Closing the Design Loop: Bayesian Optimization Assisted Hierarchical Analog Layout Synthesis

Mingjie Liu, Keren Zhu, Xiyuan Tang, Biying Xu, Wei Shi, Nan Sun, and David Z. Pan
ECE Department, The University of Texas at Austin, Austin, TX, USA
{jay_liu, keren.zhu, xitang, biying, weishi0079}@utexas.edu, nansun@mail.utexas.edu, dpan@ece.utexas.edu

***Abstract*—Existing analog layout synthesis tools provide little guarantee to post layout performance and have limited capabilities of handling system-level designs. In this paper, we present a closed-loop hierarchical analog layout synthesizer, capable of handling system designs. To ensure system performance, the building block layout implementations are optimized efficiently, utilizing post layout simulations with multi-objective Bayesian optimization. To the best of our knowledge, this is the first work demonstrating success in automated layout synthesis on generic analog system designs. Experimental results show our synthesized continuous-time $\Delta\Sigma$ modulator (CTDSM) achieves post layout performance of 65.9dB in signal to noise and distortion ratio (SNDR), compared with 67.8dB in the schematic design.**

I. Introduction

The expanding markets of emerging applications, such as automotive and Internet of Things (IoT), create large demands for analog and mixed-signal (AMS) integrated circuits (ICs). This increasing demand in consumer electronics calls for a shorter design cycle and time-to-market.

Implementing analog circuit layout is a heavily manual, time-consuming, and error-prone task. Human layout engineers often draw the circuit layouts following conventions and guidelines from the experienced circuit designers. The high cost of obtaining layouts and long turnaround time in layout generation impede efficient feedback to the designer. Moreover, the increasing layout dependent effects in modern technology nodes makes prediction and analysis of post layout performance difficult. Several efforts have been made to automate analog layout synthesis to resolve the challenges above. However, few have been adopted in practical design flows.

Traditional analog layout synthesis tools either have significant design overhead or provide limited guarantee towards the post layout performance. Procedural based analog layout generators [1] require human to directly encode layout implementations, including device placement location, routing metal layer selection, etc. These methods have demonstrated great success in design reuse and technology migrations. However, the codifying of layout implementations for different designs is still manual, costly, and time-consuming. Optimization-based layout tools [2]–[4] require circuit designers to manually input layout constraints, which are enforced during placement and routing. Heuristic constraints based methods suffer significant issues when facing practical designs. Hand-crafted constraints are often design specific, which lack flexibility and generality when transferring to different designs. Performance-driven layout approaches derive equations from consider various layout effects, either analytically [5] or through sensitivity analysis [6]. However, with increased device scaling, analytical sensitivity estimates of parasitics and mismatch are no longer accurate. The impacts of design and layout dependent effects, such as clock coupling and IR drop, also become extremely complex and difficult to estimate without intensive simulations empirically. These tools are essentially **open-loop one-shot** layout generators, lacking the iterative performance optimization process and relying on human experience to consider all layout dependent effects.

On the other hand, recent advancements in analog circuit sizing have shown significant improvements in effectively utilizing simulations for performance optimizations. Simulation-based approaches treat analog sizing as a black-box optimization problem [7], where the circuit performance are obtained by querying the circuit simulator. Compared with model-based approaches that require an understanding of the intricate design performance trade-offs [8], simulation-based methods rely on little prior knowledge on the circuit designs. Gaussian process (GP) based Bayesian optimization (BO) [9], [10] effectively combines both approaches and has demonstrated a significant reduction in simulation cost and flexibility across different designs. The success in analog circuit sizing demonstrates the importance of having performance simulations as feedback to the design process and motivates us to adopt a similar approach in developing a closed-loop analog layout synthesizer.

In this paper, we propose a **closed-loop** hierarchical analog layout synthesizer with multi-objective Bayesian optimization. In contrast to prior work [11], we assume that the circuit is well-designed and only limit the exploration to different layout implementations. Our approach efficiently utilizes post layout simulations to improve the layout implementations iteratively. We demonstrate the effectiveness of our approach at both the building block and system-level designs. The framework is fully automated and capable of handling different designs as input, with minor cost in setting up the simulation test benches. Our main contributions are summarized as follows:

- We present a closed-loop hierarchical analog layout synthesizer capable of handling system designs.
- We ensure the post layout performance of building block circuits with a design specific exploration strategy assisted by multi-objective Bayesian optimization.
- We incorporate automatic symmetry constraint detection both between devices and subcircuits.
- We integrate analytical placement with analog routing, enforcing the extracted symmetry constraints.
- We expose parameters in the synthesis flow for human-guided system-level design optimization.
- The entire framework is open-source.[1]

The rest of this paper is organized as follows: Section II gives the background on performance optimization and multi-objective Bayesian optimization; Section III explains the implementation of the proposed closed-loop hierarchical layout synthesis framework; Section IV demonstrates the experimental result on both building block and system level designs; Section V concludes the paper.

