FALCON: An ML Framework for Fully Automated Layout-Constrained Analog Circuit Design

Asal Mehradfar[1] **Xuzhe Zhao**[2] **Yilun Huang**[2] **Emir Ceyani**[1]
Yankai Yang[2] **Shihao Han**[2] **Hamidreza Aghasi**[2] **Salman Avestimehr**[1]
[1]University of Southern California [2]University of California, Irvine
mehradfa@usc.edu

Abstract

Designing analog circuits from performance specifications is a complex, multi-stage process encompassing topology selection, parameter inference, and layout feasibility. We introduce FALCON, a unified machine learning framework that enables fully automated, specification-driven analog circuit synthesis through topology selection and layout-constrained optimization. Given a target performance, FALCON first selects an appropriate circuit topology using a performance-driven classifier guided by human design heuristics. Next, it employs a custom, edge-centric graph neural network trained to map circuit topology and parameters to performance, enabling gradient-based parameter inference through the learned forward model. This inference is guided by a differentiable layout cost, derived from analytical equations capturing parasitic and frequency-dependent effects, and constrained by design rules. We train and evaluate FALCON on a large-scale custom dataset of 1M analog mm-wave circuits, generated and simulated using Cadence Spectre across 20 expert-designed topologies. Through this evaluation, FALCON demonstrates >99% accuracy in topology inference, <10% relative error in performance prediction, and efficient layout-aware design that completes in under 1 second per instance. Together, these results position FALCON as a practical and extensible foundation model for end-to-end analog circuit design automation. Our code and dataset are publicly available at https://github.com/AsalMehradfar/FALCON.

1 Introduction

Analog radio frequency (RF) and millimeter-wave (mm-wave) circuits are essential to many modern technologies [1–5]. Yet their design remains manual and heuristic-driven [6–8], challenged by a vast continuous design space, tightly coupled trade-offs (e.g., gain, noise, bandwidth, power), and layout-dependent interactions. As demand for high-performance analog blocks grows, this slow, expert-driven cycle has become a bottleneck. While ML has transformed digital design, analog efforts remain fragmented—focusing on isolated tasks like topology generation or sizing [9, 10], assuming fixed topologies [11–14], relying on non-scalable black-box optimization [15], or predicting performance without enabling inverse design [16]. Layout is typically handled in post-processing [17], and many benchmarks rely on symbolic or synthetic simulations [18], limiting realism. No current ML pipeline supports fully generalizable, layout-aware, end-to-end analog circuit design.

We propose FALCON (Fully Automated Layout-Constrained analOg circuit desigN), a scalable ML framework for end-to-end analog and RF circuit design. Trained on over one million Cadence-simulated circuits, FALCON integrates three stages (Figure 1): (1) a multilayer perceptron (MLP) selects the topology from target specifications; (2) a graph neural network (GNN) maps topology and parameters to performance on netlist-derived graphs; and (3) gradient-based optimization over the GNN recovers parameters that satisfy targets under a differentiable layout-aware loss. The GNN

Machine Learning and the Physical Sciences Workshop, NeurIPS 2025.

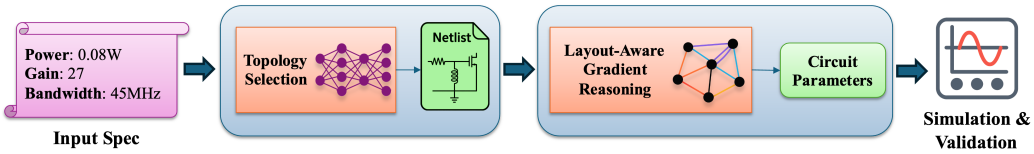


Figure 1: Our AI-based circuit design pipeline. Given a target performance specification, FALCON first selects a suitable topology, then generates design parameters through layout-aware gradient-based reasoning with GNN model. Then, the synthesized circuit is validated using Cadence simulations.

generalizes to unseen topologies with optional fine-tuning, enabling inverse design across circuit families. By embedding layout constraints directly in optimization, FALCON unifies schematic and physical considerations in a single differentiable framework.

2 Related Work

Prior ML approaches for analog design focus on specific stages without enabling unified, differentiable synthesis. **Topology generation** methods use generative models [9, 19, 20] to explore the circuit graph space, but often target low-frequency or simplified designs [9] and may yield non-manufacturable structures. **Parameter sizing and performance prediction** have been addressed via reinforcement learning [10, 21], Bayesian optimization [15, 22], and supervised learning [23, 24, 11], typically under fixed topologies. Graph models [16] improve performance regression by incorporating schematic structure but lack support for inverse design. **Layout-aware methods** either integrate parasitic estimators into black-box optimization [25] or generate layouts from fixed netlists [26, 27], without enabling co-optimization. **Datasets** rely on symbolic or synthetic simulations [13, 18], limiting realism. In contrast, FALCON unifies all stages by selecting expert-designed netlists, enabling forward and inverse modeling via a generalizable GNN, embedding layout constraints directly into the loss, and training on over one million Cadence-simulated datapoints across 20 topologies and five circuit families. To the best of our knowledge, *FALCON is the first framework to unify topology selection, parameter inference, and layout-aware optimization in a single pipeline, validated at scale on mm-wave circuits using foundry-calibrated simulations.*

3 A Large-Scale Dataset and Inverse Design Problem Formulation

Dataset Overview. We construct a large-scale dataset of analog and RF circuits using industry-grade Cadence tools [28] and a 45nm CMOS PDK. Covering five widely used mm-wave circuit types, the dataset includes four expert-designed topologies per category, totaling 20 architectures [29, 30]. Each of the over one million datapoints contains a full circuit parameter vector, a Cadence netlist, and simulated performance metrics, capturing a wide spectrum of behaviors and trade-offs. More details are available in Appendix A.

