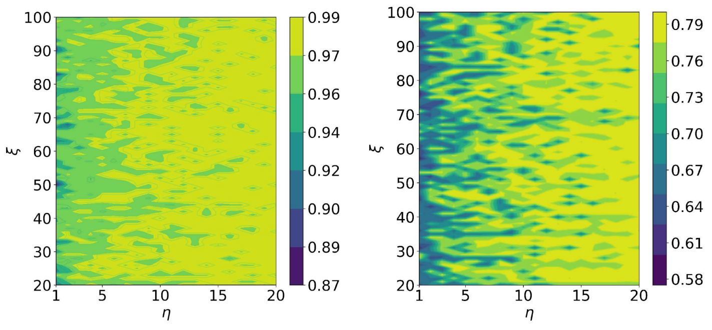
Detailed results for an OTA, two-stage OTAs, and a comparator are shown in Tables 7–10, respectively. One can observe that the wire area from wire size optimization is always smaller than that of 3× width layout and often similar to 2× width layout. Please note that wire area increase hardly affects chip area as the redundant routes use existing white space without changing device placement. We notice that wire width values in a layout after optimization are often uniformly distributed; i.e., each wire width occurs in about one-third of the nets of a circuit. This indicates that wires of different nets need to be sized differently.

The gain of CC-OTA and the offset results of Comp2 are plotted in Figure 10 for different methods. Without degradation on other performance metrics, wire sizing improves the gain by about $3dB$ for CC-OTA and reduces offset by about $7mV$ for Comp2. The plots also confirm that our techniques achieve remarkably better gain and offset than BO1.



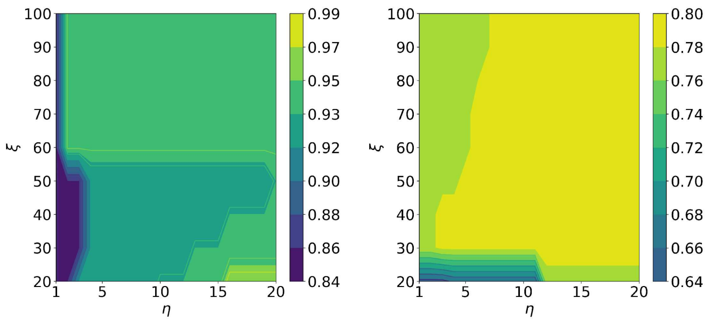
(a) CC-OTA gain (b) Comp2 offset

Fig. 10. Post-layout results of CC-OTA and Comp2.



(a) CM-OTA (b) Comp1

Fig. 11. FOM of TF with respect to ξ and η.



(a) CM-OTA (b) Comp1

Fig. 12. FOM of BO-WAGN with respect to ξ and η.

6.3 Parameter Analysis for TF- and BO-based Optimizations

Both TF- and BO-based wire size optimizations consist of multiple runs with different initial solutions, and the number of runs is denoted by ξ. At the end of TF as well as BO-WAGN, the top η solutions according to $\text{WAGN}_{\theta_\star}(\cdot)$ are simulated to find the solution with the maximum *FOM*. In our default setting $\xi = 100$ and $\eta = 10$. It is conceivable that solution quality should improve with increase of ξ and η. This effect is evaluated in a couple of circuits and the results are shown in Figures 11 and 12, where a lighter color means a higher FOM. For TF, FOM increases with η but its dependence on ξ is weak. For BO-WAGN, the effect of increasing ξ is more obvious. Interestingly, TF is sensitive to small changes of ξ and η, while BO is not. Both ξ and η are parameters whose

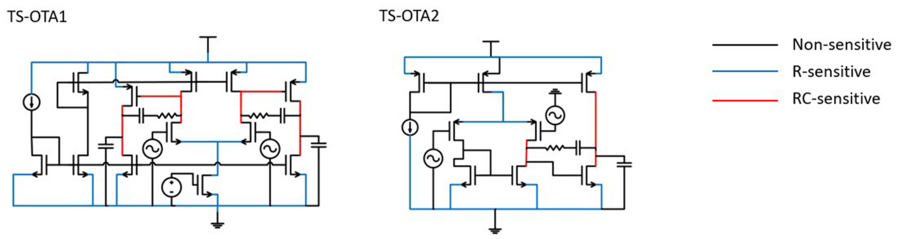


Fig. 13. Manual annotated circuits.

