From Specification to Silicon: Towards Analog/Mixed-Signal Design Automation using Surrogate NN Models with Transfer Learning

Juzheng Liu[1], Shiyu Su[1], Meghna Madhusudan[2], Mohsen Hassanpourghadi[1], Samuel Saunders[1], Qiaochu Zhang[1], Rezwan Rasul[1], Yaguang Li[3], Jiang Hu[3], Arvind Kumar Sharma[3], Sachin S. Sapatnekar[2], Ramesh Harjani[2], Anthony Levi[1], Sandeep Gupta[1] and Mike Shuo-Wei Chen[1]
[1]Department of Electrical and Computer Engineering, University of Southern California
[2]Department of Electrical and Computer Engineering, University of Minnesota
[3]Department of Electrical and Computer Engineering,Texas A&M University
Los Angeles, CA, USA
Email: juzhengl@usc.edu

2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD) | 978-1-6654-4507-8/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICCAD51958.2021.9643445

***Abstract*—We propose a complete analog mixed-signal circuit design flow from specification to silicon with minimum human-in-the-loop interaction, and verify the flow in a 12nm FinFET CMOS process. The flow consists of three key elements: neural network (NN) modeling of the parameterized circuit component, a search algorithm based on NN models to determine its sizing, and layout automation. To reduce the required training data for NN model creation, we utilize transfer learning to improve the NN accuracy from a relatively small amount of post-layout/silicon data. To prove the concept, we use a voltage-controlled oscillator (VCO) as a test vehicle and demonstrate that our design methodology can accurately model the circuit and generate designs with a wide range of specifications. We show that circuit sizing based on the transfer learned NN model from silicon measurement data yields the most accurate results.**

***Index Terms*—AMS circuit design automation, circuit modeling, layout automation, silicon verified CAD**

I. Introduction

With the development of advanced technology, analog mixed-signal (AMS) circuit design is becoming more of a challenge due to increased relative design cost. As an example, FinFET CMOS technologies involve much longer simulation time in both schematic and post-layout level, and impose more complex layout rules compared to bulk CMOS technology. This severely limits design exploration for an optimal design such as architecture, device sizing, and biasing, given a target AMS specification and limited design time. Therefore, AMS circuit design automation can play a crucial role in resolving these challenges. There have been several prior efforts aimed at implementing AMS design automation. At the circuit architecture level, one approach has been to adopt a mostly digital AMS architecture using digital standard cells, allowing conventional digital design automation flow to lower the design cost [1], [2]. However, this digital-library-based AMS circuit design still requires the involvement of human designers in schematic designs and layout iterations. For schematic design exploration, Bayesian Optimization (BO) [3] and Artificial Neural Networks (ANN) [4], [5] have been applied to automatically size the netlist and significantly reduce the effort required from human designers in the early design stages. However, this prior art mainly focused on schematic-level design without layout. [6] further explored general circuit sizing for both schematic and layout with Reinforcement Learning (RL). However, this can be very time consuming for layout design since it requires a large number of real-time post-layout simulation steps.

In this work, we propose a complete AMS circuit design automation flow from target design specification to layout with significantly reduced human intervention and design time. To achieve design automation and facilitate design-space exploration, we integrate an NN model-based circuit sizing algorithm [4] with the ALIGN [7] layout automation flow. To enhance the accuracy of the NN model for design exploration while minimizing the training cost, we apply Transfer Learning (TL) [8] to the NN model in later design stages, including post-layout and silicon measurement. In a nutshell, the NN-based circuit surrogate model is created using relatively low-cost schematic simulations. Next, TL takes a few post-layout simulation samples from the ALIGN layout flow, and effectively transfers the schematic-level model to the post-layout model, which can be used for layout-aware circuit sizing. Lastly, to overcome the modeling and layout extraction inaccuracy, especially for high frequency AMS circuit designs, we further enhance the NN model by TL with the silicon measurement data, i.e., a silicon-level NN model. To prove the effectiveness of our design flow, we taped out various VCO test structures and demonstrated that the circuit sizing based on the silicon-level NN model yields the most accurate sizing result.

From the standpoint of real applications, the proposed design flow can be useful in many ways. If the user needs to precisely design for a certain target circuit specification, the flow can efficiently model and size the circuit with the layout parasitic extraction (LPE) information included, which can be much faster than manual layout iterations or layout-

level parameter sweeps. When multiple different design targets need to be fulfilled, the advantage of this design flow can be greater. Unlike BO- or RL-based circuit sizing, the NN model based searching algorithm can quickly size the circuit to meet different specifications without the need for extra modeling or simulation, and the layout tool can automatically generate the corresponding layouts without human interaction. Furthermore, when silicon test structures are available (which is usually the case for real product development), the model can be further enhanced by TL and the sizing results can be even closer to the expected values as shown by comprehensive experiments.

The remaining part of the paper is arranged as follows: Section II will discuss the existing circuit design automation approaches. Section III will introduce the proposed design flow in detail. Section IV contains the experimental results using the proposed flow, and the last section will conclude the whole work.

II. Related Work

In this section, we will discuss several directions for circuit design automation flow and compare the advantages and disadvantages in terms of flow completeness, design time consumption, and required human intervention.

A. Digital library based fully synthesized AMS circuit

The first widely used automatic AMS circuit generation method is to use a digital standard cell library and realize the AMS circuit function via digital gates. For instance, in [2], the authors used a set of parallel NAND gates as a varactor and binary scaled inverters as a digital-to-analog converter to realize a fully synthesized digital controlled oscillator using a digital synthesis and layout tool. In [1], the authors customized the standard cell library to include some analog blocks, and used RTL to synthesize an analog-to-digital converter. The advantage of this method is that it leverages commercially available digital design flow and hence accelerates the design process. However, the mostly digital architecture cannot be generalized to cover the entire range of AMS specifications. Moreover, significant human design effort is still needed, involving intensive layout iterations. Lastly, such a design approach faces a lack of automatic netlist sizing tools that can meet target analog design specifications.