II. Background and Preliminaries

In this section, we first give the problem formulation for post layout performance optimization in Sec. II-A. Then we give a brief

[1] https://github.com/magical-eda/MAGICAL

overview of Gaussian process regression in Sec. II-B and multi-objective Bayesian optimization in Sec. II-C.

A. Problem Formulation

Prior works of analog layout synthesis [12] attempt to guarantee post layout performance by finetuning the circuit sizing parameters or changing the circuit topology. The recent work of Hakhamaneshi et al. [11] leverages deep neural network and evolutionary algorithm to optimize circuit sizing with simulation results and layout generated from BAG [1].

In contrast to these prior works, we assume that the circuit is already well-designed with thorough consideration. We focus primarily on the physical design process of generating layouts attempting to achieve performance comparable to schematic simulations. Instead of weighting and aggregating different performance costs, we adopt the formulation of a multi-objective optimization problem:

$$\min(f_1(x), \cdots, f_m(x)), \tag{1}$$

where $f_i(x)$ represent the post layout performance metric obtained through simulation and $x \in \mathbf{R}^d$ are the design specific layout parameters in Sec. III-D.

B. Gaussian Process Regression

Gaussian process [13] is a non-parametric statistical surrogate model used to expedite the optimization of computational expensive systems. Gaussian process regression (GPR) places a GP prior over the latent function with a mean function $m(x)$ and covariance kernel function $k(x, x')$.

In our setting, we use the constant mean function $m(x) = \mu_0$. We select the kernel function as the Radial Basis Function:

$$k_{RBF}(x, x') = \sigma^2 \exp(-\sum_{j=1}^{d} \frac{(x_j - x'_j)^2}{2l_j^2}), \tag{2}$$

where hyperparameters σ^2 and l_j are the signal variance and length-scales along the jth dimension. By giving the training data $D \in \{X, y\}$, the kernel function hyperparameters $\theta = \{\sigma, l_j\}$ could be obtained by maximizing the marginal log likelihood function $L(\theta)$:

$$L(\theta) = -\frac{1}{2} y^\intercal K_\theta^{-1} y - \frac{1}{2} \log |K_\theta| - \frac{n}{2} \log 2\pi, \tag{3}$$

where K_θ is the covariance matrix between all possible pairs of (x, x') in the data set.

Subsequently, given a new data point x^*, the prediction mean and variance are derived as follows:

$$\mu(x^*) = \mu_0 + k(x^*, X) K_\theta^{-1} y, \tag{4}$$

$$\sigma(x^*)^2 = \sigma^2 - k(x^*, X) K_\theta^{-1} k(X, x^*). \tag{5}$$

In contrast to parametric models with only predictive scalar outputs, GPR also allows the model to measure uncertainty in the terms of $\sigma(x^*)$. This allows more efficient trade-offs between exploration versus exploitation in surrogate model based optimization algorithms.

C. Multi-Objective Bayesian Optimization

Bayesian optimization is a black-box optimization method, efficient for handling expensive evaluation objectives over continuous domains. The key ingredients of BO include a probabilistic surrogate model of the objective function and an acquisition function for deciding the next sample point while balancing between exploration and exploitation. A commonly used surrogate model is the GP in Sec.II-B, which predicts both the mean and variance of the objective across the optimization domain. A simple yet widely used acquisition function for single objective BO is the probability of improvement over the incumbent target τ:

$$PI(x) = \int_{-\infty}^{\tau} \phi(y) dy, \tag{6}$$

where $\phi(y)$ is the posterior probability density function (PDF), in the case of GP is $N(\mu(x), \sigma(x)^2)$.

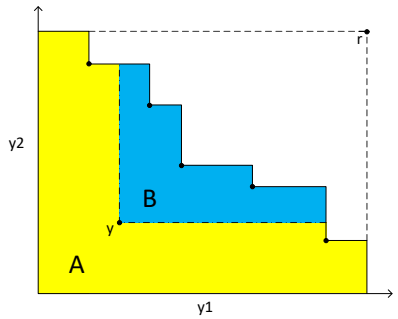


Fig. 1: Non-dominant region and exclusive hypervolume.