Inverse Design. Traditional analog circuit design follows a forward workflow: selecting a topology T and parameters x, then simulating to obtain performance $y = f(T, x)$. This process is manual and costly. Instead, inverse design aims to infer (T, x) directly from a target specification y_{target}, a challenging and ill-posed task due to non-invertibility of the true performance function f, modeled by expensive Cadence simulations. FALCON tackles this via a modular, three-stage pipeline:

Stage 1: Topology Selection. We frame topology selection as a classification problem over a curated set of K candidate topologies $\{T_1, \ldots, T_K\}$. Given a target specification y_{target}, a lightweight MLP selects the topology $T^* \in \mathcal{T}$ most likely to satisfy it, reducing the need for exhaustive search.

Stage 2: Performance Prediction. Given a topology T and parameter vector x, we train a GNN f_θ to predict the corresponding performance $\hat{y} = f_\theta(T, x)$. This model emulates the forward behavior of the simulator f, learning a continuous approximation of circuit performance across both seen and unseen topologies. By capturing the topology-conditioned mapping from parameters to performance, f_θ serves as a differentiable surrogate that enables gradient-based inference in the next stage.

Stage 3: Layout-Aware Gradient Reasoning. Given y_{target} and a selected topology T^*, we infer a parameter vector x^* by minimizing a loss over the learned forward model f_θ. Specifically, we solve:

$$x^* = \arg\min_x \ \mathcal{L}_{\text{perf}}(f_\theta(T^*, x), y_{\text{target}}) + \lambda \mathcal{L}_{\text{layout}}(x), \tag{1}$$

where $\mathcal{L}_{\text{perf}}$ measures prediction error, and $\mathcal{L}_{\text{layout}}$ encodes differentiable layout-related constraints such as estimated area and soft design-rule penalties. Optimization is performed via gradient descent, allowing layout constraints to guide the search through a physically realistic parameter space.

4 Stage 1: Performance-Driven Topology Selection

Task Setup. We cast topology selection as a supervised classification task over 20 expert-designed, manufacturable topologies, avoiding invalid netlist generation. A lightweight MLP selects the most suitable topology given a normalized 16-dimensional performance vector, with missing metrics imputed by zeros. The dataset is stratified for class balance, and while the library is fixed during training, new topologies can be added with retraining for rapid initial design.

Model Architecture and Training. We train a 5-layer MLP with hidden size 256 and ReLU activations for this problem. The model takes the normalized performance vector as input and outputs a probability distribution over 20 candidate topologies (Figure 2). We train the model using a cross-entropy loss and the Adam optimizer [31], with a batch size of 256.

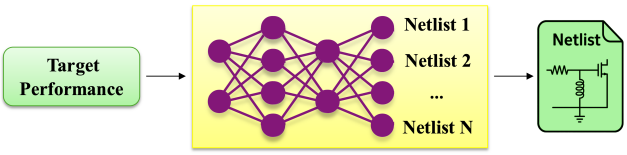


Figure 2: In Stage 1, an MLP classifier selects the most suitable circuit topology from a library of human-designed netlists, conditioned on the target performance specification.

Table 1: Classification performance on topology selection.

Metric	Score (%)
Accuracy	99.57
Balanced Accuracy	99.33
Macro Precision	99.27
Macro Recall	99.33
Macro F1	99.30
Micro F1	99.57

Evaluation. We assess classification performance using accuracy, balanced accuracy, macro precision, macro recall, macro F1, and micro F1 scores on the test set. As summarized in Table 1, the classifier achieves an overall accuracy of 99.57%, with macro F1 of 99.30% and balanced accuracy of 99.33%, demonstrating strong generalization across all topologies.

5 Stage 2: Generalizable Forward Modeling for Performance Prediction

Task Setup. Stage 2 trains a differentiable forward model to predict performance from a given topology and parameter vector. Unlike black-box simulators, this GNN model enables differentiable prediction, generalizing across circuit families with minimal fine-tuning. Since not all metrics apply to every topology, we use a **masked mean squared error** loss for this model that computes the average error only over the defined entries.

Model Architecture and Training. Each circuit is represented as an **undirected multi-edge graph** with voltage nets as nodes and circuit components as edges. All circuit parameters—both fixed and sweepable—are assigned to edges, along with categorical device types and one-hot encoded indicators. For each edge e, these attributes are concatenated to form a unified feature vector x_e. The feature set is consistent within each component type but varies across types (e.g., NMOS vs. inductor), reflecting the structure defined in Appendix B.

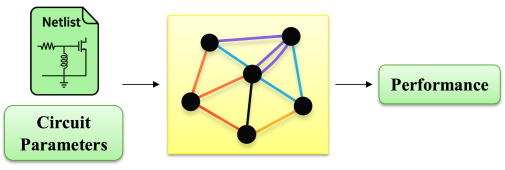


Figure 3: In Stage 2, a custom edge-centric GNN maps an undirected multi-edge graph constructed from the circuit netlist to a performance vector.

Table 2: Prediction accuracy of the forward GNN on all 16 circuit performance metrics.

Metric	DCP	VGain	PGain	CGain	S_{11}	S_{22}	NF	BW	OscF	TR	OutP	P_{SAT}	DE	PAE	PN	VSwg
Unit	mW	dB	dB	dB	dB	dB	dB	GHz	GHz	GHz	dBm	dBm	%	%	dBc/Hz	mV
R^2	1.0	1.0	0.99	1.0	0.93	1.0	0.99	0.98	0.97	0.83	0.97	1.0	1.0	1.0	0.89	1.0
RMSE	0.27	0.101	0.536	0.833	1.515	0.21	0.534	0.972	0.723	0.293	0.91	0.095	0.226	0.143	2.536	0.071
MAE	0.198	0.072	0.208	0.188	0.554	0.12	0.2	0.376	0.184	0.097	0.238	0.066	0.163	0.105	1.159	0.046
Rel. Err.	11.2%	2.6%	19.0%	6.1%	11.4%	1.9%	4.5%	6.5%	0.6%	6.5%	4.6%	4.4%	4.6%	11.0%	1.3%	1.4%

