Performance-driven Wire Sizing for Analog Integrated Circuits

YAGUANG LI, Texas A&M University, USA
YISHUANG LIN, Texas A&M University, USA
MEGHNA MADHUSUDAN, University of Minnesota, USA
ARVIND SHARMA, University of Minnesota, USA
SACHIN S. S.APATNEKAR, University of Minnesota, USA
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 8.7x 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

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

Authors' addresses: Yaguang Li, liyg@tamu.edu, Texas A&M University, USA; Yishuang Lin, lionlin@tamu.edu, Texas A&M University, USA; Meghna Madhhusudan, madhu028@umn.edu, University of Minnesota, USA; Arvind Sharma, aksharma@umn.edu, University of Minnesota, USA; Sachin S. Sapatnekar, sachin@umn.edu, University of Minnesota, USA; Ramesh Harjani, harjani@umn.edu, University of Minnesota, USA; Jiang Hu, jianghu@tamu.edu, Texas A&M University, USA.
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 OTA (Operational Transconductance Amplifier) in Figure 1, built in a 7nm technology. The picture at 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 UGF (Unity Gain Frequency) 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 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 FOM (Figure of Merit) 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.

![Figure 1](image1.png)

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

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 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] is 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], SVM (Support Vector Machine) [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.

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Specification</th>
<th>No sizing</th>
<th>Wire sizing</th>
<th>Transistor sizing</th>
<th>WS + TS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gain (dB)</td>
<td>40</td>
<td>34.75</td>
<td>37.23</td>
<td>34.17</td>
<td>35.37</td>
</tr>
<tr>
<td>VGF (GHz)</td>
<td>4.5</td>
<td>1.19</td>
<td>1.52</td>
<td>3.12</td>
<td>4.18</td>
</tr>
<tr>
<td>BW (MHz)</td>
<td>65</td>
<td>18.48</td>
<td>16.74</td>
<td>37.92</td>
<td>41.70</td>
</tr>
<tr>
<td>PM (%)</td>
<td>45</td>
<td>76.09</td>
<td>76.09</td>
<td>55.87</td>
<td>46.23</td>
</tr>
<tr>
<td>FOM</td>
<td>1.00</td>
<td>0.52</td>
<td>0.58</td>
<td>0.69</td>
<td>0.79</td>
</tr>
</tbody>
</table>

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

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

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 the 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 a Figure of Merit (FOM) as in other works [40]. The effect of wire sizing as well as the effectiveness of our techniques are demonstrated on four different types of circuits: OTA, comparator, VCO (Voltage Controlled Oscillator) and SCF (Switched Capacitor Filter). The contributions of this work are summarized as follows.

- A customized GNN-based analog circuit performance model, called WAGN (Wire Attention Graph Network), is developed. WAGN can improve true positive rate from 77.1% to 89.5% compared to a recent previous work [20] with 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 significantly 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 8.7x 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%-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 \cite{2, 12}, where discretization of device dimensions and interconnection widths are increasingly important. Therefore, we use discrete wire widths as the sizing variables. Let $s = [s_1, s_2, \ldots, s_C] \in \mathbb{Z}^C$ denote integer wire width variables for $C$ nets\(^1\). The wire sizing problem can be formulated as

$$\max_s \ FOM(s)$$

subject to

$$s_L \leq s_i \leq s_U, \quad i = 1, 2, \ldots, C$$

$$s_i \in \mathbb{Z}, \quad i = 1, 2, \ldots, C$$

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

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

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

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

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

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 (Wire Attention Graph Network), for fast FOM prediction. Its input is a circuit graph with associated features, and the output is FOM classification. WAGN is built upon Pooling with Edge Attention (PEA) network \cite{20}, which is also a customized GNN but developed for performance driven analog placement. Both of them consist of multiple attention-pooling layers and an MLP (Multi-Layer Perceptron) 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.

\(^1\)\(\mathbb{Z}\) represents a set of integer scalar and $\mathbb{Z}^C$ denotes a set of $C$ dimensional integer vectors.
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.
  - WAGN-guided Bayesian optimization (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.
  - TensorFlow-based optimization (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 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 (Graph Neural Network) \[45\], which deals with problems that can be abstracted to graphs. A graph \(G(V, E)\) composed by nodes \(V\) and edges \(E\) can be represented by adjacency matrix \(A\), node feature matrix \(X\) and edge feature tensor \(E\). \(A \in \mathbb{R}^{n \times n}\) is an \(n \times n\) matrix, where \(n = |V|\) is the number of nodes and its element \(a_{ij}\) indicates whether an edge \(e_{ij}\) exists between node \(v_i\) and node \(v_j\). \(X \in \mathbb{R}^{n \times d}\) is an \(n \times d\) matrix, and its row \(X_i \in \mathbb{R}^d\) is a \(d\)-dimension feature vector for node \(v_i\). \(E \in \mathbb{R}^{n \times n \times p}\) is an \(n \times n \times p\) tensor, and its element \(E_{ij} \in \mathbb{R}^p\) is a \(p\)-dimension feature vector for edge \(e_{ij}\). A GNN takes \(A, X\) and \(E\) 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 \(XW\), where \(W\) is a trainable matrix. Then, the aggregation is formulated as \(\Phi_A XW\), where \(\Phi_A \in \mathbb{R}^{n \times n}\) is a matrix depending on \(A\). The form of \(\Phi_A\) varies for different GNN techniques. A node embedding is generated as \(Z^{(1)} = \sigma(\Phi_A XW)\), where \(\sigma(\cdot)\) is an activation function. With repeated iterations of such procedure, multiple layers of node embeddings are generated as \(Z^{(0)} = X^{(1)} = X, Z^{(1)} = X^{(2)}, Z^{(2)} = X^{(3)} \ldots\) More generally, a graph convolution operation is described as

\[
Z^{(l)} = \sigma(\Phi_A^{(l)} X^{(l)} W^{(l)})
\]
where \( l \) is the index of layers. As feature dimension \( d \) may vary from layer to layer, the weight matrix is denoted as \( \mathbf{W}^{(l)} \in \mathbb{R}^{d_l \times d_{l+1}} \).