Table 11. Post-layout Results of TS-OTA1 and TS-OTA2

Circuit	TS-OTA1		TS-OTA2	
Method	MA	BO-WAGN	MA	BO-WAGN
Gain (dB)	36.15	36.23	35.91	46.79
UGF (MHz)	329.4	364.2	349.8	503.3
BW (MHz)	4.99	5.43	5.87	2.28
PM (degree)	63.92	63.63	61.75	64.98
FOM	**0.89**	**0.94**	**0.74**	**0.92**
Wire Area (μm^2)	2.89	2.98	1.08	0.59

values can be determined by users. Alternatively, their values can be dynamically increased during runtime till the saturation of FOM improvement.

6.4 Comparison with a Manual Annotated Method

We compare our approach with a **manual annotated (MA)** method, where the sensitivity of nets is annotated by analog designers and the wire width is sized up accordingly. As shown in Figure 13, the sensitivity of nets is annotated for two circuits, TS-OTA1 and TS-OTA2. Some nets (marked in black) are not sensitive. Some nets (marked in blue) are sensitive to R only, and the wires can be made as wide as possible. Some other nets are sensitive to both R and C (marked in red). For these nets, the wires shouldn't be too narrow (R limiting) or too wide (C limiting). In our experiment, we set 1X width for non-sensitive nets, 3X width for R-sensitive nets, and 2X width for RC-sensitive nets. Their post-layout results are shown in Table 11. We can see that our approach, BO-WAGN, is able to find better solutions than the MA method. This is not surprising, as the critical/load-sensitive nets are usually identified based on the circuit topology. However, the RC tradeoffs and their impacts on circuit performance are actually design dependent (transistor and capacitor size dependent). Such tradeoffs and impacts are hard for the designer to capture manually.

7 CONCLUSION AND FUTURE RESEARCH

Wire width can significantly affect analog IC performance, while performance-driven analog wire sizing has been rarely studied. In this work, a customized graph neural network model, WAGN, is developed for quickly estimating the performance of wire sizing solutions. Experimental results show that WAGN not only is superior to SVM and random forest but also outperforms a state-of-the-art previous work. Two wire sizing optimization techniques are investigated: one is BO guided by WAGN, and the other is based on TensorFlow training, which is also guided by WAGN. The former is slightly faster, and the latter requires significantly less implementation effort. Experimental results show that our best technique achieves either an average of 11% overall circuit performance improvement or 8.7× speedup compared to conventional Bayesian optimization.

The proposed techniques have three weaknesses. First, the effectiveness of each ML model is restricted to only one type of circuits and its knowledge cannot be easily transferred to a different type of circuit. Second, the ML models can only perform classification, while a regression model is more useful in practice. Third, the effectiveness of the ML models is difficult to scale to large circuits such as ADC/DAC. These weaknesses will be addressed in our future research. Additionally, we will further improve wire sizing techniques to account for the effect of process variations and accommodate joint transistor-wire sizing. Another future research direction will be simultaneous transistor and wire sizing.