To handle component heterogeneity, we encode each edge's raw features x_e using a **type-specific MLP** $\phi_{\text{enc}}^{(t_e)}$, yielding edge embeddings z_e. Nodes are initialized with dummy scalars and projected to a hidden dimension via a shared linear layer. We then apply a **4-layer edge-centric message-passing GNN** with residual connections. At each layer, messages from incident edges are computed using a shared function ϕ_{MSG} and aggregated at each node. Node states are updated via a shared function ϕ_{UPD}, followed by residual addition and ReLU activation:

$$m_u^{(\ell)} = \sum_{e \in \mathcal{E}_u} \phi_{\text{MSG}}\left(h_{\text{src}(e)}^{(\ell)}, z_e\right), \quad h_u^{(\ell+1)} = \text{ReLU}\left(\phi_{\text{UPD}}\left(m_u^{(\ell)}\right) + h_u^{(\ell)}\right).$$

After 4 layers, node embeddings are aggregated via **global mean pooling** to obtain a fixed-size graph representation z_{graph}, which is passed to a final MLP (`output_mlp`) to predict the 16-dimensional performance vector $\hat{y} \in \mathbb{R}^{16}$. See Figure 3 for an overview.

To stabilize training, physical parameters are rescaled by their expected units (e.g. resistance by 10^3), and performance targets are normalized to z-scores using training statistics. We train the model using the Adam optimizer (learning rate 10^{-3}, batch size 256) and a `ReduceLROnPlateau` scheduler. Xavier uniform initialization is used for all layers, and early stopping is based on validation loss. We adopt the same splits as in Section 4 for consistency in evaluation.

Evaluation. We evaluate the accuracy of the GNN forward model on a test set drawn from 19 of the 20 topologies. One topology—RVCO—is entirely excluded from training, validation, and test splits to assess generalization to unseen architectures. As summarized in Table 2, the model achieves high accuracy across all dimensions, with an average R^2 of 0.972. To evaluate end-to-end prediction accuracy at the sample level, we compute the *mean relative error per instance*, defined as the average relative error across all valid (non-masked) performance metrics for each test sample. Figure 4 shows the distribution of this quantity across the test set. Without percentile trimming, the overall mean relative error across the full test set is **9.09%**.

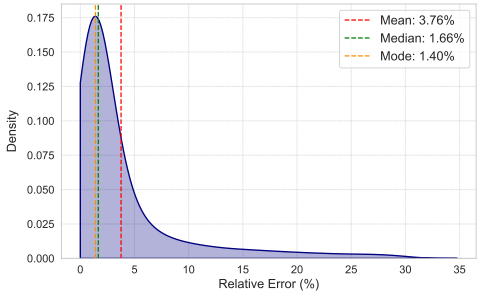


Figure 4: Distribution of relative error (%). Plot is trimmed at the 95th percentile.

Table 3: Fine-tuning results on the held-out RVCO topology. Only the output head is updated using RVCO samples.

Metric	DCP	OscF	TR	OutP	PN
Unit	W	GHz	GHz	dBm	dBc/Hz
R^2	1.0	1.0	1.0	0.97	0.98
RMSE	0.643	0.324	0.026	0.099	0.953
MAE	0.508	0.256	0.02	0.077	0.619
Rel. Err.	0.75%	0.85%	1.63%	0.69%	0.73%

Generalizing to Unseen Topologies. To assess the generalization ability of our pretrained GNN, we evaluate it on the held-out RVCO topology, which was entirely excluded from the data splits. We fine-tune the GNN by freezing all encoder and message-passing layers and updating only the final output head (`output_mlp`). Fine-tuning is performed on the RVCO training set, which contains approximately 30,000 instances, and completes in under 30 minutes on a MacBook CPU.

Even in the zero-shot setting—where the model has never seen RVCO topologies—the pretrained GNN achieves a nontrivial mean relative error of 30.4%, highlighting its strong cross-topology generalization. Fine-tuning reduces this error to just **0.9%**, demonstrating that the structural and parametric priors learned during pretraining are highly transferable. Table 3 summarizes detailed results across five key performance metrics for RVCO.

6 Stage 3: Layout-Aware Parameter Inference via Gradient Reasoning

Task Setup. Given a target performance vector and selected topology, Stage 3 aims to infer a parameter vector that minimizes a total loss combining performance error and layout-aware penalties. Optimization is initialized by sampling a plausible starting point using domain-specific scale factors (e.g., 10^{-12} for capacitors). Parameters are then iteratively refined via gradient descent, with topology-specific constants fixed and all values clipped to valid ranges to ensure physical plausibility.

Loss Function. The total loss (Eqn 1) jointly minimizes performance error and layout cost: $\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{perf}} + \lambda_{\text{area}} \cdot \mathcal{L}_{\text{layout}} \cdot g(\mathcal{L}_{\text{perf}})$, where $\mathcal{L}_{\text{perf}}$ is the masked MSE between predicted and target performance, and $\mathcal{L}_{\text{layout}}$ is a normalized area penalty. The gating term $g(\mathcal{L}_{\text{perf}}) = 1 - \sigma(\gamma(\mathcal{L}_{\text{perf}} - \tau))$ deactivates layout loss when performance error exceeds $\tau = 0.05$, prioritizing functionality. We set $\gamma = 50$, normalize area by $1\,\text{mm}^2$, and use $\lambda_{\text{area}} = 0.02$ to balance objectives. This formulation enables manufacturable parameter recovery via physics-informed learning. See Appendix C for extensions.

Layout Modeling. The layout model is deterministic and non-learned. It estimates area contributions from passive components, i.e. capacitors, inductors, and resistors, as these dominate total area and exhibit layout-sensitive behavior. For a given parameter vector x, the total layout loss is computed as: $\mathcal{L}_{\text{layout}}(x) = \sum_{e \in \mathcal{E}_{\text{passive}}} A_e(x)$, where $\mathcal{E}_{\text{passive}}$ is the set of passive elements, and $A_e(x)$ is the area of the created layout for the passive component based on analytical physics-based equations.