In the case of multi-objective Bayesian optimization (MOBO), the acquisition function needs to be augmented to assess the improvement over a Pareto set. We utilize hypervolume-based probability of improvement [14] as the acquisition function for quantitatively measuring the expected improvement:

$$PI_{hv}(x) = \int_{y \in A} I(y, P) \prod_{i=1}^{m} \phi_i(y_i) dy_i, \tag{7}$$

where A is the non-dominated region, $I(y, P)$ is the exclusive hypervolume [15], and $\phi_i(y_i) \sim N(\mu_i(x), \sigma_i(x)^2)$ is the PDF for the ith objective function. As shown in Fig. 1, the colored region A denote the non-dominate region of the Pareto set bounded by a reference point r. The blue-colored region B indicates the exclusive hypervolume of y relative to the Pareto set. The benefit of PI_{hv} as the acquisition function is that the evaluation of multiple objectives could be concentrated to a single closed-form representation.

III. Hierarchical Analog Layout Synthesis

In this section, we briefly explain the proposed hierarchical analog layout synthesis methodology. In this paper we mainly focus on the synthesis framework and refer the readers to our prior work [16], [17] for implementation details. We first explain the bottom-up design methodology in Sec. III-A. The automatic symmetry constraint detection, analog placement and routing are briefly explained in Sec III-B and Sec. III-C. Section III-D summarizes the parameters of the layout generation process. Finally, the closed-loop layout synthesis framework is presented in Sec. III-E.

A. Hierarchical Design Methodology

Human designers, as well as prior works on analog sizing [12] adopt a top-down design methodology, where system-level performance are translated to lower-level building block specifications. Thus most schematics of analog systems naturally contain hierarchy information, which largely captures the designers' intent.

The layout design process, on the other hand, should be bottom-up, where the building block circuits layout needs to be completed before the top-level layout could be implemented. Though several works have documented similar approaches for analog layout synthesis [3], [12], most of these works focus on further decomposing building

block circuits (amplifiers) into smaller subcircuits (e.g. differential input, bias circuits, and cascade topology). To the best of our knowledge, no prior work on analog layout synthesis has demonstrated success on generic analog system-level designs.

We adopt the bottom-up design methodology for analog layout synthesis. The hierarchical analog layout synthesis flow is shown in Algorithm 1. After parsing the circuit netlist, the hierarchy tree is constructed, with nodes representing sub-block circuits, and the edges indicating dependencies. The layout for the system is generated by recursively visiting the hierarchy tree and clearing layout dependencies. In this approach, the top-level layouts would always be implemented after all its dependent circuit layouts are completed. In addition to determining dependencies, the sub-blocks are automatically labeled into the following types:

- **Devices:** Transistors, resistors, and capacitors.
- **Standard Cells:** Digital standard cells.
- **Building Blocks:** Circuits consisting only of devices.
- **Macro Blocks:** Circuits that reference other building blocks or macro blocks.

The device and standard cell layouts could be obtained with a PCell generator or directly from the technology process design kit (PDK). Building blocks and macro blocks have to be placed and routed with different considerations of symmetry constraints, which will be explained in detail in the following sections.

Algorithm 1 Hierarchical Analog Layout Synthesis

Input: Hierarchical Netlist N
Output: System Layout L
1: Generate hierarchy tree T
2: Node $v \leftarrow T.root$
3: LayGen(v,T)
4: **function** `LayGen`(v,T)
5: **if** v is device or standard cell **then**
6: Obtain layout L_v from PDK
7: **else**
8: **for** u in $v.children$ **do**
9: **if** L_u is not implemented **then**
10: $L_u \leftarrow$ LayGen(u,T)
11: Generate symmetry constraints
12: Place and route L_v
13: **return** Layout L_v
14: **end function**

B. Symmetry Constraint Detection

Symmetry constraints are one of the most essential and widely adopted constraints applied during analog layout synthesis. Since the characteristics of symmetry among devices and between building blocks are vastly different, we use different methods to generate symmetry constraints for building blocks and macro blocks. We classify symmetry constraints into two categories and briefly discuss our constraint detection method for each type.

1) Device Symmetry: Only the symmetry constraints between devices need to be considered for building blocks. We adopt a method similar to the works of [18]. The building block circuit is abstracted into a graph. A pattern library of the commonly used differential topology of transistors is predefined. We use graph isomorphic algorithms to detect matching patterns on the building block circuits with the pattern library. The circuit graph is then traversed from the matched patterns to recognize new symmetry constraints of passive devices and self symmetry devices. The symmetry between bias circuits and net symmetry are also generated during graph traversal. The graph traversal step is analogous to following the differential current in the small-signal analysis.