The popular Graph Attention Network (GAT) [39] combines \( \Phi_A \) with the attention mechanism, where the attention coefficient \( \alpha_{ij} \) from node \( v_j \) to \( v_i \) is defined by

\[
\alpha_{ij} = \text{softmax}_{\text{row}}(r_{ij}) = \frac{e^{r_{ij}}}{\sum_{k \in \mathcal{N}B_j} e^{r_{ik}}}
\]

\[
r_{ij} = \text{LeakyReLU} \left( a^{(l)} \cdot [(\mathbf{W}^{(l)})^T \mathbf{X}^{(l)}] || [(\mathbf{W}^{(l)})^T \mathbf{X}^{(l)}]^T \right) \tag{5}
\]

where \( a^{(l)} \in \mathbb{R}^{2d_{l+1}} \) is a trainable weight vector, \( \mathbf{X}^{(l)} \) is the feature vector of node \( v_i \), \( \mathcal{N}B_j \) is the set of neighboring nodes of \( v_i \), \( \cdot^T \) means vector transposition and \( || \) is 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

\[
\mathbf{Z}^{(l)} = \sigma(\alpha \mathbf{X}^{(l)} \mathbf{W}^{(l)}) \tag{7}
\]

where \( \alpha \in \mathbb{R}^{n \times n} \) 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 \( \mathbf{S}^{(l)} \in \mathbb{R}^{n_l \times n_{l+1}} \), 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

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

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

\[
\mathbf{X}^{(l+1)} = \mathbf{S}^{(l)T} \mathbf{Z}^{(l)} \tag{9}
\]

and then perform soft clustering by

\[
A^{(l+1)} = \mathbf{S}^{(l)T} A^{(l)} \mathbf{S}^{(l)} \tag{10}
\]

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 $G(V, E)$, where device pins, IO pins and Steiner nodes in routing constitute graph nodes $V$ and connections between nodes are indicated by graph edges $E$.

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

- Node type: PMOS, NMOS, capacitor, current source, GND, Steiner node, etc;
- Functional module where 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, $E_{ij} \in \mathbb{R}^p, i, j = 1, 2, ..., n$, 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, etc. The types of node $i$ and node $j$ may affect the current direction of the associated edge in a circuit.

![Fig. 4. A differential pair and its graph encoding.](image)

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 neighbouring 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 PEA network, but has a significant difference in its attention-pooling layers. Each attention-pooling layer consists of four steps.

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

- Raw attention construction defined as
  \[ \hat{a}_{ijk}^{(l)} = \tau_{ij} E_{ijk}^{(l)} \]  
  where \( \tau_{ij} \) is a function of node features \( X_{i}^{(l)} \) and \( X_{j}^{(l)} \), \( E_{ijk}^{(l)} \) is the \( k \)-th channel of edge feature \( E_{ij}^{(l)} \) and \( l \) is the layer index.
- Bidirection normalization to obtain 3D attention, which is defined as
  \[
  \alpha_{ijk}^{(l)} = \begin{cases} 
  \hat{a}_{ijk}^{(l)} = \text{softmax}_{\text{row}}(\hat{a}_{ijk}^{(l)}) \\
  \sum_{m=1}^{n} \hat{a}_{imk}^{(l)} \hat{a}_{jmk}^{(l)} \
  \sum_{u=1}^{n} \hat{a}_{umk}^{(l)} 
  \end{cases} 
  \]  
  \[ \alpha_{ij}^{(l)} = \frac{\hat{a}_{ijk}^{(l)}}{\sum_{k=1}^{m} \hat{a}_{ijk}^{(l)}} \]  
  \[ \alpha_{ij}^{(l)} = \frac{\hat{a}_{ijk}^{(l)}}{\sum_{k=1}^{m} \hat{a}_{ijk}^{(l)}} \]  
  \[ \alpha_{ij}^{(l)} = \frac{\hat{a}_{ijk}^{(l)}}{\sum_{k=1}^{m} \hat{a}_{ijk}^{(l)}} \]  
  \[ \alpha_{ij}^{(l)} = \frac{\hat{a}_{ijk}^{(l)}}{\sum_{k=1}^{m} \hat{a}_{ijk}^{(l)}} \]  
  where \( \alpha_{ij}^{(l)} \) is a trainable vector. In this way, vector \( \alpha_{ij}^{(l)} \) is transformed into a scalar and the 3D attention \( \alpha_{ij}^{(l)} \) 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 $\tau_{ij}$, which plays a critical role in each attention-pooling layer. In PEA [20], $\tau_{ij}$ is a LeakyReLU function of a single linear kernel of node features $X^{(l)}_i$ and $X^{(l)}_j$. 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

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

(14)

where $B_r(x, x')$ is the $r$-th kernel function and $x$ and $x'$ are feature vectors after linear transformations.

- $B_1(x, x')$ is a Gaussian kernel function where $\gamma$ is a hyper parameter. This kernel can describe the difference of the input vectors.
- $B_2(x, x')$ is a polynomial kernel function where $c$ is trainable variable and $d$ is a hyper parameter. The inner product of two input vectors can describe their similarity and interaction.
- $B_3(x, x')$ 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 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

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

(16)

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

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

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

$$y = P(FOM < T)$$

(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 (Operational Transconductance Amplifier) 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 $W^{(l)}$, $W^{(l)}_{pool}$, $W_{edge}$ and $b^{(l)}$, whose sizes depend on node feature size $d^{(l)}$, edge channel size $p^{(l)}$, and number of node $n_l$. Thus, we can identify a graph and a convolution/pooling layer in WAGN by 3 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 $n^{(l)} \geq n^{(l+1)} + 1$, $d^{(l)} \geq d^{(l+1)} + 1$ and $p^{(l)} \geq p^{(l+1)}$, in the $l^{th}$ attention-pooling layer, the computation complexity is $O(n^{2}_l(n^{(l)} + p^{(l)} + d^{2}_l))$ for attention construction and compression, $O(n^{2}_ld^{(l)} + n^{(l)}d^{2}_l)$ for graph convolution, $O(n^{2}_l(n^{(l)} + d^{(l)}))$ for node pooling and $O(n^{2}_l(n^{(l)}p^{(l)} + p^{2}_l))$ for edge pooling. The overall computation complexity for the $l^{th}$ attention-pooling layer is $O(n^{2}_l(n^{(l)}p^{(l)} + d^{2}_l + p^{2}_l))$. 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 $i$-th sample includes circuit graph $G_i$ and label $y_i$, which is 0 for satisfactory performance and 1 for unsatisfactory performance. Let $\theta$ denote trainable parameters of a WAGN model, including $W^{(l)}$, $b^{(l)}$ and others, the model can be trained by minimizing the cross entropy for all data samples, defined as follows.