APPENDIX

Detailed Experimental Results on Circuit Performance Models

Table 12. Comparisons of Circuit Performance Models

Circuit	WAGN	PEA	WAGN	PEA	WAGN	PEA	WAGN	PEA	WAGN	PEA
	Accuracy		Precision		Recall/TPR		FPR		AUROC	
CC-OTA	94.0%	91.5%	58.5%	53.1%	87.8%	68.9%	5.4%	5.9%	0.974	0.920
CM-OTA	97.0%	94.0%	89.9%	85.1%	98.6%	80.3%	3.5%	3.0%	0.998	0.975
5T-OTA	96.5%	94.7%	84.3%	85.3%	98.6%	95.5%	4.0%	5.6%	0.996	0.986
TS-OTA1	95.2%	92.5%	86.3%	87.1%	96.3%	82.1%	5.1%	4.0%	0.988	0.954
TS-OTA2	95.3%	93.8%	86.5%	86.4%	96.1%	86.4%	5.0%	4.0%	0.994	0.967
Comp1	92.0%	86.3%	83.0%	77.5%	82.0%	62.6%	5.0%	6.0%	0.962	0.905
Comp2	91.5%	88.0%	84.9%	79.3%	82.7%	69.7%	5.3%	6.0%	0.961	0.898
Comp3	91.8%	89.0%	83.0%	80.2%	83.8%	73.7%	5.7%	6.0%	0.954	0.920
VCO1	92.2%	87.9%	87.2%	81.5%	80.9%	70.7%	4.0%	5.8%	0.937	0.893
VCO2	92.3%	89.0%	86.8%	87.9%	81.6%	68.0%	4.2%	3.4%	0.933	0.919
SCF1	95.3%	92.4%	83.3%	85.6%	91.6%	83.8%	4.0%	4.7%	0.972	0.922
SCF2	95.0%	92.6%	86.1%	86.2%	94.4%	83.8%	4.9%	4.5%	0.983	0.931
Avg.	**94.0%**	**91.0%**	**83.3%**	**81.3%**	**89.5%**	**77.1%**	**4.7%**	**4.9%**	**0.971**	**0.932**
Circuit	SVM	RF	SVM	RF	SVM	RF	SVM	RF	SVM	RF
	Accuracy		Precision		Recall/TPR		FPR		AUROC	
CC-OTA	83.5%	82.8%	69.7%	77.9%	50.6%	43.5%	6.6%	4.1%	0.874	0.887
CM-OTA	89.0%	82.0%	79.1%	77.4%	73.6%	39.8%	6.2%	3.9%	0.853	0.751
5T-OTA	97.0%	78.0%	86.4%	66.7%	98.6%	24.0%	3.3%	4.0%	0.994	0.720
TS-OTA1	83.3%	83.0%	70.2%	72.0%	45.5%	40.9%	5.7%	4.7%	0.921	0.929
TS-OTA2	86.1%	84.5%	77.4%	75.9%	54.6%	46.9%	4.7%	4.3%	0.911	0.921
Comp1	82.5%	84.4%	67.2%	74.6%	46.1%	50.0%	6.7%	5.2%	0.859	0.883
Comp2	81.9%	84.4%	75.0%	79.7%	48.0%	50.4%	5.8%	4.3%	0.889	0.924
Comp3	83.5%	86.2%	78.0%	81.3%	46.5%	58.2%	4.3%	4.5%	0.880	0.929
VCO1	82.6%	84.1%	76.0%	72.6%	50.7%	50.0%	5.8%	5.7%	0.864	0.893
VCO2	85.1%	84.6%	76.8%	81.7%	57.8%	59.0%	5.8%	5.3%	0.916	0.912
SCF1	88.6%	84.4%	81.0%	80.0%	71.4%	50.0%	5.6%	4.2%	0.885	0.910
SCF2	87.0%	88.7%	82.4%	80.0%	61.3%	60.3%	4.4%	3.9%	0.863	0.914
Avg.	**85.8%**	**83.9%**	**76.6%**	**76.6%**	**58.7%**	**47.7%**	**5.4%**	**4.5%**	**0.892**	**0.881**

TPR: True-Positive Rate. FPR: False-Positive Rate. AUROC: Area under the ROC curve.