2) System Symmetry: We define system symmetry as the constraint between two building blocks or macro blocks. System symmetry differs from device symmetry because on the system level, template libraries are difficult to generate and graph isomorphic algorithms are expensive. Since the same building block could be referenced multiple times in system design, we propose to extract graphs that include the neighboring circuit topology of the building blocks to resolve the ambiguity. The extracted graphs are then compared using an efficient graph similarity metric leveraging spectral graph analysis [17]. The ambiguity between building blocks could be resolved by comparing the similarity metrics of detected symmetry pairs. Self symmetry constraints and symmetry net constraints could also be extracted similar to the approach in device symmetry.

C. Analog Placement and Routing

The layout generation for building blocks and macro blocks rely on placement and routing to generate the final layout implementation. Placement would require the bounding box shape of each component and connection information to optimize for the wirelength objective. Routing only requires the placement results, net pin location, and net connection information. The placement and routing solution should also adhere to the symmetry constraints generated in Sec. III-B. We use the same place and route engine for both building block and macro block, since the physical design process is irrelevant to the circuit design and labeled type.

The placement engine consists of global placement followed by legalization. The analytical global placement is similar to [16], which minimizes:

$$Objective = \sum_i \alpha_i \cdot f_{WL_i} + \beta \cdot f_{AREA}, \tag{8}$$

where f_{WL_i} is the wirelength for net i and f_{AREA} is the penalty term for layout area. The legalization step enforces symmetry constraints, non-overlapping of cells, and spacing design rules.

The routing engine utilizes a sequential symmetry-aware grid-based A* search algorithm [19]. Unlike digital routers, analog routing does not enforce the routed nets to align to routing tracks. The design rules of metal wires are checked during the A* search. The symmetry constraints are enforced by mirroring the routing solutions respect to the symmetry axis. The final GDSII layout is generated after routing.

D. Parameters Summary

In this section, we summarize the parameters of the hierarchical analog layout synthesis flow. We divide the parameters into design parameters, which could change according to the design and hyper-parameters that are design independent. These parameters could be either automatically determined by MOBO for building blocks, or fine-tuned by human for system designs.

1) Design Parameters: The most important design parameters are the net weighting α_i for the global placement objective in Equation 8. Different net weighting combinations could significantly change the floor plan and placement solution and result in different post layout performance. A net with larger weight would pull the connected devices closer in the layout and have smaller parasitics. The routed wire widths of each net is also a design dependent parameter, which allows trade-offs between the parasitic resistance and capacitance. The sequence of net routing is also important since late routed nets tend to have detours to avoid early routed nets.

2) Hyperparameters: Hyperparameters are independent of design but affect the behavior of placement and routing. This could be β in the Equation 8 measuring how aggressive the placement minimizes the area. Net spacing is also important since a large spacing tends to result in longer wirelength but less coupling interference.

E. Closed-loop Layout Synthesis Framework

We present a closed-loop analog layout synthesis framework by leveraging multi-objective Bayesian optimization. We select the net weighting design parameters α_i in the global placement optimization objective as the search domain. We only select symmetry and self symmetry nets detected in the Sec. III-B as optimization candidates, while setting the weights of other non-critical nets to 1. This design specific exploration strategy could significantly change the layout implementation, leading to varying post layout performance results.

Algorithm 2 Multi-objective Bayesian Optimization

Input: Sampled data $x^t, \{f_i(x^t)\}_{i=1}^m$
Output: Next net weighting x^{t+1}
Output: Stopping criteria $meetCriteria$
Initialize Pareto set P to $\emptyset$
$meetCriteria \leftarrow false$
function MOBO(x^t,$\{f_i(x^t)\}_{i=1}^m$)
  if $t < n_{random}$ **then**
    Random sample x^{t+1} from search domain
  else
    if $\{f_i(x^t)\}_{i=1}^m$ meet requirement or $t > n_{limit}$ **then**
      $meetCriteria \leftarrow true$
    Adjust Pareto set P and reference point r
    Update GP models for each objective $f_i(x)$
    Optimize $PI_{hv}(x)$ based on GP, P, and r
    Select next net weighting x^{t+1}
  return x^{t+1}, $meetCriteria$
end function

The multi-objective Bayesian optimization algorithm is presented in Algorithm 2. During each iteration, new performance results $\{f_i(x^t)\}_{i=1}^m$ are obtained from post layout simulations. The Pareto set P and reference point r is adjusted if needed. The GP model is updated with the new data point with GPR in Sec. II-B. Finally, the next net weighting x^{t+1} is determined by maximizing $PI_{hv}(x)$ with a two-staged optimizer of Monte-Carlo optimization and L-BFGS.