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

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

$$\hat{y}_i = \text{WAGN}_{\theta_{\star}}(G_i)$$ (19)

where WAGN$_{\theta_{\star}}(\cdot)$ represents a WAGN model with trained parameter $\theta_{\star}$ and $\hat{y}_i$ indicates the probability that sample $G_i$ has unsatisfactory performance.
5 PERFORMANCE-DRIVEN WIRE SIZING OPTIMIZATION

5.1 Optimization Strategy

With a trained WAGN model $WAGN_{\theta^*}$, the performance-driven wire sizing problem becomes

$$\min_s WAGN_{\theta^*}(s)$$

s.t. $s_L \leq s_i \leq s_U, i = 1, 2, \ldots, C$

$$s_i \in \mathbb{Z}, i = 1, 2, \ldots, C$$

(20)

The decision variables are integer wire width $s$ for all nets, which are a part of input features for the WAGN model. In guiding solution search, WAGN and circuit simulation have different tradeoff: WAGN is fast but relatively inaccurate while circuit simulation is accurate but slow. To integrate the strength of both, we propose a 2-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 $WAGN_{\theta^*}(s)$. 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 (1) is selected to be the final solution.

5.2 TensorFlow Training-Based Optimization

Wire size optimization (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 function in the same way as solving a nonlinear programming problem. This observation implies that the infrastructure for training $WAGN_{\theta^*}(s)$ can be reused for continuous wire sizing. In this work, TensorFlow [1] is adopted for both training model $WAGN_{\theta^*}(s)$ and wire sizing. The loss function for wire sizing in Tensorflow is defined as

$$L_{sizing} = WAGN_{\theta^*}(s) + L_c$$

$$L_c = \phi \sum_{j=1}^{C} \left( \text{ReLU}(s_L - s_j) + \text{ReLU}(s_j - s_U) \right)$$

(21)

where $\theta^*$ indicates WAGN parameters after training by (18) and are fixed during wire sizing. Instead of $\theta^*$, $s$ 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 $\phi$ 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 $\phi$ is sufficiently large, optimized wire width solutions can be ensured in $[s_L, s_U]$. The TensorFlow training minimizes $L_{\text{sizings}}$ and then obtains a feasible solution with the minimum $\text{WAGN}_{\theta_*}$ value, i.e., the minimum probability of violating performance specifications.

TensorFlow minimizes Equation (21) through 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.

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

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 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 Sec. 4.4.

6 EXPERIMENTS

![Fig. 7. Analog circuit testcases. Comp1 and Comp2 share the same topology but with different transistor sizes. SCF1 and SCF2 share the same switch structures width different opamps. VCO1 has 4 repeating oscillator structures while VCO2 has 6 repeating structures.](circuit_benchmark3.jpg)

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 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 testcases include five different OTA designs (5-transistor OTA, cascode OTA, current mirror OTA, two two-stage OTAs), three comparator designs, two VCO (voltage controlled oscillator) designs and two SCF (switched capacitor filter) designs. The ASAP 7nm (ASAP7) process technology [12] and GlobalFoundries

\[ ^2 \text{Both Hspice and Spectre are used. Spectre is used for SCF circuits for the PAC (Periodic AC) and PSS (Periodic Steady-State) analysis. Hspice is used for the simulations of the rest circuits.} \]
Performance-driven Wire Sizing for Analog Integrated Circuits

<table>
<thead>
<tr>
<th>Design</th>
<th>Specifications</th>
</tr>
</thead>
<tbody>
<tr>
<td>CC-OTA</td>
<td>Gain: 37.0dB, UGF:1522.9MHz, BW:21.82MHz, PM:82.10°</td>
</tr>
<tr>
<td>CM-OTA</td>
<td>Gain: 32.57dB, UGF:531.0MHz, BW:12.4MHz, PM: 82.82°</td>
</tr>
<tr>
<td>5T-OTA</td>
<td>Gain: 32.43dB, UGF:1105MHz, BW:26.45MHz, PM:86.47°</td>
</tr>
<tr>
<td>TS-OTA1</td>
<td>Gain: 37dB, UGF: 400MHz, BW:6.0MHz, PM:60°</td>
</tr>
<tr>
<td>TS-OTA2</td>
<td>Gain: 48dB, UGF: 550MHz, BW:2.5MHz, PM:60°</td>
</tr>
<tr>
<td>Comp1</td>
<td>Evaluation delay: 22.27ps, Precharge delay: 26.48ps, Power: 108µW, Offset: 2.60mV</td>
</tr>
<tr>
<td>Comp2</td>
<td>Evaluation delay: 37.28ps, Precharge delay: 17.65ps, Power: 297.5µW, Offset:2.40 mV</td>
</tr>
<tr>
<td>Comp3</td>
<td>Evaluation delay: 71.32ps, Precharge delay: 9.73ps, Power: 91.13µW, Offset: 2.00mV</td>
</tr>
<tr>
<td>VCO1/VCO2</td>
<td>Power: 31.7mW, Max Frequency: 1GHz, Min Frequency:0.38GHz</td>
</tr>
<tr>
<td>SCF1/SCF2</td>
<td>Gain: 17.5dB, UGF:3.5MHz, BW:1.2MHz</td>
</tr>
</tbody>
</table>

Table 1. Circuit specifications.