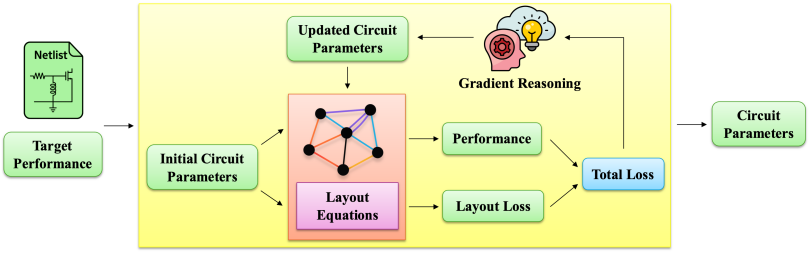


Figure 5: In Stage 3, gradient reasoning iteratively updates parameters to minimize a loss combining performance error and layout cost, computed via a differentiable analytical model.

Gradient Reasoning Procedure. Given the initialized parameter vector, we iteratively optimize using the frozen GNN forward model to compute performance, evaluate the total loss $\mathcal{L}_{\text{total}}$, and update circuit parameters via gradient backpropagation with the Adam optimizer (see Figure 5). Each instance starts with a learning rate of 10^{-6}, adjusted by a `ReduceLROnPlateau` scheduler. If performance or layout constraints are not met, we restart with a more exploratory rate. Early stopping is triggered by stagnating loss, enabling rapid convergence to valid solutions—typically in milliseconds to under one second per instance.

Evaluation. We evaluate Stage 3 on 9,500 test instances (500 per topology), defining convergence as achieving both a predicted mean relative error below 10% and a layout area under 1 mm^2 (or 1.5 mm^2 for DLNA, DohPA, ClassBPA). Success requires the final Cadence-simulated performance to deviate less than 20% from the target. Our method attains a **78.5%** success rate with **17.7%** mean relative error on converged designs, averaging **under 1 second** per instance on a MacBook CPU. Notably, success rate is coupled with the convergence threshold: tighter error bounds yield higher accuracy but require more iterations—critical for large-scale design tasks. Appendix D shows more representative results, demonstrating FALCON's unique ability to support layout-constrained optimization.

7 Conclusion and Future Work

We presented FALCON, a modular framework for end-to-end analog and RF circuit design that unifies topology selection, performance prediction, and layout-aware parameter optimization. Trained on over one million Cadence-simulated mm-wave circuits, FALCON combines a lightweight MLP, a generalizable GNN, and differentiable gradient reasoning to synthesize circuits from specification to layout-constrained parameters. FALCON achieves >99% topology selection accuracy, <10% prediction error, and efficient inverse design—all within sub-second inference. In addition, the GNN forward model generalizes to unseen topologies with minimal fine-tuning, supporting broad practical deployment. Further discussion of efficiency and limitations are provided in Appendix E.

Acknowledgments and Disclosure of Funding

We thank Andrea Villasenor and Tanqin He for their assistance with circuit data generation. We also thank Mohammad Shahab Sepehri for his insightful discussions and thoughtful feedback during the development of this work.

References

[1] Vilem Kledrowetz, Roman Prokop, Lukas Fujcik, and Jiri Haze. A fully differential analog front-end for signal processing from emg sensor in 28 nm fdsoi technology. *Sensors*, 23(7), 2023.

[2] Wei Hong, Zhi Hao Jiang, Chao Yu, Debin Hou, Haiming Wang, Chong Guo, Yun Hu, Le Kuai, Yingrui Yu, Zhengbo Jiang, Zhe Chen, Jixin Chen, Zhiqiang Yu, Jianfeng Zhai, Nianzu Zhang, Ling Tian, Fan Wu, Guangqi Yang, Zhang-Cheng Hao, and Jian Yi Zhou. The role of millimeter-wave technologies in 5g/6g wireless communications. *IEEE Journal of Microwaves*, 1(1):101–122, 2021.

[3] Yingying Chi, Haifeng Zhang, Zhe Zheng, Rui Liu, Lei Qiao, and Wenpeng Cui. Analog front-end circuit design for wireless sensor system-on-chip. In *2020 IEEE 4th Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*, volume 1, pages 38–42, 2020.

[4] Xuyang Liu, Md. Hedayatullah Maktoomi, Mahdi Alesheikh, Payam Heydari, and Hamidreza Aghasi. A cmos 49–63-ghz phase-locked stepped-chirp fmcw radar transceiver. *IEEE Journal of Solid-State Circuits*, pages 1–15, 2025.

[5] Med Nariman, Farid Shirinfar, Anna Papió Toda, Sudhakar Pamarti, Ahmadreza Rofougaran, and Franco De Flaviis. A compact 60-ghz wireless power transfer system. *IEEE Transactions on Microwave Theory and Techniques*, 64(8):2664–2677, 2016.

[6] Phillip E Allen and Douglas R Holberg. *CMOS analog circuit design*. Elsevier, 2011.

[7] Willy M. C. Sansen. *analog design essentials*. SpringerLink, 2011.

[8] Shady A Abdelaal, Ahmed Hussein, and Hassan Mostafa. A bayesian optimization framework for analog circuits optimization. In *2020 15th International Conference on Computer Engineering and Systems (ICCES)*, pages 1–4. IEEE, 2020.

[9] Zehao Dong, Weidong Cao, Muhan Zhang, Dacheng Tao, Yixin Chen, and Xuan Zhang. CktGNN: Circuit graph neural network for electronic design automation. In *The Eleventh International Conference on Learning Representations*, 2023.

[10] Hanrui Wang, Kuan Wang, Jiacheng Yang, Linxiao Shen, Nan Sun, Hae-Seung Lee, and Song Han. Gcn-rl circuit designer: Transferable transistor sizing with graph neural networks and reinforcement learning. In *2020 57th ACM/IEEE Design Automation Conference (DAC)*, pages 1–6, 2020.