The entire framework is shown in Fig. 2. The layout synthesis flow is fully automated and capable of handling different input design netlists, with minor cost in setting up the simulation test benches. The automated closed-loop layout synthesis is shown in Algorithm 3. The parasitic extractions and simulations are automated and conducted with industrial-level design tools. The MOBO algorithm treats the layout generation and performance simulation as black-box functions, iteratively updates the GP model, and generates new net weighting for the layout synthesizer.

IV. Experimental Results

We implement the proposed hierarchical flow with symmetry constraint generation in Python. The analog routing and placement algorithms are implemented in C++. The MOBO algorithm is based on GPflow [20]. All designs are in TSMC 40nm technology, extracted for parasitics with Calibre PEX, and simulated with Cadence Spectre. All experiments were performed on a Linux workstation with Intel 3.4GHz i7-3770 CPU and 32GB memory.

Algorithm 3 Closed-Loop Analog Layout Synthesis

Input: Netlist N
Input: Symmetric Constraints C
Output: Layout L
$meetCriteria \leftarrow false$
Initialize net weighting x^0 and $t = 0$
while $meetCriteria$ is $false$ **do**
  $L_p \leftarrow$ Place(N, C, x^t)
  $L_r \leftarrow$ Route(L_p, N, C)
  $N_{ext} \leftarrow$ ParasiticExtraction(L_r, N)
  $\{f_i(x^t)\}_{i=1}^m$ = Simulation(N_{ext})
  x^{t+1}, $meetCriteria$ = MOBO(x^t,$\{f_i(x^t)\}_{i=1}^m$)
  $t \leftarrow t + 1$
Obtain optimized layout L from Pareto set P
return Layout L

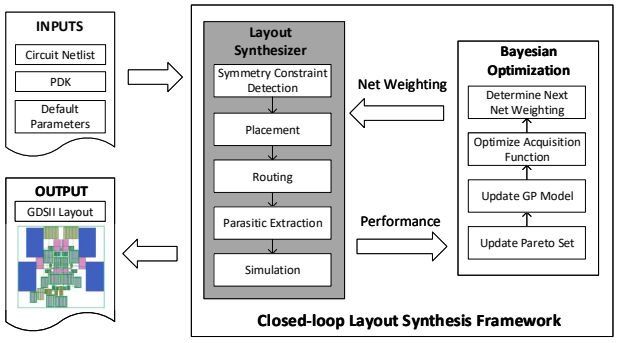


Fig. 2: Closed-loop layout synthesis.

We demonstrate the effectiveness of our framework with two examples. Section IV-A is a two-stage operational amplifier design, and the performance are optimized fully automated with the proposed MOBO algorithm. Section IV-B is a complex system design of a continuous-time $\Delta\Sigma$ modulator. We demonstrate that without closing the design loop for its building blocks, the performance of the system would severely degrade. We also identify the key issue of long simulation time in system designs. By embedding human design experience with little manual overhead, we were able to further decrease the distortion and improve the performance of the CTDSM.

A. Two-Stage Operational Amplifier

In this section, we demonstrate the fully automated layout synthesis flow with a two-stage operational amplifier. The circuit is miller compensated and includes a common-mode feedback loop, as shown in Fig. 3. The design contains 36 devices in total, with 14 design parameters of critical net weighting selected according to Sec. III-E. For our MOBO, the number of initial random samples is set to 20, and the maximum number of simulations is limited to 200.

We first demonstrate the result jointly optimizing the common-mode rejection ratio (CMRR) and absolute input-referred offset voltage. These two objectives are selected since they are affected much more by the layout implementation than the design sizing. We compare the MOBO algorithm with random sampling. The comparison of the obtained Pareto set is shown in Fig. 4(a). Although random sampling obtains a better Pareto set in 50 iterations compared with MOBO, little improvement is achieved by increasing the number of iterations to 200. It could be observed that better Pareto sets could be obtained with more exploration of the MOBO algorithm that outperforms random sampling.