12nm (GF12) technology are employed in the testcases. 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 each schematic design, 2000 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 4 attention-pooling layers and 5 MLP layers. Table 2 summarizes the configuration of the WAGN network.

<table>
<thead>
<tr>
<th>Feature</th>
<th>#nodes</th>
<th>#node features</th>
<th>#edge features</th>
</tr>
</thead>
<tbody>
<tr>
<td>WAGN</td>
<td>12</td>
<td>11</td>
<td>31</td>
</tr>
<tr>
<td></td>
<td>12</td>
<td>11</td>
<td>31</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>5</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>5</td>
<td>12</td>
</tr>
</tbody>
</table>

Table 2. WAGN network configuration.

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 3 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 2000 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 Fig. 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.
6.1 Results on ML Performance Models

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

- **Recall**, a.k.a. TPR (True Positive Rate): \[ \frac{TP}{TP+FN} \]
- **FPR** (False Positive Rate): \[ \frac{FP}{FP+TN} \]
- **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. ROC (Receiver Operating Characteristic) curve shows the TPR-FPR tradeoff. AUROC (Area under the ROC curve) 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], RF (Random Forest) and SVM (Support Vector Machine).

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.

### Table 3. Comparison among different circuit performance models on average of the 12 testcases and details are provided in Table 12. TPR: True Positive Rate; FPR: False Positive Rate; AUROC: Area Under Receiver Operating Characteristic curve.

<table>
<thead>
<tr>
<th></th>
<th>Accuracy</th>
<th>Precision</th>
<th>TPR</th>
<th>FPR</th>
<th>AUROC</th>
</tr>
</thead>
<tbody>
<tr>
<td>WAGN</td>
<td>94.0%</td>
<td>83.3%</td>
<td>89.5%</td>
<td>4.7%</td>
<td>0.971</td>
</tr>
<tr>
<td>PEA</td>
<td>91.0%</td>
<td>81.3%</td>
<td>77.1%</td>
<td>4.9%</td>
<td>0.932</td>
</tr>
<tr>
<td>SVM</td>
<td>85.0%</td>
<td>76.6%</td>
<td>58.7%</td>
<td>5.4%</td>
<td>0.892</td>
</tr>
<tr>
<td>RF</td>
<td>83.9%</td>
<td>76.6%</td>
<td>47.4%</td>
<td>4.5%</td>
<td>0.881</td>
</tr>
</tbody>
</table>

Table 4 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.

### Table 4. Source-target topology pairs for transfer learning.

<table>
<thead>
<tr>
<th>Source</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>CC-OTA</td>
<td>CM-OTA</td>
</tr>
<tr>
<td>CC-OTA</td>
<td>CM-OTA</td>
</tr>
<tr>
<td>CM-OTA</td>
<td>TS-OTA</td>
</tr>
<tr>
<td>CM-OTA</td>
<td>TS-OTA</td>
</tr>
<tr>
<td>TS-OTA</td>
<td>TS-OTA</td>
</tr>
<tr>
<td>TS-OTA</td>
<td>Comp1</td>
</tr>
<tr>
<td>TS-OTA</td>
<td>Comp2</td>
</tr>
<tr>
<td>Comp1</td>
<td>Comp1</td>
</tr>
<tr>
<td>Comp2</td>
<td>Comp3</td>
</tr>
<tr>
<td>Comp3</td>
<td>VCO1</td>
</tr>
<tr>
<td>VCO1</td>
<td>VCO2</td>
</tr>
<tr>
<td>VCO2</td>
<td>SCF1</td>
</tr>
<tr>
<td>SCF1</td>
<td>SCF2</td>
</tr>
<tr>
<td>SCF2</td>
<td>SCF1</td>
</tr>
</tbody>
</table>
We also evaluate 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 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 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.

Table 5. Transfer learning results averaged from the 12 circuits. Detailed results are provided in Table 13. TPR: True Positive Rate; FPR: False Positive Rate; AUROC: Area Under Receiver Operating Characteristic curve.

<table>
<thead>
<tr>
<th></th>
<th>Accuracy</th>
<th>Precision</th>
<th>TPR</th>
<th>FPR</th>
<th>AUROC</th>
</tr>
</thead>
<tbody>
<tr>
<td>WAGN transfer</td>
<td>90.0%</td>
<td>81.2%</td>
<td>73.9%</td>
<td>4.9%</td>
<td>0.916</td>
</tr>
<tr>
<td>WAGN fine-tune-only</td>
<td>86.3%</td>
<td>77.4%</td>
<td>61.0%</td>
<td>5.3%</td>
<td>0.872</td>
</tr>
<tr>
<td>PEA transfer</td>
<td>87.4%</td>
<td>79.6%</td>
<td>63.7%</td>
<td>4.9%</td>
<td>0.895</td>
</tr>
<tr>
<td>PEA fine-tune-only</td>
<td>86.3%</td>
<td>78.1%</td>
<td>59.1%</td>
<td>5.8%</td>
<td>0.886</td>
</tr>
</tbody>
</table>

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 (Sec. 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 (Sec. 6.1) and 10 circuit simulations are employed for each design.
Please note 3x width is realized by three parallel routes for a net, and possibly with different wirelength resulted from detailed routing. Therefore, the wire area, which is the product of wirelength and wire width, of 3x width can be more than 3x of the wire area of 1x 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%-21% average FOM improvement compared to the uniform sizing strategy (1x, 2x, 3x width). The reason why 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.

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 8.7x slower. According to [40], machine learning model transfer among different technology generations is not hard. Thus, our WAGN train cost can
Table 6. Runtime comparison of different methods. 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.