Table 13. Detailed Transfer Learning Results

Model	Circuit S	Circuit T	Transfer Accuracy	FTO Accuracy	Transfer Precision	FTO Precision	Transfer Recall/TPR	FTO Recall/TPR	Transfer FPR	FTO FPR	Transfer AUROC	FTO AUROC
WAGN	CC-OTA	CM-OTA	88.0%	83.6%	79.0%	75.6%	68.1%	47.2%	5.7%	4.9%	0.898	0.808
	CC-OTA	5T-OTA	94.8%	93.8%	80.5%	81.9%	93.0%	83.1%	4.9%	4.0%	0.975	0.960
	CM-OTA	5T-OTA	96.0%	93.8%	84.8%	81.9%	94.4%	83.1%	3.7%	4.0%	0.977	0.960
	TS-OTA2	TS-OTA1	95.9%	89.4%	87.1%	80.7%	97.9%	75.1%	4.8%	5.9%	0.989	0.928
	TS-OTA1	TS-OTA2	93.4%	88.5%	83.9%	80.1%	90.7%	71.2%	5.7%	5.8%	0.976	0.930
	Comp1	Comp2	85.4%	83.6%	81.5%	78.4%	58.7%	53.3%	4.9%	5.3%	0.858	0.848
	Comp2	Comp1	85.6%	86.4%	81.6%	75.7%	58.9%	59.6%	5.0%	5.7%	0.858	0.801
	Comp1	Comp3	86.8%	84.8%	79.5%	75.7%	62.6%	56.6%	5.3%	6.0%	0.899	0.885
	VCO1	VCO2	86.4%	82.9%	82.2%	77.6%	58.2%	50.7%	4.2%	5.3%	0.860	0.816
	VCO2	VCO1	86.1%	82.9%	82.1%	77.6%	61.3%	50.7%	4.9%	5.3%	0.857	0.809
	SCF1	SCF2	90.8%	84.3%	75.8%	72.6%	70.4%	50.6%	4.9%	5.7%	0.936	0.848
	SCF2	SCF1	91.3%	84.1%	76.5%	71.4%	73.2%	50.6%	4.9%	6.0%	0.911	0.876
	Avg.		90.0%	86.5%	81.2%	77.4%	73.9%	61.0%	4.9%	5.3%	0.916	0.872
PEA	CC-OTA	CM-OTA	82.8%	80.9%	75.5%	73.0%	46.3%	37.5%	5.0%	4.6%	0.870	0.803
	CC-OTA	5T-OTA	93.8%	92.0%	78.8%	76.7%	88.7%	78.9%	5.2%	5.2%	0.990	0.983
	CM-OTA	5T-OTA	94.3%	92.0%	81.6%	76.7%	87.3%	78.9%	4.3%	5.2%	0.974	0.983
	TS-OTA2	TS-OTA1	92.7%	92.2%	86.6%	85.0%	84.0%	83.8%	4.4%	5.0%	0.976	0.972
	TS-OTA1	TS-OTA2	90.0%	88.9%	83.4%	83.7%	75.3%	69.5%	5.0%	5.4%	0.958	0.949
	Comp1	Comp2	85.0%	83.5%	79.1%	76.2%	53.5%	48.5%	4.7%	5.0%	0.858	0.876
	Comp2	Comp1	84.0%	83.5%	77.8%	77.1%	49.5%	47.5%	4.7%	4.7%	0.840	0.826
	Comp1	Comp3	85.0%	84.5%	77.5%	76.1%	55.6%	54.6%	5.3%	5.7%	0.866	0.869
	VCO1	VCO2	83.9%	83.4%	77.3%	75.2%	50.4%	50.6%	5.0%	5.6%	0.816	0.811
	VCO2	VCO1	84.0%	83.7%	75.1%	76.3%	53.9%	50.6%	6.0%	5.3%	0.836	0.814
	SCF1	SCF2	86.7%	85.0%	81.2%	79.2%	60.9%	54.4%	4.7%	4.8%	0.871	0.872
	SCF2	SCF1	86.4%	85.5%	81.5%	80.8%	59.0%	55.1%	4.5%	4.4%	0.886	0.869
	Avg.		87.4%	86.3%	79.6%	78.1%	63.7%	59.1%	4.9%	5.0%	0.895	0.886

FTO: fine-tune-only.

REFERENCES

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mane, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viegas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2016. TensorFlow: Large-scale machine learning on heterogeneous distributed systems. *CoRR* abs/1603.04467 (March 2016).

[2] Robert Aitken, Greg Yeric, Brian Cline, Saurabh Sinha, Lucian Shifren, Imran Iqbal, and Vikas Chandra. 2014. Physical design and FinFETs. In *Proceedings of International Symposium on Physical Design*. 65–68.