B. Reinforcement learning-based circuit design automation

Different from the first approach, [6] leverages RL to design the circuit for a target along a trajectory to automatically size the components in the netlist and generate the schematic. From a certain starting design point, the RL agent will make a decision on how to change each design parameter, and the simulation result for this step will be collected for the next decision. The RL agent is trained by the policy gradient, and requires a large number of environmental steps. After training, for most of the design targets in the predefined target space, the agent can successfully find the right design within a few steps. From one perspective, this design approach is able to generate different designs efficiently without human intervention. Nevertheless, when considering the post-layout design stage, the design agent requires real-time post-layout simulations steps which can consume a significant amount of computation time. Furthermore, when several designs with various specifications are needed, the RL agent needs to generate the trajectory with post-layout simulation steps for each of the designs, which can be even more time-consuming.

C. Bayesian optimization-based circuit design automation

Another circuit sizing method uses BO to directly tune the circuit under design. After determining the design parameters and metrics, a reward is constructed as a function of the metrics according to the design target. A few samples are randomly selected from the design space to initialize the Gaussian Process Regression (GPR) model between the design parameters and reward functions. Then an acquisition function is constructed according to the probabilistic prediction of the GPR model in the whole design space, and the next sample that maximizes the acquisition function will be selected and used to update the GPR model. [3] shows that by deliberately designing the reward and acquisition function, BO can effectively optimize the circuit towards the target. Unfortunately, layout-level design examples were not demonstrated in [3]. In circumstances where a layout automation tool is incorporated with the BO-based circuit sizing algorithm, this approach can be time consuming, since the optimization process may involve hundreds or even thousands of post-layout simulations.

D. Model-based circuit design automation

This class of methods avoids the time-consuming real-time simulation requirement, mentioned in Sections II-B and II-C, by first training a circuit surrogate model and then directly applying search algorithms based on the surrogate model. In [5], the authors train an NN as the surrogate model of the circuit under design and apply a local minimum search to find optimal circuit sizing. [4] further improves the model-based search algorithm by breaking large-scale AMS circuits into a set of smaller-scale modules, each represented by a surrogate NN model. It then constructs a complete surrogate NN model by connecting all the modules with a module-linking graph to facilitate netlist sizing via a gradient-based search algorithm. While the search over the surrogate model is fast, the disadvantage of the NN model-based approach is the need for a training dataset, and that it can involve expensive simulation time especially when post-layout simulations are considered.

To alleviate the aforementioned limitations, we propose an AMS circuit design automation flow that combines the model-based search algorithm and the ALIGN layout automation tool to complete a design flow from design specification to silicon. We further apply the TL technique to the model-based searching flow to generate the post-layout/silicon-level model and perform the circuit sizing with better efficiency and accuracy.

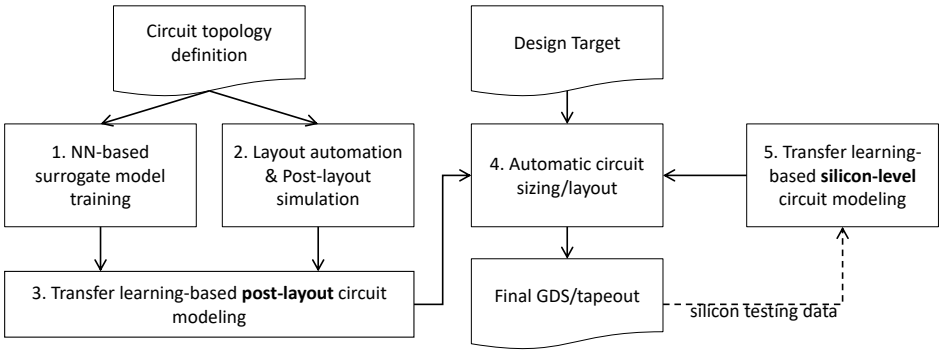

Fig. 1: Proposed design flow from specification to silicon

III. Design Automation Flow

In this work, we propose a complete design automation flow, as shown in Fig. 1, which can efficiently and accurately size and layout a circuit netlist for a given specification. Given a certain AMS circuit topology, we will first create a circuit surrogate model in steps 1 to 3 with the layout information considered, and then size and layout the circuit in step 4 without the need for further simulations. In step 1, we start with modeling the circuit Parameters-to-Metrics (P2M) function by sampling with low-cost schematic simulations and training of an NN as the circuit surrogate model from scratch. To incorporate the LPE information into the surrogate model, step 2 uses the ALIGN layout automation tool and creates several layouts that are randomly selected in the design space. In step 3, with the layouts from step 2 extracted and simulated, TL is applied to efficiently transfer the schematic-level circuit model to the post-layout model. Once the surrogate model is prepared, in step 4, a fast gradient-based search algorithm is applied to find the optimal design parameters given one or multiple design specifications, and the same layout automation tool is applied to generate the final GDSII. Furthermore, if the design proceeds to fabrication, in step 5, the silicon measurement results can be utilized by TL to further improve the circuit modeling accuracy, and a more precise circuit sizing can be derived. The details of each design step will be discussed as follows.

A. NN-based surrogate model training

The first step of the proposed design flow is to characterize the behavior of the AMS circuit under design. We build a parameterized netlist and model the parameters ($\mathbf{p}$) to metrics ($\mathbf{m}$) function:

$$\mathbf{m} = f(\mathbf{p}), \tag{1}$$

where $\mathbf{p}$ is defined as a vector of tunable circuit parameters such as transistor sizes, and $\mathbf{m}$ is defined as the vector of performance metrics of the circuit block such as power consumption. To model the function f by an NN, we generate the training dataset through SPICE simulations. More specifically, in the design space where each dimension is a parameter, we define the upper and lower bond of each parameter and randomly sample for design points. SPICE simulation is then performed for each point, and the corresponding metrics are used as the golden reference in the training.