<table>
<thead>
<tr>
<th></th>
<th>BO1 (min)</th>
<th>BO2 (min)</th>
<th>BO-WAGN (min) WOTC</th>
<th>BO-WAGN-transfer (min)</th>
<th>TF (min) WOTC</th>
<th>TF-transfer (min)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ST-OTA</td>
<td>17.1</td>
<td>116.8</td>
<td>11.2 35.9</td>
<td>14.6</td>
<td>17.1 45.4</td>
<td>20.0</td>
</tr>
<tr>
<td>CC-OTA</td>
<td>19.4</td>
<td>130.9</td>
<td>12.2 33.3</td>
<td>14.3</td>
<td>15.9 37.0</td>
<td>18.0</td>
</tr>
<tr>
<td>CM-OTA</td>
<td>18.1</td>
<td>124.9</td>
<td>11.8 28.9</td>
<td>13.5</td>
<td>12.6 29.7</td>
<td>14.3</td>
</tr>
<tr>
<td>TS-OTA1</td>
<td>68.6</td>
<td>510.9</td>
<td>48.10 208.51</td>
<td>59.18</td>
<td>54.7 221.61</td>
<td>72.54</td>
</tr>
<tr>
<td>TS-OTA2</td>
<td>48.0</td>
<td>363.3</td>
<td>33.01 134.97</td>
<td>44.11</td>
<td>35.1 141.86</td>
<td>46.70</td>
</tr>
<tr>
<td>Comp1</td>
<td>35.2</td>
<td>244.3</td>
<td>21.3 60.6</td>
<td>25.2</td>
<td>27.1 66.4</td>
<td>31.0</td>
</tr>
<tr>
<td>Comp2</td>
<td>53.8</td>
<td>386.2</td>
<td>32.4 70.0</td>
<td>36.2</td>
<td>25.8 63.4</td>
<td>29.6</td>
</tr>
<tr>
<td>Comp3</td>
<td>35.1</td>
<td>273.0</td>
<td>23.3 60.5</td>
<td>27.0</td>
<td>26.9 64.1</td>
<td>30.6</td>
</tr>
<tr>
<td>VCO1</td>
<td>128.4</td>
<td>934.7</td>
<td>88.3 262.5</td>
<td>105.7</td>
<td>132.5 306.7</td>
<td>150.0</td>
</tr>
<tr>
<td>VCO2</td>
<td>122.5</td>
<td>892.3</td>
<td>93.1 300.6</td>
<td>113.8</td>
<td>230.4 437.4</td>
<td>251.1</td>
</tr>
<tr>
<td>SCF1</td>
<td>129.3</td>
<td>962.0</td>
<td>99.0 278.4</td>
<td>116.9</td>
<td>97.3 276.3</td>
<td>115.2</td>
</tr>
<tr>
<td>SCF2</td>
<td>94.6</td>
<td>669.0</td>
<td>62.5 168.6</td>
<td>73.1</td>
<td>61.4 167.4</td>
<td>72.0</td>
</tr>
<tr>
<td>Avg.</td>
<td>64.2</td>
<td>467.4</td>
<td>44.7 137.2</td>
<td>53.6</td>
<td>61.4 154.77</td>
<td>70.93</td>
</tr>
<tr>
<td>Norm</td>
<td>1.2×</td>
<td>8.7×</td>
<td>0.8× 2.6×</td>
<td>1×</td>
<td>1.1× 2.9×</td>
<td>1.3×</td>
</tr>
</tbody>
</table>

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.

Table 7. Post-layout results of CM-OTA.

<table>
<thead>
<tr>
<th></th>
<th>1x width</th>
<th>2x width</th>
<th>3x width</th>
<th>BO1</th>
<th>BO2</th>
<th>BO-WAGN</th>
<th>BO-WAGN-transfer</th>
<th>TF</th>
<th>TF-transfer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gain (dB)</td>
<td>33.1</td>
<td>32.3</td>
<td>31.1</td>
<td>29.9</td>
<td>32.6</td>
<td>32.4</td>
<td>32.6</td>
<td>32.6</td>
<td>32.4</td>
</tr>
<tr>
<td>UGF (MHz)</td>
<td>451.0</td>
<td>470.5</td>
<td>511.4</td>
<td>517.0</td>
<td>516.5</td>
<td>524.7</td>
<td>507.2</td>
<td>520.0</td>
<td>513.7</td>
</tr>
<tr>
<td>BW (MHz)</td>
<td>10.2</td>
<td>11.4</td>
<td>14.5</td>
<td>16.9</td>
<td>12.3</td>
<td>12.9</td>
<td>13.2</td>
<td>12.1</td>
<td>12.5</td>
</tr>
<tr>
<td>PM (°)</td>
<td>78.5</td>
<td>77.7</td>
<td>77.7</td>
<td>76.9</td>
<td>77.4</td>
<td>77.6</td>
<td>76.4</td>
<td>77.5</td>
<td>77.3</td>
</tr>
<tr>
<td>FOM</td>
<td>0.92</td>
<td>0.93</td>
<td>0.90</td>
<td>0.91</td>
<td>1.00</td>
<td>0.99</td>
<td>1.00</td>
<td>0.97</td>
<td>1.00</td>
</tr>
<tr>
<td>Wire Area (µm²)</td>
<td>0.24</td>
<td>0.63</td>
<td>0.86</td>
<td>0.52</td>
<td>0.57</td>
<td>0.44</td>
<td>0.51</td>
<td>0.66</td>
<td>0.66</td>
</tr>
</tbody>
</table>

Detailed results for an OTA, two-stage OTAs and a comparator are shown in Tables 7, 8, 9 and 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 1/3 of the nets of a circuit. This indicates that wires of different nets need to be sized differently.
Table 8. Post-layout results of TS-OTA1.