[3] K. Antreich, J. Eckmueller, H. Graeb, M. Pronath, F. Schenkel, R. Schwencker, and S. Zizala. 2000. WiCkeD: Analog circuit synthesis incorporating mismatch. In *Proceedings of Custom Integrated Circuits Conference*. 511–514.

[4] K. J. Antreich, H. E. Graeb, and C. U. Wieser. 1994. Circuit analysis and optimization driven by worst-case distances. *IEEE Transactions on Computer-aided Design of Integrated Circuits and Systems* 13, 1 (1994), 57–71.

[5] S. Bhattacharya, N. Jangkrajarng, and C.-J. R. Shi. 2005. Template-driven parasitic-aware optimization of analog integrated circuit layouts. In *Proceedings of Design Automation Conference*. 644–647.

[6] Richard H. Byrd, Peihuang Lu, Jorge Nocedal, and Ciyou Zhu. 1995. A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing* 16, 5 (1995), 1190–1208.

[7] J. G. Carbonell and J. Siekmann. 2005. *Gaussian Processes in Machine Learning*. MIT Press.

[8] Edoardo Charbon, Enrico Malavasi, and Alberto Sangiovanni-Vincentelli. 1993. Generalized constraint generation for analog circuit design. In *Proceedings of International Conference on Computer-aided Design*. 408–414.

[9] Chung-Ping Chen, Yao-Ping Chen, and D. F. Wong. 1996. Optimal wire-sizing formula under the Elmore delay model. In *Proceedings of Design Automation Conference*. 487–490.

[10] Hao-Yu Chi, Hwa-Yi Tseng, Chien-Nan Jimmy Liu, and Hung-Ming Chen. 2018. Performance-preserved analog routing methodology via wire load reduction. In *Proceedings of Asia and South Pacific Design Automation Conference*. 482–487.

[11] Umakanta Choudhury and Alberto Sangiovanni-Vincentelli. 1993. Automatic generation of parasitic constraints for performance-constrained physical design of analog circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 12, 2 (1993), 208–224.

[12] Lawrence T. Clark, Vinay Vashishtha, Lucian Shifren, Aditya Gujja, Saurabh Sinha, Cline Brian, Ramamurthy Chandarasekaran, and Yeric Greg. 2016. ASAP7: A 7-nm FinFET predictive process design kit. *Microelectronics Journal* 53 (2016), 105–115.

[13] Jason Cong and Cheng-Kok Koh. 1994. Simultaneous driver and wire sizing for performance and power optimization. *IEEE Transactions on Very Large Scale Integration Systems* 2, 4 (1994), 408–425.

[14] F. De Bernardinis, M. I. Jordan, and A. SangiovanniVincentelli. 2003. Support vector machines for analog circuit performance representation. In *Proceedings of Design Automation Conference*. 1–6.

[15] Michael Eick and Helmut E. Graeb. 2012. MARS: Matching-driven analog sizing. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 31, 8 (2012), 1145–1158.

[16] R. S. Gyurcsik and J.-C. Jeen. 1989. A generalized approach to routing mixing analog and digital signal nets in a channel. *IEEE Journal of Solid-State Circuits* 24, 2 (1989), 436–442.

[17] Husni Habal and Helmut Graeb. 2011. Constraint-based layout-driven sizing of analog circuits. *IEEE Transactions on Computer-aided Design of Integrated Circuits and Systems* 30, 8 (2011), 1089–1102.

[18] Donald R. Jones, Matthias Schonlau, and William J. Welch. 1998. Efficient global optimization of expensive black-box functions. *Journal of Global Optimization* 13, 4 (1998), 455–492.

[19] Yaguang Li, Yishuang Lin, Meghna Madhusudan, Arvind Sharma, Sachin Sapatnekar, Ramesh Harjani, and Jiang Hu. 2021. A circuit attention network-based actor-critic learning approach to robust analog transistor sizing. In *Proceedings of Workshop on Machine Learning for CAD*. 1–6.