When training an NN surrogate model, we try to minimize the Mean Squared Error (MSE) between the model predicted metrics and the simulation results. Since the numerical value of different parameters/metrics can be different in orders of magnitude, each parameter/metric is linearly re-scaled to $[-1, 1]$ according to the corresponding minimum and maximum value of the whole dataset. When performing the training, we use the Adam [9] optimizer to minimize the MSE defined as:

$$MSE = \frac{1}{k}\sum_{i=1}^{k}(\hat{m}_i - m_i)^2, \tag{2}$$

where $\hat{m}_i$ is the predicted i^{th} metric from the model, m_i is the ground truth of the i^{th} metric from the simulation, and k is the number of metrics for this circuit block.

B. Automatic layout generation

Layout parasitics and layout-dependent effects are significant in advanced technology nodes [10], [11] and they affect circuit performance to a large extent. Both steps 2 and 4 of the flow use a fast automatic layout generator and extract layout parasitics to analyze the post-layout performance of a design. The open-source ALIGN [7], [12] software is used for layout generation.

ALIGN ("Analog Layout, Intelligently Generated from Netlists") takes a SPICE netlist of a circuit as input and generates its layout as a GDSII file. For the proposed flow, parameterized netlists of a circuit are fed into ALIGN, and their corresponding layouts are extracted and simulated. Several optimization steps are added to ALIGN to ensure a high-performing layout is generated for each input circuit netlist.

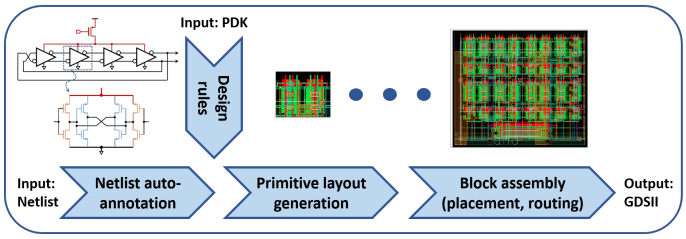

Fig. 2: Overview of the ALIGN layout generation process [12]

ALIGN has a modular approach as shown in Fig. 2. The main modules in ALIGN that were used in this work include netlist annotation where common building blocks called "primitives" and symmetries in the circuit are detected. Design rule capture is used to abstract the proprietary PDK into a simplified grid which the other modules obey. Parameterized

primitive cell generation automatically builds layouts for the primitives using the number of fins and fingers in a transistor as parameters. Hierarchical block assembly performs placement and routing while meeting geometric and electrical constraints provided to it.

To enhance layout quality so that it is comparable to manual layouts, we perform several optimizations that are added on top of the baseline version of ALIGN [13]. These include 1) primitive-level optimization, 2) signal routing optimization, and 3) power mesh and power routing optimization. In this section, we highlight these optimizations with the aid of a ring-oscillator-based VCO circuit example. The important performance metrics of the circuit are the oscillation frequency (Fosc) and power consumption (PW) at different control voltages (Vctrl). The circuit is laid out using a commercial 12nm CMOS FinFET technology.

TABLE I: VCO ALIGN layout comparison without optimization (baseline) and with primitive (P), signal routing (SR), and power routing (PR) optimization

Performance metric		ALIGN (baseline)	ALIGN (P)	ALIGN (P+SR)	ALIGN (P+SR+PR)
Vctrl = 0V	Fosc (GHz)	2.8	3.2	4.5	4.8
	PW (μW)	851	943	1272	1423
Vctrl = 0.4V	Fosc (GHz)	1.4	1.4	1.5	1.6
	PW (μW)	212	212	213	230

1) Primitive-level optimization: In ALIGN, primitives refer to lowest-level blocks, consisting of a small number of devices in the schematic, such as differential pairs and current mirrors. Primitives can be generated with multiple aspect ratios, and different combinations of fins and fingers for the same schematic sizing. Mesh routing is used inside the primitives. Increasing the number of parallel metal wires and vias in the mesh reduces resistive and increases capacitive parasitics that affect the primitive performance. We automatically generate the layout of a set of primitives with different aspect ratios and with mesh routing structures within the primitive to mitigate high wire resistances in lower metal layers of advanced FinFET processes. We then simulate the performance of each candidate primitive layout and choose the best-performing primitives to be passed on to the hierarchical block assembly module in ALIGN.

In Table I, the performance of ALIGN layouts for the VCO circuit, with optimizations at different stages of ALIGN, are compared. This circuit has a current starved inverter primitive. The "ALIGN (baseline)" column shows the ALIGN layout performance for the baseline implementation when no additional optimizations are carried out. The "ALIGN (P)" column shows the performance of the final ALIGN layout when the above primitive-level optimization is carried out, choosing suitable aspect ratios and mesh routing structures for the current-starved inverter primitive. The oscillation frequency shows an improvement over the baseline due to this primitive-level optimization.

2) Signal routing optimization: The signal routing step in ALIGN connects various primitives and hierarchical blocks in the layout. These routes are typically much longer than those found inside the primitive. In FinFET technologies, wire and via resistances are significant in lower-level interconnect layers. Resistive bottlenecks can be overcome by providing wider routes and multiple vias (implemented as multiple parallel wires with vias in FinFET technologies); however, the capacitive parasitics increase in this case. The trade-off between the resistive and capacitive parasitics is circuit- and design-specific. The optimal number of parallel routes for each signal net is determined, considering the impact of both resistive and capacitive parasitics on circuit performance. This number is then provided to the global and detailed router, which uses this specification to create multiple parallel routes at the nets.

In Table I, the "ALIGN (P+SR)" column shows the ALIGN layout performance when both primitive-level and signal routing optimizations are carried out. There is a significant improvement in oscillation frequency at Vctrl = 0V when both optimizations are applied. At this control voltage, the circuit is resistance-limited as it draws a high current and the IR drop at the nets is significant. Therefore, by adding parallel wires, resistance and IR drop are lowered. At Vctrl = 0.4V, the circuit is not as limited by resistance since it draws a smaller current and hence the improvement in frequency after adding an optimal number of parallel wires is smaller.

3) Power mesh construction and power routing optimization: Routing from the power supply ports to the circuit terminals is critical in analog circuits that continuously carry large amounts of current. A power mesh structure is created to reduce the resistance along this path and distribute the supply across different parts of the circuit. The layers and number of tracks in the power mesh can be controlled to provide low resistance while using optimum resources. The routing from the lower layers of the power mesh to the circuit terminals is also critical. We add a further optimization to the baseline version of ALIGN by adding parallel power routes to reduce losses along this high resistance path.