<table>
<thead>
<tr>
<th>Width</th>
<th>BO1</th>
<th>BO2</th>
<th>BO-WAGN</th>
<th>BO-WAGN-transfer</th>
<th>TF</th>
<th>TF-transfer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gain (dB)</td>
<td>16.25</td>
<td>35.6</td>
<td>36.1</td>
<td>35.76</td>
<td>36.23</td>
<td>36.25</td>
</tr>
<tr>
<td>UGF (MHz)</td>
<td>88.0</td>
<td>283.6</td>
<td>324.6</td>
<td>302.8</td>
<td>364.9</td>
<td>364.2</td>
</tr>
<tr>
<td>BW (MHz)</td>
<td>10.79</td>
<td>4.46</td>
<td>4.92</td>
<td>4.66</td>
<td>5.43</td>
<td>5.43</td>
</tr>
<tr>
<td>PM (degree)</td>
<td>76.12</td>
<td>65.56</td>
<td>64.79</td>
<td>62.86</td>
<td>61.97</td>
<td>62.63</td>
</tr>
<tr>
<td>FOM</td>
<td>0.59</td>
<td>0.83</td>
<td>0.88</td>
<td>0.86</td>
<td>0.94</td>
<td>0.94</td>
</tr>
<tr>
<td>Wire Area (µm²)</td>
<td>1.13</td>
<td>2.35</td>
<td>3.68</td>
<td>2.08</td>
<td>2.79</td>
<td>2.98</td>
</tr>
</tbody>
</table>

Table 9. Post-layout results of TS-OTA2.

<table>
<thead>
<tr>
<th>Width</th>
<th>BO1</th>
<th>BO2</th>
<th>BO-WAGN</th>
<th>BO-WAGN-transfer</th>
<th>TF</th>
<th>TF-transfer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gain (dB)</td>
<td>36.70</td>
<td>42.26</td>
<td>46.4</td>
<td>41.86</td>
<td>47.43</td>
<td>46.79</td>
</tr>
<tr>
<td>UGF (MHz)</td>
<td>356.3</td>
<td>412.2</td>
<td>470.5</td>
<td>437.8</td>
<td>517.8</td>
<td>503.3</td>
</tr>
<tr>
<td>BW (MHz)</td>
<td>5.43</td>
<td>3.27</td>
<td>3.00</td>
<td>2.14</td>
<td>2.08</td>
<td>2.14</td>
</tr>
<tr>
<td>PM (degree)</td>
<td>61.48</td>
<td>58.08</td>
<td>61.6</td>
<td>58.98</td>
<td>67.9</td>
<td>65.98</td>
</tr>
<tr>
<td>FOM</td>
<td>0.73</td>
<td>0.82</td>
<td>0.91</td>
<td>0.82</td>
<td>0.93</td>
<td>0.92</td>
</tr>
<tr>
<td>Wire Area (µm²)</td>
<td>0.38</td>
<td>0.72</td>
<td>1.15</td>
<td>1.01</td>
<td>1.02</td>
<td>0.59</td>
</tr>
</tbody>
</table>

Table 10. Post-layout results of Comp1.

<table>
<thead>
<tr>
<th>Width</th>
<th>BO1</th>
<th>BO2</th>
<th>BO-WAGN</th>
<th>BO-WAGN-transfer</th>
<th>TF</th>
<th>TF-transfer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Evaluation delay (ps)</td>
<td>36.2</td>
<td>31.7</td>
<td>31.1</td>
<td>32.9</td>
<td>32.4</td>
<td>32.4</td>
</tr>
<tr>
<td>Precharge delay (ps)</td>
<td>45.6</td>
<td>43.5</td>
<td>40.7</td>
<td>44.1</td>
<td>41.1</td>
<td>41.4</td>
</tr>
<tr>
<td>Power (µW)</td>
<td>124.0</td>
<td>123.3</td>
<td>125.1</td>
<td>125.0</td>
<td>125.3</td>
<td>125.0</td>
</tr>
<tr>
<td>Offset (mV)</td>
<td>17.8</td>
<td>17.8</td>
<td>13.4</td>
<td>10.5</td>
<td>1.8</td>
<td>1.8</td>
</tr>
<tr>
<td>FOM</td>
<td>0.60</td>
<td>0.64</td>
<td>0.65</td>
<td>0.64</td>
<td>0.77</td>
<td>0.77</td>
</tr>
<tr>
<td>Wire Area (µm²)</td>
<td>0.16</td>
<td>0.40</td>
<td>0.68</td>
<td>0.36</td>
<td>0.39</td>
<td>0.35</td>
</tr>
</tbody>
</table>

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

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

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.
6.3 Parameter Analysis for TF and BO-based Optimizations

Both TF (TensorFlow) and BO (Bayesian Optimization)-based wire size optimizations consist of multiple runs with different initial solutions, and the number of runs is denoted by $\xi$. At the end of TF as well as BO-WAGN, top $\eta$ solutions according to $WAGN_{\theta^*}(\cdot)$ 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 $\xi$ and $\eta$. 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 $\eta$ but its dependence on $\xi$ is weak. For BO-WAGN, the effect of increasing $\xi$ is more obvious. Interestingly, TF is sensitive to small changes of $\xi$ and $\eta$ while BO is not. Both $\xi$ and $\eta$ are parameters whose 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 trade-offs and their impacts on circuit performance are actually design-dependent (transistor and capacitor size-dependent). Such trade-offs 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 (Wire Attention Graph Network), 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 Bayesian optimization (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 achieve 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 circuits. Second, the ML models can only perform classification while a regression model is more useful in practice. Third, the effectiveness of the ML models are 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.

REFERENCES


Performance-driven Wire Sizing for Analog Integrated Circuits


APPENDIX

Detailed Experimental Results on Circuit Performance Models

Table 12. Comparisons of circuit performance models. TPR: True Positive Rate. FPR: False Positive Rate. AUROC: Area under the ROC curve.