[20] Yaguang Li, Yishuang Lin, Meghna Madhusudan, Arvind Sharma, Wenbin Xu, Sachin S. Sapatnekar, Ramesh Harjani, and Jiang Hu. 2020. A customized graph neural network model for guiding analog IC placement. In *Proceedings of International Conference on Computer-aided Design*. 1–8.

[21] Tuotian Liao and Lihong Zhang. 2020. Efficient parasitic-aware gm/ID-based hybrid sizing methodology for analog and RF integrated circuits. *ACM Transactions on Design Automation of Electronic Systems* 26, 2 (2020), 1–31.

[22] Jens Lienig and Göran Jerke. 2003. Current-driven wire planning for electromigration avoidance in analog circuits. In *Proceedings of Asia and South Pacific Design Automation Conference*. 1–6.

[23] Jens Lienig, Göran Jerke, and Thorsten Adler. 2002. Electromigration avoidance in analog circuits: Two methodologies for current-driven routing. In *Proceedings of Asia and South Pacific Design Automation Conference*. 372–378.

[24] Mark Po-Hung Lin, Yi-Ting He, Vincent Wei-Hao Hsiao, Rong-Guey Chang, and Shuenn-Yuh Lee. 2013. Common-centroid capacitor layout generation considering device matching and parasitic minimization. *IEEE Transactions on Computer-aided Design of Integrated Circuits and Systems* 32, 7 (2013), 991–1002.

[25] Bo Liu, Yan Wang, Zhiping Yu, Leibo Liu, Miao Li, Zheng Wang, Jing Lu, and Francisco V. Fernández. 2009. Analog circuit optimization system based on hybrid evolutionary algorithms. *Integration the VLSI Journal* 42, 2 (2009), 137–148.

[26] Mingjie Liu, Walker J. Turner, George F. Kokai, Brucek Khailany, David Z. Pan, and Haoxing Ren. 2021. Parasitic-aware analog circuit sizing with graph neural networks and Bayesian optimization. In *Proceedings of Conference on Design, Automation and Test in Europe*. 1372–1377.

[27] Alvin L. S. Loke, Da Yang, Tin Tin Wee, Jonathan L. Holland, Patrick Isakanian, Kern Rim, Sam Yang, Jacob S. Schneider, Giri Nallapati, Sreeker Dundigal, Hasnain Lakdawala, Behnam Amelifard, Chulkyu Lee, Betty McGovern, Paul S. Holdaway, Xiaohua Kong, and Burton M. Leary. 2018. Analog/mixed-signal design challenges in 7-nm CMOS and beyond. In *Proceedings of Custom Integrated Circuits Conference*. 1–8.

[28] N. Lourenço, E. Afacan, R. Martins, F. Passos, A. Canelas, R. Póvoa, N. Horta, and G. Dundar. 2019. Using polynomial regression and artificial neural networks for reusable analog IC sizing. In *Proceedings of International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design*. 13–16.

[29] Wenlong Lyu, Pan Xue, Fan Yang, Changhao Yan, Zhiliang Hong, Xuan Zeng, and Dian Zhou. 2017. An efficient Bayesian optimization approach for automated optimization of analog circuits. *IEEE Transactions on Circuits and Systems I* 65, 6 (2017), 1954–1967.

[30] Qiang Ma, Linfu Xiao, Yiu-Cheong Tam, and Evangeline F. Y. Young. 2011. Simultaneous handling of symmetry, common centroid, and general placement constraints. *IEEE Transactions on Computer-aided Design of Integrated Circuits and Systems* 30, 1 (2011), 85–95.

[31] F. Medeiro-Hidalgo, R. Dominguez-Castro, A. Rodriguez-Vazquez, and J. L. Huertas. 1992. A prototype tool for optimum analog sizing using simulated annealing. In *Proceedings of International Symposium on Circuits and Systems*. 1933–1936.

[32] Oghenekarho Okobiah, Saraju P. Mohanty, and Elias Kougianos. 2014. Exploring kriging for fast and accurate design optimization of nanoscale analog circuits. In *Proceedings of Computer Society Annual Symposium on VLSI*. 244–247.