In Table I, the "ALIGN (P+SR+PR)" column shows the ALIGN layout performance with primitive-level, signal and power mesh and power routing optimization added to the baseline version. The performance of the circuit has improved considerably. As compared to the baseline, these optimizations collectively improve the frequency by 70% for Vctrl = 0V and by 14% at Vctrl = 0.4V. As explained earlier, a lower percentage improvement in the latter case is expected because the circuit is not resistance-limited and adding parallel wires does not significantly improve performance.

4) Runtime for VCO layout generation: The ALIGN layout generator, modified to incorporate all three optimizations, is capable of generating layouts for multiple input netlists simultaneously and is suitable for generating training data. It can generate a VCO layout with a runtime of 100s on a RedHat system with an Intel(R) Xeon(R) Silver 4114 CPU @2.20 GHz and 20 cores. It has a peak memory requirement of 0.2% or 320 MB and 500 such layouts can be generated in parallel on this system.

C. TL-based model enhancement for accurate P2M modeling

For step 3 of the proposed design flow, we need to incorporate the LPE information into the surrogate model to ensure that the model can predict the real-world silicon result more accurately. However, it is difficult to train a model from scratch using only the post-layout dataset because post-layout simulations can take a long time. For instance, in 12nm FinFET technology, post-layout simulations typically consumes 20 to 30 times longer than the schematic-level simulations for the same circuit. To make post-layout modeling feasible, we utilize the TL technique from [8] to reuse the schematic-level model. More specifically, we add two linear layers to the input and output of a well-trained schematic-level surrogate model. When performing the training, only the two newly added layers are updated, and the remaining layers are fixed. Mathematically, the post-layout model is constructed as follows:

$$\hat{f}_{layout}(\mathbf{p}) = \mathbf{A}\hat{f}_{sch}(\mathbf{Cp}+\mathbf{d})+\mathbf{b} = \hat{\mathbf{m}}_{\mathbf{lay}}, \tag{3}$$

where $\hat{f}_{sch}$ is the schematic-level surrogate model, and $\hat{f}_{layout}$ is the post-layout-level surrogate model. With n parameters and k metrics, $\mathbf{C}$ (an $n \times n$ matrix) and $\mathbf{d}$ (a $1 \times n$ bias vector) are the mapping function of the parameters, i.e., the added input linear layer, while $\mathbf{A}$ (a $k \times k$ matrix) and $\mathbf{b}$ (a $1 \times k$ bias vector) are the mapping function of the metrics, or the added output linear layer. To justify the use of linear TL layers between schematic and post-layout model, we perform propagation delay analysis of an inverter for illustration purpose. If we express the inverter delay as a function of the width and length of the PMOS and NMOS, assuming NMOS and PMOS have the same size, the schematic-level delay without layout parasitics can be approximated as follows:

$$t_{d,sch}(\mathbf{p}) = 0.69 C_L R_{on}; \mathbf{p} = (W, L), \tag{4}$$

$$C_L = C_{ox}WL + 2C_{ov}W + 2C_{db}, \tag{5}$$

$$R_{on} = \frac{3V_{DD}}{4k'[(V_{DD}-V_T)V_{dsat} - \frac{V_{dsat}^2}{2}]\frac{W}{L}}\left(1 - \frac{7}{9}\lambda V_{DD}\right) \tag{6}$$

If we regard the terms that are irrelavent to W and L in (6) as constants and lump them into α, the equation can be simplified as follows:

$$t_{d,sch} = \alpha(C_{ox}WL + 2C_{ov}W + 2C_{db})\frac{L}{W} \tag{7}$$

After the inverter is laid out, there will be a parasitic resistance in series with R_{on}, and a parasitic capacitance in parallel with C_L, which can be reasonably estimated as

$$R_p \propto 1/W = R_{unit}/W \tag{8}$$

$$C_p \propto W = C_{unit}W \tag{9}$$

With the parasitic effect, the delay will be changed to:

$$t_{d,layout} = \alpha(C_{ox}WL + 2C_{ov}W + 2C_{db} + C_{unit}W)\frac{L + R_{unit}/\alpha}{W} \tag{10}$$

Corresponding to (3), to transfer the layout-level delay (10) from schematic-level (7) as:

$$t_{d,layout} = \mathbf{A}t_{d,sch}(\mathbf{Cp}+\mathbf{d})+\mathbf{b}; \mathbf{p} = (W, L) \tag{11}$$

we can annotate two new variables as:

$$W' = \frac{2C_{ov} + C_{unit} - C_{ox}R_{unit}/\alpha}{2C_{ov}}W = \theta_w W \tag{12}$$
$$L' = (L - R_{unit}/\alpha)/\theta_w$$

and rewrite (10) into

$$\begin{aligned} t_{d,layout} &= \theta_w^2 \alpha(C_{ox}W'L' + 2C_{ov}W' + 2C_{db})\frac{L'}{W'} \\ &= \theta_w^2 t_{d,sch}(\mathbf{p}'); \mathbf{p}' = (W', L') \end{aligned} \tag{13}$$

Compared to (11), we can get

$$\begin{aligned} \mathbf{A} &= (\theta_w^2) \\ \mathbf{b} &= (0) \\ \mathbf{C} &= \begin{pmatrix} \theta_w & 0 \\ 0 & 1/\theta_w \end{pmatrix} \\ \mathbf{d} &= (0, -R_{unit}/(\alpha\theta_w)) \end{aligned} \tag{14}$$

for the input and output linear mapping.

In conclusion, by adding two linear layers to the input and output of the pre-trained schematic-level model, and only training the two layers using post-layout simulation results, it would be sufficient for modeling inverter delay in the post layout stage. As a result, TL can significantly improves the post-layout modeling accuracy with small number of training samples and prevents over-fitting.