<table>
<thead>
<tr>
<th>Circuit</th>
<th>WAGN PEA Accuracy</th>
<th>WAGN PEA Precision</th>
<th>WAGN PEA Recall/TPR</th>
<th>WAGN PEA FPR</th>
<th>WAGN PEA AUROC</th>
</tr>
</thead>
<tbody>
<tr>
<td>CC-OTA</td>
<td>94.0%</td>
<td>91.5%</td>
<td>58.5%</td>
<td>53.1%</td>
<td>87.8%</td>
</tr>
<tr>
<td>CM-OTA</td>
<td>97.0%</td>
<td>94.0%</td>
<td>89.9%</td>
<td>85.1%</td>
<td>98.6%</td>
</tr>
<tr>
<td>ST-OTA</td>
<td>96.5%</td>
<td>94.7%</td>
<td>84.3%</td>
<td>85.3%</td>
<td>98.6%</td>
</tr>
<tr>
<td>TS-OTA1</td>
<td>95.2%</td>
<td>92.5%</td>
<td>86.3%</td>
<td>87.1%</td>
<td>96.3%</td>
</tr>
<tr>
<td>TS-OTA2</td>
<td>95.3%</td>
<td>93.8%</td>
<td>86.5%</td>
<td>86.4%</td>
<td>96.1%</td>
</tr>
<tr>
<td>Comp1</td>
<td>92.0%</td>
<td>86.3%</td>
<td>83.0%</td>
<td>77.5%</td>
<td>82.0%</td>
</tr>
<tr>
<td>Comp2</td>
<td>91.5%</td>
<td>88.0%</td>
<td>84.9%</td>
<td>79.3%</td>
<td>82.7%</td>
</tr>
<tr>
<td>Comp3</td>
<td>91.8%</td>
<td>89.0%</td>
<td>83.0%</td>
<td>80.2%</td>
<td>83.8%</td>
</tr>
<tr>
<td>VCO1</td>
<td>92.2%</td>
<td>87.9%</td>
<td>87.2%</td>
<td>81.5%</td>
<td>80.9%</td>
</tr>
<tr>
<td>VCO2</td>
<td>92.3%</td>
<td>89.0%</td>
<td>86.8%</td>
<td>87.9%</td>
<td>81.6%</td>
</tr>
<tr>
<td>SCF1</td>
<td>95.3%</td>
<td>92.4%</td>
<td>83.3%</td>
<td>85.6%</td>
<td>91.6%</td>
</tr>
<tr>
<td>SCF2</td>
<td>95.0%</td>
<td>92.6%</td>
<td>86.1%</td>
<td>86.2%</td>
<td>94.4%</td>
</tr>
<tr>
<td><strong>Avg.</strong></td>
<td><strong>94.0%</strong></td>
<td><strong>91.0%</strong></td>
<td><strong>83.3%</strong></td>
<td><strong>81.3%</strong></td>
<td><strong>89.5%</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Circuit</th>
<th>SVM RF Accuracy</th>
<th>SVM RF Precision</th>
<th>SVM RF Recall/TPR</th>
<th>SVM RF FPR</th>
<th>SVM RF AUROC</th>
</tr>
</thead>
<tbody>
<tr>
<td>CC-OTA</td>
<td>83.5%</td>
<td>82.8%</td>
<td>69.7%</td>
<td>77.9%</td>
<td>50.6%</td>
</tr>
<tr>
<td>CM-OTA</td>
<td>89.6%</td>
<td>82.0%</td>
<td>79.1%</td>
<td>77.4%</td>
<td>73.6%</td>
</tr>
<tr>
<td>ST-OTA</td>
<td>97.0%</td>
<td>87.0%</td>
<td>86.4%</td>
<td>66.7%</td>
<td>98.6%</td>
</tr>
<tr>
<td>TS-OTA1</td>
<td>83.3%</td>
<td>83.0%</td>
<td>70.2%</td>
<td>72.0%</td>
<td>45.5%</td>
</tr>
<tr>
<td>TS-OTA2</td>
<td>86.1%</td>
<td>84.5%</td>
<td>77.4%</td>
<td>75.9%</td>
<td>54.6%</td>
</tr>
<tr>
<td>Comp1</td>
<td>82.5%</td>
<td>84.4%</td>
<td>67.2%</td>
<td>74.6%</td>
<td>46.1%</td>
</tr>
<tr>
<td>Comp2</td>
<td>81.9%</td>
<td>84.4%</td>
<td>75.0%</td>
<td>79.7%</td>
<td>48.0%</td>
</tr>
<tr>
<td>Comp3</td>
<td>83.5%</td>
<td>86.2%</td>
<td>78.0%</td>
<td>81.3%</td>
<td>46.5%</td>
</tr>
<tr>
<td>VCO1</td>
<td>82.6%</td>
<td>84.1%</td>
<td>76.0%</td>
<td>72.6%</td>
<td>50.7%</td>
</tr>
<tr>
<td>VCO2</td>
<td>85.1%</td>
<td>84.6%</td>
<td>76.8%</td>
<td>81.7%</td>
<td>57.8%</td>
</tr>
<tr>
<td>SCF1</td>
<td>88.6%</td>
<td>84.4%</td>
<td>81.0%</td>
<td>80.0%</td>
<td>71.4%</td>
</tr>
<tr>
<td>SCF2</td>
<td>87.0%</td>
<td>88.7%</td>
<td>82.4%</td>
<td>80.0%</td>
<td>61.3%</td>
</tr>
<tr>
<td><strong>Avg.</strong></td>
<td><strong>85.8%</strong></td>
<td><strong>83.9%</strong></td>
<td><strong>76.6%</strong></td>
<td><strong>76.6%</strong></td>
<td><strong>58.7%</strong></td>
</tr>
</tbody>
</table>
Table 13. Detailed transfer learning results. FTO: fine-tune-only.