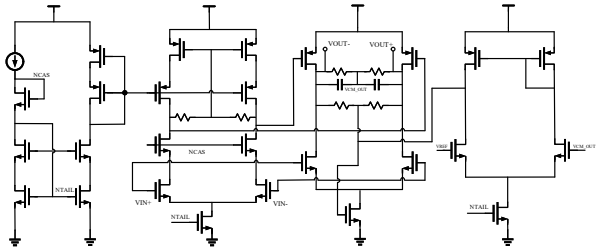
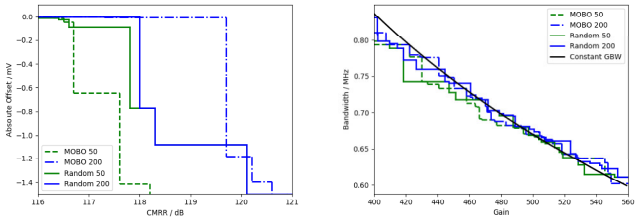


Fig. 3: Schematic of two-stage operational amplifier.

We also observe that the implemented layouts suffer degradation of bandwidth and shift in phase margin from the schematic. This is mainly due to the layout parasitics. In Fig. 4(b), we plot the Pareto set for joint optimizing the gain and bandwidth. We observe that the Pareto set has a constant gain bandwidth product (GBW) and that almost no improvement is made by design exploration. This suggests that for performance highly correlated with circuit design and parasitics, the improvement gained from optimization is severely limited. These performance are largely bounded by the circuit design and layout parasitics from suboptimal placement and routing.

After the Pareto set is obtained, the designer could conduct more detailed simulations on the implemented layout and select a layout design that meets the requirement. A random implementation would be selected from the Pareto set otherwise. The final selected layout performance is shown in Table I. We also show a particularly bad layout result one-shot generated with default parameters.



(a) (b)

Fig. 4: Pareto sets from MOBO. (a) CMRR and Offset. (b) Gain and Bandwidth.

TABLE I: Two-Stage Operational Amplifier

	Schematic	Optimized Layout	One-shot Layout
Gain (dB)	54.0	54.1	32.8
GBW (MHz)	487	344	335
Phase Margin	50.6°	67.7°	14.5°
CMRR (dB)	–	119.7	56.4
Offset (mV)	–	0.007	4.8

B. Continuous-time ΔΣ Modulator

In this section, we demonstrate our hierarchical layout synthesis on a 2nd order continuous-time ΔΣ modulator. The system architecture is shown in Fig. 5. The first stage operational amplifier used is the design in Sec. IV-A. The second stage amplifier has the same circuit topology but slightly modified sizing. In total, this design consists of 177 devices with 6 digital standard cells.

We demonstrate the severe limitations of simulation-based optimization approach on system-level designs. Table II shows the

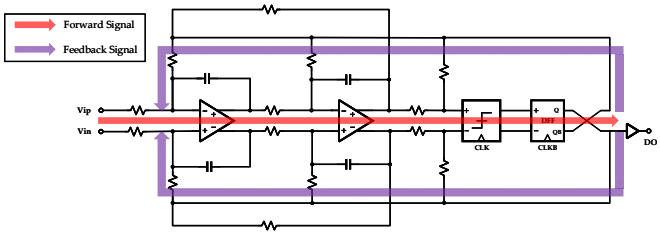


Fig. 5: Schematic of 2nd Order CTDSM.

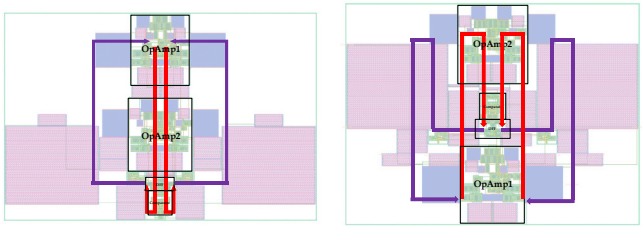
runtime for different design complexity. It could be observed that the simulation time becomes unmanageable for system designs. As an example of this CTDSM design, an 8192 point Fast Fourier Transform (FFT) needs to be conducted to characterize its performance. This results in a long and expensive transient simulation. For post layout performance, costly transient simulations have to be done at the transistor level to capture the layout dependent effects, such as well proximity effects, signal integrity, and clock coupling.