Similarly, the same TL technique can be applied to silicon-level circuit modeling after the circuit is fabricated and tested, i.e. step 5 of the proposed design flow. Based on the transfer-learned post-layout circuit model, we can cascade additional TL layers and re-train an accurate silicon-level model with only a few silicon measurement samples. The experimental results of the modeling accuracy will be demonstrated in Section IV-A.

D. Automatic circuit sizing with post-layout/silicon-level model

After preparing the surrogate model, we perform the automatic sizing of the circuit block to satisfy the desired design targets. In step 4 of the proposed design flow, we use the search algorithm illustrated in [4] to find multiple circuit parameter candidates. This algorithm incorporates a gradient-based parameter search using NN models of the circuit blocks. In [4], the search algorithm only utilized NN models trained from schematic-level simulations. In this paper, we further enhance the model with post-layout simulation and/or silicon measurement results, leading to much improved search accuracy in terms of matching with the final silicon performance.

Since we use an optimization-based search methodology, a penalty function is designed to help find the optimal circuit

parameters. Additionally, this penalty function should be differentiable everywhere for the gradient-based optimizer. Here, we define the circuit sizing problem as:

$$\begin{aligned} \arg\min_{\mathbf{p}} \quad & \boldsymbol{g}^o(\hat{\mathbf{m}}), \\ s.t.: \quad & \boldsymbol{g}^i(\hat{\mathbf{m}}) \geq 0, \\ & \boldsymbol{g}^e(\hat{\mathbf{m}}) = 0, \\ & \hat{\mathbf{m}} = \hat{f}(\mathbf{p}), \end{aligned} \tag{15}$$

where $\hat{f}$ is the circuit surrogate NN model, $\boldsymbol{g}^o$ includes the specifications that should be minimized, $\boldsymbol{g}^i$ includes the inequality constraints, and $\boldsymbol{g}^e$ includes the equality constraints that should be satisfied. For example, $\boldsymbol{g}^o$ can be the power consumption that should be minimized, $\boldsymbol{g}^i$ can be the bandwidth of a system desired to be larger than 10MHz, and $\boldsymbol{g}^e$ can be the gain of a feedback system that should be exactly equal to 2. For simplification, we assume $\boldsymbol{g}^o$, $\boldsymbol{g}^i$ and $\boldsymbol{g}^e$ are a list of subfunctions (i.e., $\boldsymbol{g} = [g_1, g_2, ...]$) that are only related to the circuit's design metrics and are differentiable for the given inputs. Knowing (15), we can construct the penalty function for the automatic sizing problem as:

$$\begin{aligned} penalty(\mathbf{p}) = & \sum_j w_j^o \times g_j^o(\hat{f}(\mathbf{p})) \\ & + \sum_k elu(w_k^i \times g_k^i(\hat{f}(\mathbf{p}))) \\ & + \sum_l w_l^e \times (g_l^e(\hat{f}(\mathbf{p})))^2, \end{aligned} \tag{16}$$

where $\boldsymbol{w}$s are the optimization weights determined by the importance of each specification and elu is the exponential linear unit function. Function elu linearly increases the penalty when the inequality is not satisfied and exponentially reduces it if satisfied, and it is differentiable everywhere. To satisfy the equality constraints, we use the MSE which is differentiable and increases the penalty if it is not satisfied. To calculate the gradients for the gradient-based optimization, we can use the chain rule to first find the $\frac{\partial penalty}{\partial \hat{\mathbf{m}}}$ and then derive $\frac{\partial \hat{\mathbf{m}}}{\partial \mathbf{p}}$. Machine learning tools such as TensorFlow can easily complete both tasks and therefore compute the gradients of the penalty function with respect to the design parameters.

The significant aspect of the NN model-based circuit sizing algorithm is its capability to perform fast parameter search without real-time simulations. Instead of conventional global optimizers, such as simulated annealing used in most CAD tools, it exploits advanced gradient-based optimizers such as Adam. The Adam optimizer converges with fewer iterations compared to the conventional global ones. However, since it is a local optimizer, the final results can depend on the initialization point. Therefore, a Monte-Carlo on the initialization of the optimization is incorporated. In each Monte-Carlo sample, the starting point is chosen randomly from the parameter range; thus, Adam converges to different parameter candidates. As shown in [4], the addition of the Monte-Carlo technique increases the probability of finding globally optimal results, within the precision of the NN modeling. It can surpass

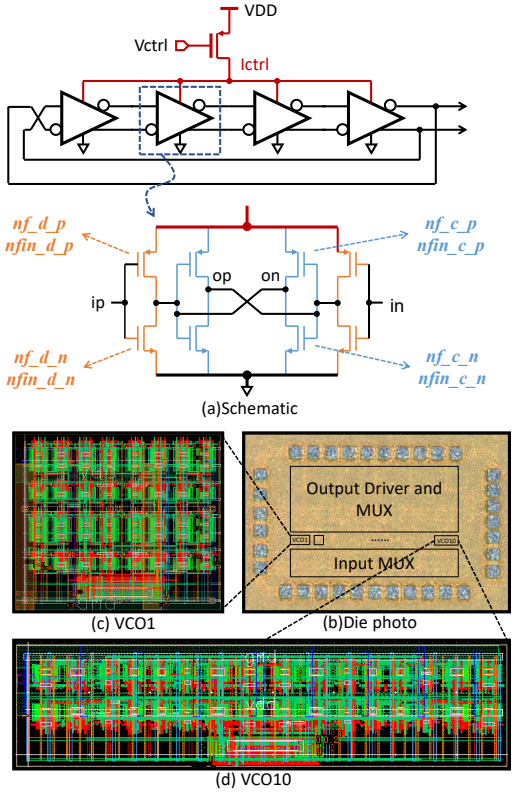

Fig. 3: VCO design example (a) Schematic of the VCO (b) Die photo of the fabricated chip (c) Layout illustration of VCO 1 (d) Layout illustration of VCO 10

the conventional global optimizer performance, in terms of both fewer iterations for convergence and more optimized final results. After the sizing is done, we use the layout automation tool described in Section III-B for the final GDSII generation. To verify the effectiveness of the search algorithm, in Section IV-B, we used the silicon measurement results as the design target, performed the search using post-layout and silicon-level models, and compared the results with the actual circuit sizes.