<table>
<thead>
<tr>
<th>Model</th>
<th>Circuit</th>
<th>Transfer FTO Accuracy</th>
<th>Transfer FTO Precision</th>
<th>Transfer FTO Recall/TPR</th>
<th>Transfer FTO FPR</th>
<th>Transfer FTO AUROC</th>
</tr>
</thead>
<tbody>
<tr>
<td>WAGN</td>
<td>CC-OTA</td>
<td>88.0% 83.6%</td>
<td>79.0% 75.6%</td>
<td>68.1% 47.2%</td>
<td>5.7% 4.9%</td>
<td>0.898 0.808</td>
</tr>
<tr>
<td></td>
<td>CM-OTA</td>
<td>94.8% 93.8%</td>
<td>80.5% 81.9%</td>
<td>93.0% 83.1%</td>
<td>4.9% 4.0%</td>
<td>0.975 0.960</td>
</tr>
<tr>
<td></td>
<td>TS-OTA2</td>
<td>90.6% 93.8%</td>
<td>84.8% 81.9%</td>
<td>94.4% 83.1%</td>
<td>3.7% 4.0%</td>
<td>0.977 0.960</td>
</tr>
<tr>
<td></td>
<td>TS-OTA1</td>
<td>95.9% 89.4%</td>
<td>87.1% 80.7%</td>
<td>97.9% 75.1%</td>
<td>4.8% 5.9%</td>
<td>0.989 0.928</td>
</tr>
<tr>
<td></td>
<td>CC-OTA</td>
<td>93.4% 88.5%</td>
<td>83.9% 80.1%</td>
<td>90.7% 71.2%</td>
<td>5.7% 5.8%</td>
<td>0.976 0.930</td>
</tr>
<tr>
<td></td>
<td>Comp1</td>
<td>85.4% 83.6%</td>
<td>81.5% 78.4%</td>
<td>58.7% 53.3%</td>
<td>4.9% 5.3%</td>
<td>0.858 0.848</td>
</tr>
<tr>
<td></td>
<td>Comp2</td>
<td>85.6% 86.4%</td>
<td>81.6% 75.7%</td>
<td>58.9% 59.6%</td>
<td>5.0% 5.7%</td>
<td>0.858 0.801</td>
</tr>
<tr>
<td></td>
<td>Comp1</td>
<td>86.8% 84.8%</td>
<td>79.5% 75.7%</td>
<td>62.6% 56.6%</td>
<td>5.3% 6.0%</td>
<td>0.899 0.885</td>
</tr>
<tr>
<td></td>
<td>VCO1</td>
<td>86.4% 82.9%</td>
<td>82.2% 77.6%</td>
<td>58.2% 50.7%</td>
<td>4.2% 5.3%</td>
<td>0.860 0.816</td>
</tr>
<tr>
<td></td>
<td>VCO2</td>
<td>86.1% 82.9%</td>
<td>82.1% 77.6%</td>
<td>61.3% 50.7%</td>
<td>4.9% 5.3%</td>
<td>0.857 0.809</td>
</tr>
<tr>
<td></td>
<td>SCF1</td>
<td>90.8% 84.5%</td>
<td>75.8% 72.6%</td>
<td>70.4% 50.6%</td>
<td>4.9% 5.7%</td>
<td>0.936 0.848</td>
</tr>
<tr>
<td></td>
<td>SCF2</td>
<td>91.3% 84.1%</td>
<td>76.5% 71.4%</td>
<td>73.2% 50.6%</td>
<td>4.9% 6.0%</td>
<td>0.911 0.876</td>
</tr>
<tr>
<td>Avg.</td>
<td></td>
<td>90.0% 86.5%</td>
<td>81.2% 77.4%</td>
<td>73.9% 61.0%</td>
<td>4.9% 5.3%</td>
<td>0.916 0.872</td>
</tr>
<tr>
<td>PEA</td>
<td>CC-OTA</td>
<td>82.8% 80.9%</td>
<td>75.5% 73.0%</td>
<td>46.3% 37.5%</td>
<td>5.0% 4.6%</td>
<td>0.870 0.803</td>
</tr>
<tr>
<td></td>
<td>CM-OTA</td>
<td>93.8% 92.0%</td>
<td>78.8% 76.7%</td>
<td>88.7% 78.9%</td>
<td>5.2% 5.2%</td>
<td>0.990 0.983</td>
</tr>
<tr>
<td></td>
<td>TS-OTA2</td>
<td>94.3% 92.0%</td>
<td>81.6% 76.7%</td>
<td>87.3% 78.9%</td>
<td>4.3% 5.2%</td>
<td>0.974 0.983</td>
</tr>
<tr>
<td></td>
<td>TS-OTA1</td>
<td>92.7% 92.2%</td>
<td>86.6% 85.0%</td>
<td>84.0% 83.8%</td>
<td>4.4% 5.0%</td>
<td>0.976 0.972</td>
</tr>
<tr>
<td></td>
<td>Comp1</td>
<td>85.0% 83.5%</td>
<td>79.1% 76.2%</td>
<td>53.5% 48.5%</td>
<td>4.7% 5.0%</td>
<td>0.858 0.876</td>
</tr>
<tr>
<td></td>
<td>Comp2</td>
<td>84.0% 83.5%</td>
<td>77.8% 77.1%</td>
<td>49.5% 47.3%</td>
<td>4.7% 4.7%</td>
<td>0.840 0.826</td>
</tr>
<tr>
<td></td>
<td>Comp1</td>
<td>85.0% 84.5%</td>
<td>77.5% 76.1%</td>
<td>55.6% 54.6%</td>
<td>5.3% 5.7%</td>
<td>0.866 0.869</td>
</tr>
<tr>
<td></td>
<td>VCO1</td>
<td>83.9% 83.4%</td>
<td>77.3% 75.2%</td>
<td>50.4% 50.6%</td>
<td>5.0% 5.6%</td>
<td>0.816 0.811</td>
</tr>
<tr>
<td></td>
<td>VCO2</td>
<td>84.0% 83.7%</td>
<td>75.1% 76.3%</td>
<td>53.9% 50.6%</td>
<td>6.0% 5.3%</td>
<td>0.836 0.814</td>
</tr>
<tr>
<td></td>
<td>SCF1</td>
<td>86.7% 85.0%</td>
<td>81.2% 79.2%</td>
<td>60.9% 54.4%</td>
<td>4.7% 4.8%</td>
<td>0.871 0.872</td>
</tr>
<tr>
<td></td>
<td>SCF2</td>
<td>86.4% 85.5%</td>
<td>81.5% 80.8%</td>
<td>59.0% 55.1%</td>
<td>4.5% 4.4%</td>
<td>0.866 0.869</td>
</tr>
<tr>
<td>Avg.</td>
<td></td>
<td>87.4% 86.5%</td>
<td>79.6% 78.1%</td>
<td>63.7% 59.1%</td>
<td>4.9% 5.0%</td>
<td>0.895 0.886</td>
</tr>
</tbody>
</table>