[11] Dmitrii Krylov, Pooya Khajeh, Junhan Ouyang, Thomas Reeves, Tongkai Liu, Hiba Ajmal, Hamidreza Aghasi, and Roy Fox. Learning to design analog circuits to meet threshold specifications. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023.

[12] Hanrui Wang, Jiacheng Yang, Hae-Seung Lee, and Song Han. Learning to design circuits. *arXiv preprint arXiv:1812.02734*, 2018.

[13] Keertana Settaluri, Ameer Haj-Ali, Qijing Huang, Kourosh Hakhamaneshi, and Borivoje Nikolic. Autockt: deep reinforcement learning of analog circuit designs. In *Proceedings of the 23rd Conference on Design, Automation and Test in Europe*, DATE '20, page 490–495, San Jose, CA, USA, 2020. EDA Consortium.

[14] Yaguang Li, Yishuang Lin, Meghna Madhusudan, Arvind Sharma, Sachin Sapatnekar, Ramesh Harjani, and Jiang Hu. A circuit attention network-based actor-critic learning approach to robust analog transistor sizing. In *2021 ACM/IEEE 3rd Workshop on Machine Learning for CAD (MLCAD)*, pages 1–6, 2021.

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

[16] Kourosh Hakhamaneshi, Marcel Nassar, Mariano Phielipp, Pieter Abbeel, and Vladimir Stojanovic. Pretraining graph neural networks for few-shot analog circuit modeling and design. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 42(7):2163–2173, 2022.

[17] Morteza Fayazi, Morteza Tavakoli Taba, Ehsan Afshari, and Ronald Dreslinski. Angel: Fully-automated analog circuit generator using a neural network assisted semi-supervised learning approach. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 2023.

[18] Jintao Li, Haochang Zhi, Ruiyu Lyu, Wangzhen Li, Zhaori Bi, Keren Zhu, Yanhan Zeng, Weiwei Shan, Changhao Yan, Fan Yang, Yun Li, and Xuan Zeng. Analoggym: An open and practical testing suite for analog circuit synthesis. In *International Conference on Computer Aided Design*, 2024.

[19] Chen-Chia Chang, Yikang Shen, Shaoze Fan, Jing Li, Shun Zhang, Ningyuan Cao, Yiran Chen, and Xin Zhang. Lamagic: Language-model-based topology generation for analog integrated circuits. *arXiv preprint arXiv:2407.18269*, 2024.

[20] Yao Lai, Sungyoung Lee, Guojin Chen, Souradip Poddar, Mengkang Hu, David Z Pan, and Ping Luo. Analogcoder: Analog circuit design via training-free code generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 379–387, 2025.

[21] Weidong Cao, Mouhacine Benosman, Xuan Zhang, and Rui Ma. Domain knowledge-based automated analog circuit design with deep reinforcement learning. *arXiv preprint arXiv:2202.13185*, 2022.

[22] Ahmet Faruk Budak, Miguel Gandara, Wei Shi, David Z. Pan, Nan Sun, and Bo Liu. An efficient analog circuit sizing method based on machine learning assisted global optimization. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 41(5):1209–1221, 2022.

[23] Asal Mehradfar, Xuzhe Zhao, Yue Niu, Sara Babakniya, Mahdi Alesheikh, Hamidreza Aghasi, and Salman Avestimehr. AICircuit: A Multi-Level Dataset and Benchmark for AI-Driven Analog Integrated Circuit Design. *Machine Learning and the Physical Sciences Workshop @ NeurIPS*, 2024.

[24] Asal Mehradfar, Xuzhe Zhao, Yue Niu, Sara Babakniya, Mahdi Alesheikh, Hamidreza Aghasi, and Salman Avestimehr. Supervised learning for analog and rf circuit design: Benchmarks and comparative insights. *arXiv preprint arXiv:2501.11839*, 2025.

[25] Mingjie Liu, Walker J. Turner, George F. Kokai, Brucek Khailany, David Z. Pan, and Haoxing Ren. Parasitic-aware analog circuit sizing with graph neural networks and bayesian optimization. In *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1372–1377, 2021.

[26] Tonmoy Dhar, Kishor Kunal, Yaguang Li, Meghna Madhusudan, Jitesh Poojary, Arvind K Sharma, Wenbin Xu, Steven M Burns, Ramesh Harjani, Jiang Hu, et al. Align: A system for automating analog layout. *IEEE Design & Test*, 38(2):8–18, 2020.

[27] Bingyang Liu, Haoyi Zhang, Xiaohan Gao, Zichen Kong, Xiyuan Tang, Yibo Lin, Runsheng Wang, and Ru Huang. Layoutcopilot: An llm-powered multi-agent collaborative framework for interactive analog layout design. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2025.

[28] Antonio J Lopez Martin. Cadence design environment. *New Mexico State University, Tutorial paper*, 35, 2002.

[29] Sorin Voinigescu. *High-frequency integrated circuits*. Cambridge University Press, 2013.

[30] Behzad Razavi. *RF microelectronics*, volume 2. Prentice hall New York, 2012.

[31] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[32] Thomas H. Lee. *The Design of CMOS Radio-Frequency Integrated Circuits*. Cambridge University Press, 2nd edition, 2004.

[33] B. Henderson and E. Camargo. *Microwave Mixer Technology and Applications*. Microwave & RF. Artech House, 2013.

[34] Hua Wang, Peter M. Asbeck, and Christian Fager. Millimeter-wave power amplifier integrated circuits for high dynamic range signals. *IEEE Journal of Microwaves*, 1(1):299–316, 2021.

[35] M.K. Kazimierczuk. *RF Power Amplifiers*. Wiley, 2014.

[36] Behzad Razavi. *Design of Analog CMOS Integrated Circuits*. McGraw-Hill Education, 2016.

[37] N. R. Sivaraaj and K. K. Abdul Majeed. A comparative study of ring vco and lc-vco: Design, performance analysis, and future trends. *IEEE Access*, 11:127987–128017, 2023.