IV. Experimental Results

To demonstrate the effectiveness and efficiency of the proposed AMS circuit design automation flow from specification to the silicon level, we implement an array of VCOs in 12nm CMOS FinFET technology using the design flow. VCOs are widely used in different kinds of AMS circuits such as phase lock loop [14], analog to digital converters [15], and computing circuits [16]. The schematic of the design is shown in Fig. 3(a). This inverter-coupled VCO consists of four identical stages, each stage has two differential inverter drivers (in orange color) and two cross-coupled inverters (in blue color). To be able to size this VCO design to satisfy various design specifications, we assigned in total eight design parameters to the VCO, which are the number of fingers (nf) and number of fins per finger (nfin) for the NMOS (n) and PMOS (p) of the inverter driver (d) and cross-coupled inverters (c). With the help of layout automation, 10 different VCO designs were taped-out and measured in terms of oscillation frequency (Fosc) and power consumption (PW) with different

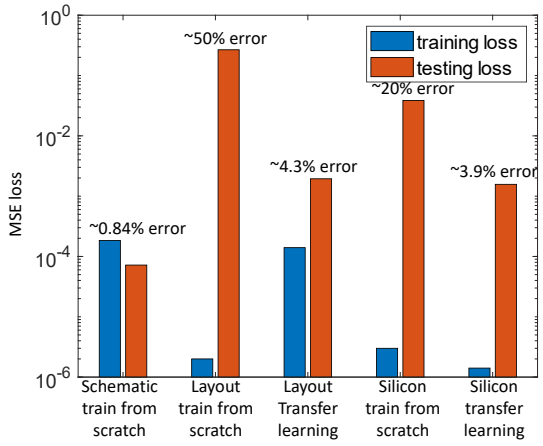

Fig. 4: Training and testing MSE loss comparison

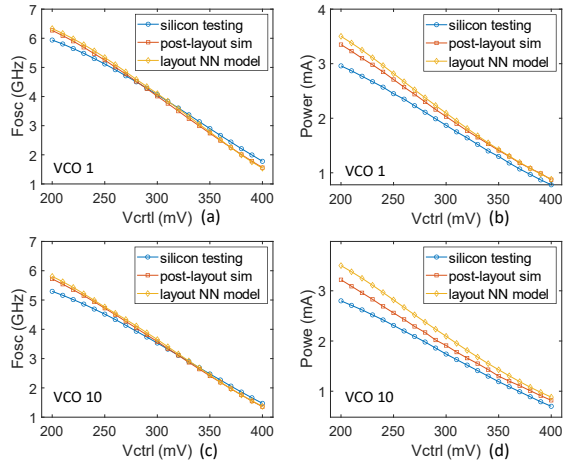

Fig. 5: Layout NN model prediction, post-layout simulation and silicon testing results comparison: (a) Fosc vs. Vctrl for VCO 1 (b) power vs. Vctrl for VCO 1 (c) Fosc vs. Vctrl for VCO 10 (d) power vs. Vctrl for VCO 10

control voltages. The die photo is shown in Fig. 3(b), and the zoomed in VCO layout details are shown in Fig. 3(c) and (d)

A. Post-layout modeling verification

Following the circuit modeling steps in Section III, we first densely sampled the parameters (the aforementioned eight parameters) to metrics (Fosc and PW) function in the design space via low cost schematic-level simulation, and trained a 3-hidden-layer MLP (number of neurons per layer: [8, 16, 32, 16, 2]) from scratch. All the parameters and metrics have been linearly re-scaled to [-1, 1] according to the minimum and maximum value in the dataset before training. As shown in the first column of Fig. 4, with a large number of training samples the VCO surrogate model can precisely predict the schematic-level performance metric. In this particular case, we used 5,250 training samples and 500 testing samples, and the training sample generation took around 95 minutes with parallel threads. We take the square root of the testing MSE loss as the approximated prediction error since both metrics have been re-scaled to [-1, 1].

With the well-trained schematic-level model, we utilized the TL technique mentioned in Section III-C to efficiently include the layout information into the model, and compared the training and testing loss with the case when the post-layout

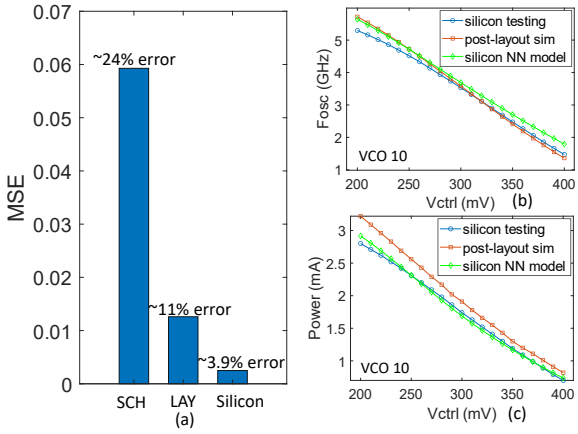

Fig. 6: (a) Prediction errors of silicon result using schematic-, layout- and silicon- level NN model (b) Fosc vs. Vctrl for VCO 10 (c) power vs. Vctrl for VCO 10 from post-layout simulation, silicon-level model prediction and silicon testing

model is trained from scratch. In this example, we only used a single layout to generate 20 training samples at 20 different control voltages, and the trained model was tested with another 180 samples with different VCO parameters. Each of the post-layout simulations took around 24 minutes. As shown in the second and third columns of Fig. 4, while the model trained from scratch have a 50% testing error because of overfitting, TL can effectively reuse the information in the schematic-level model and the transfer-learned model can predict the post-layout metrics with only 4.3% error.

To intuitively demonstrate the post-layout modeling accuracy, we compared the performance metrics from two of the VCO designs using: 1. post-layout simulation, 2. post-layout model prediction, and 3. silicon measurement result. The results are shown in Fig. 5. For both VCO designs, the surrogate model predictions can accurately follow the simulation results, but they are still different from the silicon test result. This discrepancy can be caused by modeling and layout extraction inaccuracy especially for the high frequency cases, and parasitic capacitance and resistance from the peripheral testing circuitry.