TABLE II: Runtime Comparison

	Layout	Extraction	Simulation
Amplifier	18s	34s	26s
CTDSM	1m14s	41s	1h59m25s

In our experiment, we optimize the building block level circuits, mainly the two operational amplifiers, with the proposed closed-loop layout synthesis framework. Since repetitive simulations for the CTDSM system is unaffordable, we attempt to optimize the system performance by embedding human experience into the layout generation process. We consider signal integrity and clock coupling at the system level and make minor adjustments to the placement and routing algorithms by only changing the parameters of the layout generation process introduced in Sec. III-D. The details of our heuristic implementations are as follows:

- **Net Weighting:** We increase the weight of the critical nets on the forward signal chain shown in Fig. 5. This will cause the layout synthesizer to place the building blocks according to the signal flow.
- **Routing Sequence:** We prioritize routing clock nets first and power/ground the last. This will create shorter clock routing and less coupling to signal nets.
- **Routing Spacing:** We increase the routing metal spacing for the top-level system routing. The increased spacing would decrease the coupling between nets.



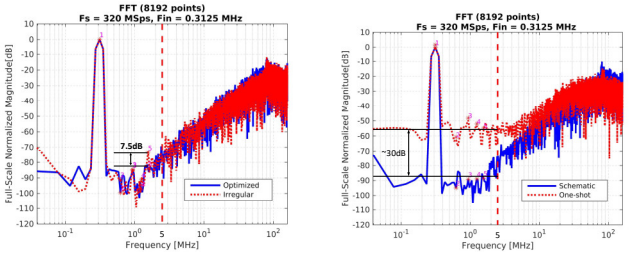
(a) (b)

Fig. 6: CTDSM layout implementations. (a) Optimized signal flow. (b) Irregular signal flow.

Figure 6 shows the comparison of two different layout implementations. The forward signal path is highlighted in red, and the

feedback path is in purple. By changing the weights of the parameters of the layout generation process, we can embed the designers' understanding of the system and affect the layout implementation. The simulation result comparison is shown in Fig. 7(a). We can observe that the layout with optimized signal flow has lower harmonic distortion and 7.5dB of improvement in the spurious-free dynamic range (SFDR) compared with the layout with irregular signal flow. We believe this is largely due to mitigated clock coupling to critical signal nets in the forward and feedback paths.

To further demonstrate the importance of having a closed-loop layout synthesizer, we implement a one-shot generated layout without optimizing for the building block circuits performance. The first stage operational amplifier is the one-shot layout shown in Table I in Sec. IV-A. Without the guarantee of building block circuit performance, the system design performance would significantly degrade. The simulation result is shown in Fig. 7(b).



(a) (b)

Fig. 7: Simulation results. (a) Optimized vs irregular signal flow. (b) Open-loop layout.

Table III shows the CTDSM performance comparisons between different implementations based on simulations. One-shot and Irregular are layouts generated fully automatically with no human-in-the-loop. Schematic refers to the schematic design. One-shot is the layout design without closed-loop optimization of the building blocks. Irregular is the layout with performance optimization and default parameter settings in the synthesis flow, resulting in irregular signal paths. Optimized is the layout with human embedded experience and optimized signal flow paths. We can observe that having a close-loop layout synthesis flow to guarantee building block circuits performance is critical in ensuring the successful implementation of system-level layout. By embedding human experience in finetuning the parameters, we can further boost the performance with 7.5dB improvement in the SFDR. Our synthesized layout achieves SNDR of 65.9dB compared to the schematic design of 67.8dB.

TABLE III: CTDSM Performance

	Schematic	One-shot	Irregular	Optimized
Supply (V)	1.2			
F_s (MHz)	320			
BW (MHz)	5			
SNDR (dB)	67.8	39.6	65.2	65.9
SFDR (dB)	84.7	48.5	73.0	80.5
Power (mW)	0.84	0.91	0.86	0.86
Area (um^2)	–	8094	8188	9450

V. Conclusion

In this work, we present a closed-loop hierarchical analog layout synthesis framework. The framework is open-source, fully automated, and capable of handling different designs as input. The performance for building block circuits are guaranteed by leveraging post layout simulations and multi-objective Bayesian optimization. We demonstrate the high cost of simulation in system designs and further optimize the system performance by embedding human experience. Our obtained CTDSM achieves a post layout performance of 65.9dB in SNDR, compared with 67.8dB in the schematic design.

Acknowledgement

This work is supported in part by the NSF under Grant No. 1704758, and the DARPA ERI IDEA program.

References

[1] J. Crossley, A. Puggelli, H. . Le, B. Yang, R. Nancollas, K. Jung, L. Kong, N. Narevsky, Y. Lu, N. Sutardja, E. J. An, A. L. Sangiovanni-Vincentelli, and E. Alon, "Bag: A designer-oriented integrated framework for the development of ams circuit generators," in *ICCAD*, Nov 2013, pp. 74–81.