[38] Sungyoung Lee, Ziyi Wang, Seunggeun Kim, Taekyun Lee, and David Z Pan. Self-supervised graph contrastive pretraining for device-level integrated circuits. *arXiv preprint arXiv:2502.08949*, 2025.

A Dataset Details and Performance Metric Definitions

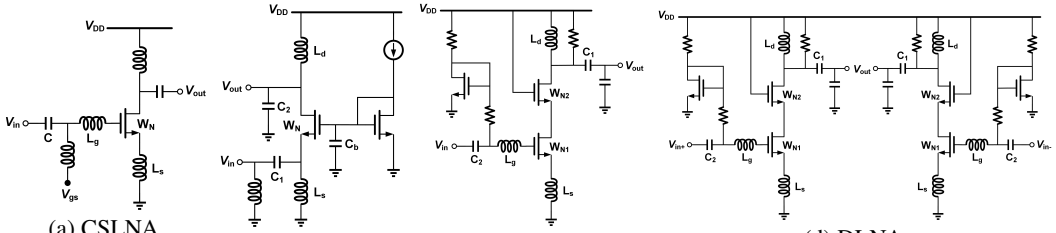
During dataset generation, each simulated circuit instance is annotated with a set of performance metrics that capture its functional characteristics. All simulations are performed at a fixed frequency of 30 GHz, ensuring consistency across circuit types and relevance to mm-wave design. A total of 16 metrics are defined across all circuits—spanning gain, efficiency, impedance matching, noise, and frequency-domain behavior—though the specific metrics used vary by topology. For example, phase noise is only applicable to oscillators. An overview of all performance metrics is provided in Table 4.

Table 4: Overview of 16 performance metrics used during dataset generation.

Performance Name	Description
DC Power Consumption (DCP)	Total power drawn from the DC supply indicating energy consumption of the circuit
Voltage Gain (VGain)	Ratio of output voltage amplitude to input voltage amplitude
Power Gain (PGain)	Ratio of output power to input power
Conversion Gain (CGain)	Ratio of output power at the desired frequency to input power at the original frequency
$\mathbf{S}_{11}$	Input reflection coefficient indicating impedance matching at the input terminal
$\mathbf{S}_{22}$	Output reflection coefficient indicating impedance matching at the output terminal
Noise Figure (NF)	Ratio of input signal-to-noise ratio to output signal-to-noise ratio
Bandwidth (BW)	Frequency span over which the circuit maintains specified performance characteristics
Oscillation Frequency (OscF)	Steady-state frequency at which the oscillator generates a periodic signal
Tuning Range (TR)	Range of achievable oscillation frequencies through variation of control voltages
Output Power (OutP)	Power delivered to the load
$\mathbf{P}_{\mathbf{SAT}}$	Maximum output power level beyond which gain compression begins to occur
Drain Efficiency (DE)	Ratio of RF output power to DC power consumption.
Power-Added Efficiency (PAE)	Ratio of the difference between output power and input power to DC power consumption
Phase Noise (PN)	Measure of oscillator stability represented in the frequency domain at a specified offset
Voltage Swing (VSwg)	Maximum peak voltage level achievable at the output node

A.1 Low-Noise Amplifiers (LNAs)

Low-noise amplifiers (LNAs) are critical components in receiver front-ends, responsible for amplifying weak antenna signals while introducing minimal additional noise. Their performance directly influences downstream blocks such as mixers and analog-to-digital converters (ADCs), ultimately determining system-level fidelity [32]. To capture the architectural diversity of practical radio-frequency (RF) designs, we include four widely used LNA topologies in this study—common-source LNA (CSLNA), common-gate LNA (CGLNA), cascode LNA (CLNA), and differential LNA (DLNA)—as shown in Figure 6.

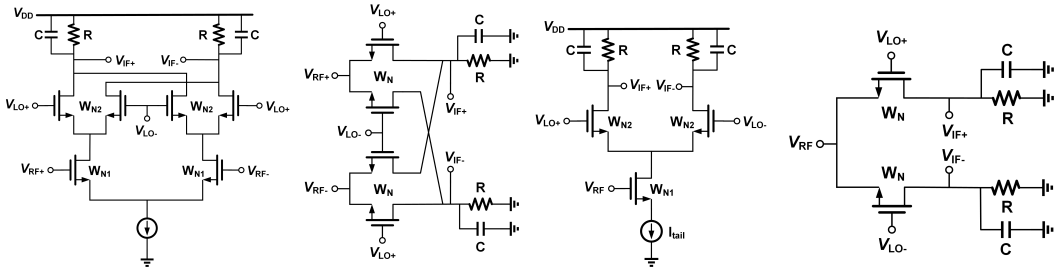


(a) CSLNA (b) CGLNA (c) CLNA (d) DLNA

Figure 6: Schematic diagrams of the four LNA topologies.

A.2 Mixers

Mixers are fundamental nonlinear components in RF systems, responsible for frequency translation by combining two input signals to produce outputs at the sum and difference of their frequencies. This functionality is essential for transferring signals across frequency domains and is widely used in both transmission and reception paths [33]. To capture diverse mixer architectures, we implement four representative topologies in this work—double-balanced active mixer (DBAMixer), double-balanced passive mixer (DBPMixer), single-balanced active mixer (SBAMixer), and single-balanced passive mixer (SBPMixer)—as shown in Figure 7.