B. Silicon results enhanced modeling and sizing

To fix the aforementioned discrepancy between post-layout model prediction and silicon measurement results, we utilized the post-layout surrogate model and performed TL with 40 training samples from the silicon measurement of VCO 1 and 2. For the testing, we used 160 measurement samples from VCO 3 to 10, with 20 samples from each VCO by applying 20 different control voltages. As shown in the last two columns of Fig. 4, we can obtain a much more accurate silicon-level VCO model with TL compared to the model trained from scratch given the same number of training samples.

To examine the prediction accuracy regarding the silicon result, we used the 1. schematic-level model, 2. post-layout model trained by TL, and 3. silicon-level model trained by TL. As shown in Fig. 6(a), if we directly use the schematic-level model or the post-layout model to predict the silicon test results, the MSEs of predictions are approximately 8 and 3

TABLE II: VCO sizing results comparison (VCO 1 and 2 used for model training)

VCO#		nf_c_n	nf_c_p	nf_d_n	nf_d_p	nfin_c_p	nfin_c_n	nfin_d_n	nfin_d_p
VCO 3	schematic	5.21(5)	5.21(5)	10.40(10)	10.40(10)	9.60(10)	7.68(8)	7.68(8)	9.60(10)
	post-layout	4.41(4)	4.41(4)	8.82(9)	8.82(9)	9.98(10)	7.99(8)	7.99(8)	9.98(10)
	silicon-level	4.12(4)	4.12(4)	8.23(8)	8.23(8)	10.00(10)	8.00(8)	8.00(8)	10.00(10)
	actual value	4	4	8	8	10	8	8	10
VCO 4	schematic	7.97(8)	7.97(8)	15.90(16)	15.90(16)	9.00(9)	7.20(7)	7.20(7)	9.00(9)
	post-layout	7.02(7)	7.02(7)	14.00(14)	14.00(14)	9.99(10)	7.99(8)	7.99(8)	9.99(10)
	silicon-level	6.02(6)	6.02(6)	12.00(12)	12.00(12)	9.99(10)	7.99(8)	7.99(8)	9.99(10)
	actual value	6	6	12	12	10	8	8	10
VCO 5	schematic	8.71(9)	8.71(9)	17.40(17)	17.40(17)	9.86(10)	7.89(8)	7.89(8)	9.86(10)
	post-layout	9.15(9)	9.15(9)	18.30(18)	18.30(18)	9.98(10)	7.99(8)	7.99(8)	9.98(10)
	silicon-level	8.25(8)	8.25(8)	16.49(16)	16.49(16)	10.00(10)	8.00(8)	8.00(8)	10.00(10)
	actual value	8	8	16	16	10	8	8	10
VCO 6	schematic	9.44(9)	9.44(9)	18.90(19)	18.90(19)	9.71(10)	7.77(8)	7.77(8)	9.71(10)
	post-layout	9.72(10)	9.72(10)	19.40(19)	19.40(19)	9.95(10)	7.96(8)	7.96(8)	9.95(10)
	silicon-level	9.91(10)	9.91(10)	19.80(20)	19.80(20)	9.95(10)	7.96(8)	7.96(8)	9.95(10)
	actual value	10	10	20	20	10	8	8	10
VCO 7	schematic	5.75(6)	5.75(6)	11.50(12)	11.50(12)	7.75(8)	6.20(6)	6.20(6)	7.75(8)
	post-layout	3.15(3)	3.15(3)	6.29(6)	6.29(6)	9.77(10)	7.81(8)	7.81(8)	9.77(10)
	silicon-level	4.21(4)	4.21(4)	8.43(8)	8.43(8)	4.85 (5)	3.88(4)	3.88(4)	4.85(5)
	actual value	4	4	8	8	5	4	4	5
VCO 8	schematic	6.14(6)	6.14(6)	12.30(12)	12.30(12)	7.20(7)	5.76(6)	5.76(6)	7.20(7)
	post-layout	2.82(3)	2.82(3)	5.63(6)	5.63(6)	9.93(10)	7.95(8)	7.95(8)	9.93(10)
	silicon-level	2.21(2)	2.21(2)	4.42(4)	4.42(4)	5.82(6)	4.66(5)	4.66(5)	5.82(6)
	actual value	2	2	4	4	5	4	4	5
VCO 9	schematic	3.73(4)	3.73(4)	7.45(7)	7.45(7)	9.88(10)	7.90(8)	7.90(8)	9.88(10)
	post-layout	3.65(4)	3.65(4)	7.29(7)	7.29(7)	8.60(9)	6.88(7)	6.88(7)	8.60(9)
	silicon-level	3.09(3)	3.09(3)	6.19(6)	6.19(6)	8.66(9)	6.93(7)	6.93(7)	8.66(9)
	actual value	3	3	6	6	10	8	8	10
VCO 10	schematic	6.72(7)	6.72(7)	13.40(13)	13.40(13)	9.29(9)	7.43(7)	7.43(7)	9.29(9)
	post-layout	5.72(6)	5.72(6)	11.40(11)	11.40(11)	9.99(10)	7.99(8)	7.99(8)	9.99(10)
	silicon-level	5.52(6)	5.52(6)	11.00(11)	11.00(11)	9.60(10)	7.68(8)	7.68(8)	9.60(10)
	actual value	5	5	10	10	10	8	8	10

times higher than that of the transfer learned silicon model, respectively. The oscillation frequency and power consumption prediction by the silicon-level model are shown in Fig. 6(b) and (c). Compared with Fig. 5, the model prediction can precisely follow the silicon results (within 5% throughout the frequency tuning range of VCO).