[2] V. Meyer zu Bexten, C. Moraga, R. Klinke, W. Brockherde, and K. . Hess, "Alsyn: flexible rule-based layout synthesis for analog ic's," *JSSC*, vol. 28, no. 3, pp. 261–268, March 1993.

[3] J. M. Cohn, D. J. Garrod, R. A. Rutenbar, and L. R. Carley, "Koan/anagram ii: New tools for device-level analog placement and routing," *JSSC*, vol. 26, no. 3, pp. 330–342, 1991.

[4] K. Kunal, M. Madhusudan, A. K. Sharma, W. Xu, S. M. Burns, R. Harjani, J. Hu, D. A. Kirkpatrick, and S. S. Sapatnekar, "Invited: Align – open-source analog layout automation from the ground up," in *DAC*, June 2019.

[5] H.-C. Ou, K.-H. Tseng, J.-Y. Liu, I. Wu, and Y.-W. Chang, "Layout-dependent-effects-aware analytical analog placement," in *DAC*, 2015.

[6] K. Lampaert, G. Gielen, and W. M. Sansen, "A performance-driven placement tool for analog integrated circuits," *JSSC*, vol. 30, no. 7, pp. 773–780, 1995.

[7] B. Liu, F. V. Fernández, Q. Zhang, M. Pak, S. Sipahi, and G. Gielen, "An enhanced moea/d-de and its application to multiobjective analog cell sizing," in *IEEE Congress on Evolutionary Computation*, July 2010.

[8] M. del Mar Hershenson, "Design of pipeline analog-to-digital converters via geometric programming," in *ICCAD*, Nov 2002, pp. 317–324.

[9] W. Lyu, F. Yang, C. Yan, D. Zhou, and X. Zeng, "Multi-objective bayesian optimization for analog/rf circuit synthesis," in *DAC*, 2018.

[10] S. Zhang, W. Lyu, F. Yang, C. Yan, D. Zhou, X. Zeng, and X. Hu, "An efficient multi-fidelity bayesian optimization approach for analog circuit synthesis," in *DAC*, 2019.

[11] K. Hakhamaneshi, N. Werblun, P. Abbeel, and V. Stojanović, "Late breaking results: Analog circuit generator based on deep neural network enhanced combinatorial optimization," in *DAC*, 2019.

[12] G. G. E. Gielen and R. A. Rutenbar, "Computer-aided design of analog and mixed-signal integrated circuits," *Proceedings of the IEEE*, vol. 88, no. 12, pp. 1825–1854, Dec 2000.

[13] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. The MIT Press, 2005.

[14] I. Couckuyt, D. Deschrijver, and T. Dhaene, "Fast calculation of multi-objective probability of improvement and expected improvement criteria for pareto optimization," *Journal of Global Optimization*, vol. 60, no. 3, pp. 575–594, 2014.

[15] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. G. da Fonseca, "Performance assessment of multiobjective optimizers: an analysis and review," *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 2, pp. 117–132, April 2003.

[16] B. Xu, K. Zhu, M. Liu, Y. Lin, S. Li, X. Tang, N. Sun, and D. Z. Pan, "Magical: Toward fully automated analog ic layout leveraging human and machine intelligence," in *ICCAD*, 2019.

[17] M. Liu, W. Li, K. Zhu, B. Xu, Y. Lin, L. Shen, X. Tang, N. Sun, and D. Z. Pan, "S3det: Detecting system symmetry constraints for analog circuits with graph similarity," in *ASPDAC*, Jan 2020.

[18] M. Eick, M. Strasser, K. Lu, U. Schlichtmann, and H. E. Graeb, "Comprehensive generation of hierarchical placement rules for analog integrated circuits," *IEEE TCAD*, vol. 30, no. 2, pp. 180–193, Feb 2011.

[19] K. Zhu, M. Liu, Y. Lin, B. Xu, S. Li, X. Tang, N. Sun, and D. Z. Pan, "Geniusroute: A new analog routing paradigm using generative neural network guidance," in *ICCAD*, Nov 2019.

[20] A. G. d. G. Matthews, M. van der Wilk, T. Nickson, K. Fujii, A. Boukouvalas, P. León-Villagrá, Z. Ghahramani, and J. Hensman, "GPflow: A Gaussian process library using TensorFlow," *Journal of Machine Learning Research*, vol. 18, no. 40, pp. 1–6, Apr 2017.