[33] Bo Peng, Fan Yang, Changhao Yan, Xuan Zeng, and Dian Zhou. 2016. Efficient multiple starting point optimization for automated analog circuit optimization via recycling simulation data. In *Proceedings of Conference on Design, Automation and Test in Europe*. 1417–1422.

[34] Almitra Pradhan and Ranga Vemuri. 2009. Efficient synthesis of a uniformly spread layout aware Pareto surface for analog circuits. In *Proceedings of International Conference on VLSI Design*. 131–136.

[35] J. Rijmenants, J. B. Litsios, T. R. Schwarz, and M. G. R. Degrauwe. 1989. ILAC: An automated layout tool for analog CMOS circuits. *IEEE Journal of Solid-State Circuits* 24, 2 (1989), 417–425.

[36] Sachin S. Sapatnekar. 1994. RC interconnect optimization under the Elmore delay model. In *Proceedings of Design Automation Conference*. 1–5.

[37] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P. Adams, and Nando de Freitas. 2015. Taking the human out of the loop: A review of Bayesian optimization. *Proceedings of the IEEE* 104, 1 (2015), 148–175.

[38] Mohammad Torabi and Lihong Zhang. 2018. Electromigration- and parasitic-aware ILP-based analog router. *IEEE Transactions on Very Large Scale Integration Systems* 26, 10 (2018), 1854–1867.

[39] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2018. Graph attention networks. In *Proceedings of International Conference on Learning Representations*. 1–12.

[40] Hanrui Wang, Kuan Wang, Jiacheng Yang, Linxiao Shen, Nan Sun, Hae-Seung Lee, and Song Han. 2020. GCN-RL circuit designer: Transferable transistor sizing with graph neural networks and reinforcement learning. In *Proceedings of Design Automation Conference*. 1–6.

[41] Laung-Terng Wang, Yao-wen Zhang, and Cheng Kwang-Ting. 2009. *Electronic Design Automation: Synthesis, Verification, and Test*. Morgan Kaufmann.

[42] Shuo-Hui Wang, Guan-Hong Liou, Yen-Yu Su, and Mark Po-Hung Lin. 2019. IR-aware power net routing for multi-voltage mixed-signal design. In *Proceedings of Conference on Design, Automation and Test in Europe*. 72–77.

[43] Ye Wang, Michael Orshansky, and Constantine Caramanis. 2014. Enabling efficient analog synthesis by coupling sparse regression and polynomial optimization. In *Proceedings of Design Automation Conference*. 1–6.

[44] Lyu Wenlong, Yang Fan, Yan Changhao, and Zeng Xuan. 2018. Batch Bayesian optimization via multi-objective acquisition ensemble for automated analog circuit design. In *Proceedings of International Conference on Machine Learning*. 3306–3314.

[45] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. 2019. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems* 1, 1 (2019), 1–21.

[46] Linfu Xiao, Evangeline F. Y. Young, Xiaoyong He, and K. P. Pun. 2010. Practical placement and routing techniques for analog circuit designs. In *Proceedings of International Conference on Computer-aided Design*. 675–679.

[47] Hailong Yao, Yici Cai, and Qiang Gao. 2012. LEMAR: A novel length matching routing algorithm for analog and mixed signal circuits. In *Proceedings of Asia and South Pacific Design Automation Conference*. 157–162.

[48] Ender Yilmaz and GÜnhan Dundar. 2009. Analog layout generator for CMOS circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 28, 1 (2009), 32–45.

[49] Rex Ying, Jiaxuan You, Christopher Morris, Xiang Ren, William L. Hamilton, and Jure Leskovec. 2018. Hierarchical graph representation learning with differentiable pooling. In *Advances in Neural Information Processing Systems*. 4800–4810.

[50] Shuhan Zhang, Wenlong Lyu, Fan Yang, Changhao Yan, Dian Zhou, and Xuan Zeng. 2019. Bayesian optimization approach for analog circuit synthesis using neural network. In *Proceedings of Conference on Design, Automation and Test in Europe*. 1463–1468.

Received 23 February 2022; revised 5 July 2022; accepted 19 August 2022