Using the accurate post-layout/silicon-level VCO model, we can further perform the circuit sizing algorithm described in Section III-D to validate the effectiveness of the proposed design flow. We used the silicon measurement results of testing VCOs (3-10) as the design targets (Fosc and PW at a certain control voltage) and perform the circuit sizing algorithm with the VCO surrogate model at different design stages (schematic-level, post-layout, silicon-level). On an NVIDIA 1080 computing platform, the maximum time consumption for one design is 70 seconds. The search results based on the surrogate models are compared to the actual VCO's sizing, as shown in Table II. Note that, the sizing results from the model-based search are continuous values, but the 12nm FinFET technology requires discrete numbers for device sizes. Therefore, the continuous sizing results are rounded and annotated inside parentheses in the table. As shown in the table, the sizing results from the schematic-level model are significantly different from the actual parameter values in silicon, with an average sizing difference of 35%. Sizing with a post-layout model can find much closer design points as compared to the actual parameter values in most cases, and the average sizing difference reduces to 21%. With the silicon-level model, the sizing results show much improved precision in all cases as the average sizing difference further reduces to only 5%. Accordingly, this proposed design flow can significantly accelerate the design process and find the desired design points with different design specifications, especially with the silicon-level circuit model.

V. Conclusion

In this work, we propose a complete and efficient AMS circuit design automation for circuit modeling, sizing, and layout, with good generalization ability for various design specifications. Proved by comprehensive experiments, with the TL-based efficient modeling method and ALIGN-based layout automation, we can generate a layout- or even silicon-level surrogate NN model with excellent efficiency and accuracy, and the model-based search algorithm can rapidly size the circuit under design for one or multiple design specifications without the need for further simulations or re-creating surrogate models. The flow can effectively accelerate the design process, find the desired design points, and therefore reduce the number of layout/silicon iterations in practical AMS circuit design scenarios.

Acknowledgment

The authors wish to acknowledge support from the DARPA POSH program (FA8650-18-2-7853) and program manager Serge Leef. We also thank Global Foundries for access to GF12LP technology. Qiaochu Zhang also acknowledges the funding support from the University of Southern California Provost's Fellowship.

REFERENCES

[1] A. Waters and U.-K. Moon, "A fully automated verilog-to-layout synthesized ADC demonstrating 56dB-SNDR with 2MHz-BW," in *2015 IEEE Asian Solid-State Circuits Conference (A-SSCC)*, 2015, pp. 1–4.

[2] W. Deng, D. Yang, T. Ueno, T. Siriburanon, S. Kondo, K. Okada, and A. Matsuzawa, "A fully synthesizable all-digital PLL with interpolative phase coupled oscillator, current-output DAC, and fine-resolution digital varactor using gated edge injection technique," *IEEE Journal of Solid-State Circuits*, vol. 50, no. 1, pp. 68–80, 2015.

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

[4] M. Hassanpourghadi, R. A. Rasul, and M. S.-W. Chen, "A module-linking graph assisted hybrid optimization framework for custom analog and mixed-signal circuit parameter synthesis," *ACM Transactions on Design Automation of Electronic Systems*, Jan. 2021. [Online]. Available: https://doi.org/10.1145/3456722

[5] Y. Li, Y. Wang, Y. Li, R. Zhou, and Z. Lin, "An artificial neural network assisted optimization system for analog design space exploration," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 10, pp. 2640–2653, 2020.

[6] K. Settaluri, A. Haj-Ali, Q. Huang, K. Hakhamaneshi, and B. Nikolic, "Autockt: Deep reinforcement learning of analog circuit designs," in *2020 Design, Automation Test in Europe Conference Exhibition (DATE)*, 2020, pp. 490–495.

[7] K. Kunal, M. Madhusudan, A. K. Sharma, W. Xu, S. M. Burns, R. Harjani, J. Hu, D. A. Kirkpatrick, and S. S. Sapatnekar, "ALIGN: Open-source analog layout automation from the ground up," in *2019 ACM/IEEE Design Automation Conference (DAC)*, 2019, pp. 77–80.

[8] J. Liu, M. Hassanpourghadi, Q. Zhang, S. Su, and M. S.-W. Chen, "Transfer learning with Bayesian optimization-aided sampling for efficient AMS circuit modeling," in *2020 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, 2020, pp. 1–9.

[9] D. P. Kingma and J. Ba, "ADAM: A method for stochastic optimization," in *2015 International Conference on Machine Learning (ICML)*, 2015.

[10] R. A. Rutenbar, "Analog circuit and layout synthesis revisited," in *2015 ACM International Symposium on Physical Design (ISPD)*, 2015, p. 83.

[11] A. L. S. Loke, D. Yang, T. T. Wee, J. L. Holland, P. Isakanian, K. Rim, S. Yang, J. S. Schneider, G. Nallapati, S. Dundigal, H. Lakdawala, B. Amelifard, C. Lee, B. McGovern, P. S. Holdaway, X. Kong, and B. M. Leary, "Analog/mixed-signal design challenges in 7-nm CMOS and beyond," in *2019 IEEE Custom Integrated Circuits Conference (CICC)*, 2019, pp. 1–8.

[12] T. Dhar, K. Kunal, Y. Li, M. Madhusudan, J. Poojary, A. K. Sharma, W. Xu, S. M. Burns, R. Harjani, J. Hu *et al.*, "ALIGN: A system for automating analog layout," *IEEE Design & Test*, vol. 38, no. 2, pp. 8–18, 2020.

[13] https://github.com/ALIGN-analoglayout/ALIGN-public.

[14] Q. Zhang, S. Su, C.-R. Ho, and M. S.-W. Chen, "A fractional-N digital MDLL with background two-point DTC calibration achieving -60dBc fractional spur," in *2021 IEEE International Solid- State Circuits Conference (ISSCC)*, vol. 64, 2021, pp. 410–412.

[15] T.-F. Wu and M. S.-W. Chen, "A 40MHz-BW 76.2dB/78.0dB SNDR/DR noise-shaping nonuniform sampling ADC with single phase-domain level crossing and embedded nonuniform digital signal processor in 28nm CMOS," in *2020 IEEE International Solid- State Circuits Conference - (ISSCC)*, 2020, pp. 262–264.

[16] I. Ahmed, P.-W. Chiu, and C. H. Kim, "A probabilistic self-annealing compute fabric based on 560 hexagonally coupled ring oscillators for solving combinatorial optimization problems," in *2020 IEEE Symposium on VLSI Circuits*, 2020, pp. 1–2.