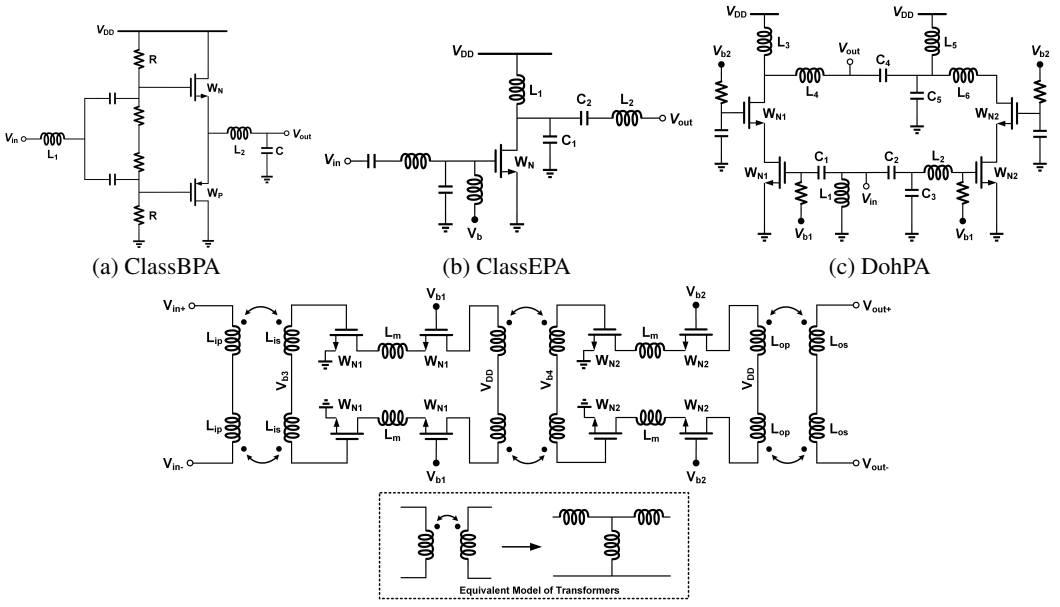


(a) DBAMixer (b) DBPMixer (c) SBAMixer (d) SBPMixer

Figure 7: Schematic diagrams of the four Mixer topologies.

A.3 Power Amplifiers (PAs)

Power amplifiers (PAs) are the most power-intensive components in radio-frequency (RF) systems and serve as the final interface between transceiver electronics and the antenna. Given their widespread use and the stringent demands of modern communication standards, PA design requires careful trade-offs across key performance metrics [34]. Based on the transistor operating mode, PAs are typically grouped into several canonical classes [35]. In this work, we implement four representative topologies—Class-B PA (ClassBPA), Class-E PA (ClassEPA), Doherty PA (DohPA), and differential PA (DPA)—as shown in Figure 8.



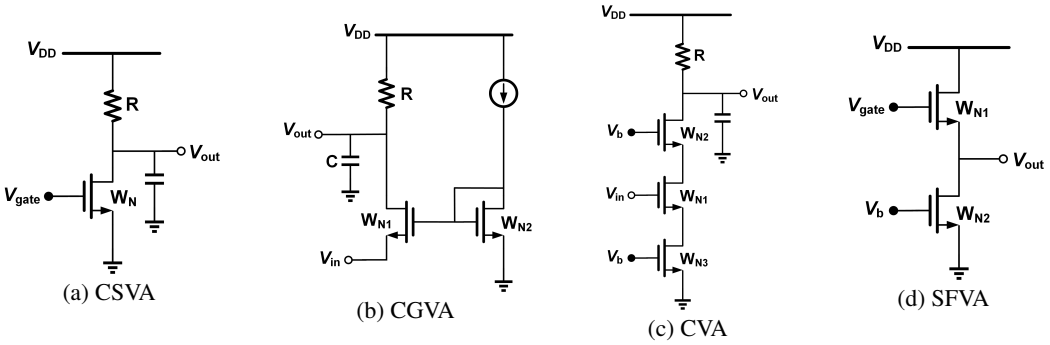
(a) ClassBPA (b) ClassEPA (c) DohPA

(d) DPA

Figure 8: Schematic diagrams of the four PA topologies.

A.4 Voltage Amplifiers (VAs)

Voltage amplifiers (VAs) are fundamental components in analog circuit design, responsible for increasing signal amplitude while preserving waveform integrity. Effective VA design requires balancing key performance metrics tailored to both RF and baseband operating conditions [36]. In this work, we implement four widely used VA topologies—common-source VA (CSVA), common-gate VA (CGVA), cascode VA (CVA), and source follower VA (SFVA)—as shown in Figure 9.

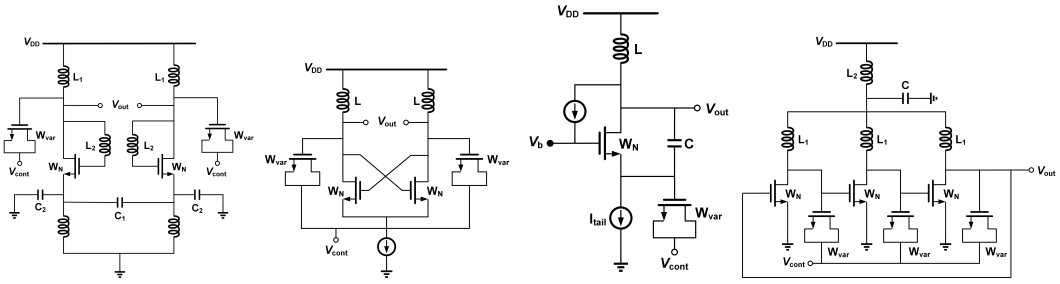


(a) CSVA (b) CGVA (c) CVA (d) SFVA

Figure 9: Schematic diagrams of the four VA topologies.

A.5 Voltage-Controlled Oscillators (VCOs)

Voltage-controlled oscillators (VCOs) are essential building blocks in analog and RF systems, responsible for generating periodic waveforms with frequencies modulated by a control voltage. These circuits rely on amplification, feedback, and resonance to sustain stable oscillations. Owing to their wide tuning range, low power consumption, and ease of integration, VCOs are broadly used in systems such as phase-locked loops (PLLs), frequency synthesizers, and clock recovery circuits [37]. In this work, we implement four representative VCO topologies—inductive-feedback VCO (IFVCO), cross-coupled VCO (CCVCO), Colpitts VCO (ColVCO), and ring VCO (RVCO)—as shown in Figure 10.



(a) IFVCO (b) CCVCO (c) ColVCO (d) RVCO

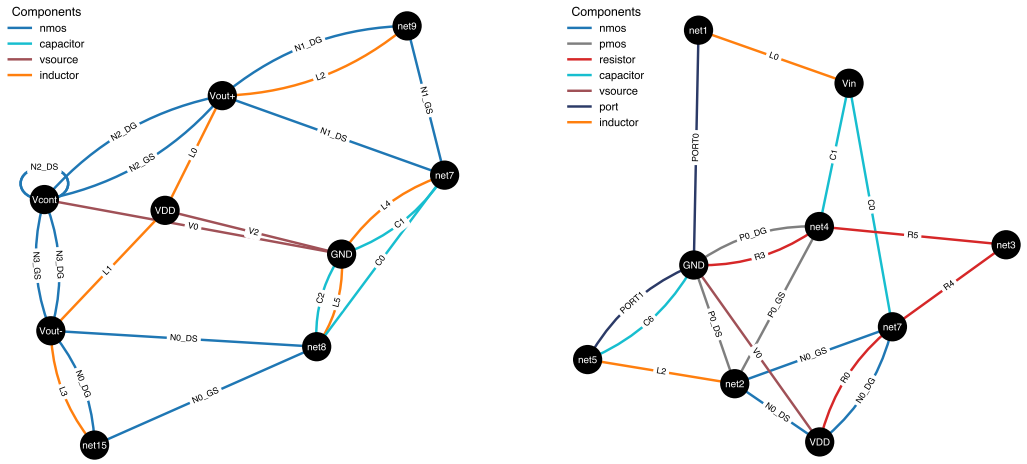
Figure 10: Schematic diagrams of the four VCO topologies.

B Graph-Based Circuit Representation

To enable flexible and topology-agnostic learning, we represent each analog circuit as a graph extracted from its corresponding Cadence netlist. Nodes correspond to voltage nets (i.e., electrical connection points), and edges represent circuit elements such as transistors, resistors, capacitors, or sources. Multi-terminal devices—such as transistors and baluns—are decomposed into multiple edges, and multiple components may connect the same node pair, resulting in **heterogeneous, multi-edged graphs** that preserve structural and functional diversity.

Recent works such as DICE [38] have explored transistor-level circuit-to-graph conversions for self-supervised learning, highlighting the challenges of faithfully capturing device structure and connectivity. In contrast, our approach maintains a native representation aligned with foundry-compatible netlists. Rather than flattening or reinterpreting device abstractions, we preserve symbolic parameters, multi-edge connections, and device-specific edge decomposition directly from the schematic source, enabling scalable learning across diverse analog circuit families.

To support learning over such structured graphs, each edge is annotated with a rich set of attributes: (i) a **categorical device type**, specifying the component and connected terminal pair (e.g., NMOS drain–gate, resistor); (ii) **numeric attributes**, such as channel length or port resistance, fixed by the schematic; (iii) **parametric attributes**, defined symbolically in the netlist (e.g., *W1*, *R3*) and



(a) IFVCO (b) ClassBPA

Figure 11: Graph representations of two analog circuit topologies from our dataset: (a) IFVCO and (b) ClassBPA. Nodes represent electrical nets, and colored edges denote circuit components such as transistors, capacitors, inductors, and sources. Each component type is visually distinguished by color and labeled with its name and terminal role (e.g., N2_GS, V0). For transistors, labels such as GS, DS, and DG denote source-to-gate, drain-to-source, and drain-to-gate connections, respectively. These graphs serve as input to our GNN-based performance modeling and inverse design pipeline.

resolved numerically during preprocessing; (iv) **one-hot categorical features**, such as source type (DC, AC, or none); and (v) **computational attributes**, such as diffusion areas (*Ad*, *As*) derived from sizing. This rule-based graph construction generalizes across circuit families without task-specific customization. Graphs in the FALCON dataset range from 4–40 nodes and 7–70 edges, reflecting the variability of practical analog designs. Figure 11 shows two representative graph examples from our dataset—IFVCO and ClassBPA.

C User-Defined Loss Functions for Gradient Reasoning

Stage 3 of FALCON employs gradient reasoning with the forward GNN fixed, enabling the optimization objective to be redefined without retraining or fine-tuning the predictive model. This design allows users to flexibly adapt the loss function to capture specific trade-offs or constraints. We illustrate this flexibility with two examples.

Weighted Performance Loss. Rather than treating all performance metrics equally, users can specify weights α_i for each target metric:

$$\mathcal{L}_{\text{perf-weighted}} = \frac{1}{\sum_i m_i \alpha_i} \sum_{i=1}^{d} m_i \alpha_i \, (\hat{y}_i - y_i^{\text{target}})^2,$$

where larger α_i prioritize certain specifications (e.g., gain or noise figure). Here, $m_i = 1$ if the i-th metric is defined for the current sample, and 0 otherwise.

Interval-Constrained Performance Loss. Users may also define acceptable ranges for metrics rather than fixed targets. Given optional lower and/or upper bounds $y_i^{\text{lower}}, y_i^{\text{upper}}$, the interval penalty is:

$$\mathcal{L}_{\text{perf-interval}} = \frac{1}{\sum_i m_i} \sum_{i=1}^{d} m_i \Big[\mathbb{1}_{\{y_i^{\text{upper}} \text{ defined}\}} \max(0, \hat{y}_i - y_i^{\text{upper}}) + \mathbb{1}_{\{y_i^{\text{lower}} \text{ defined}\}} \max(0, y_i^{\text{lower}} - \hat{y}_i) \Big],$$

where the indicator $\mathbb{1}_{\{\cdot\}}$ indicates whether the corresponding bound is specified. This formulation naturally handles the cases where only an upper bound, only a lower bound, or both bounds are provided. As above, $m_i = 1$ if the i-th metric is defined for the current sample, and 0 otherwise.

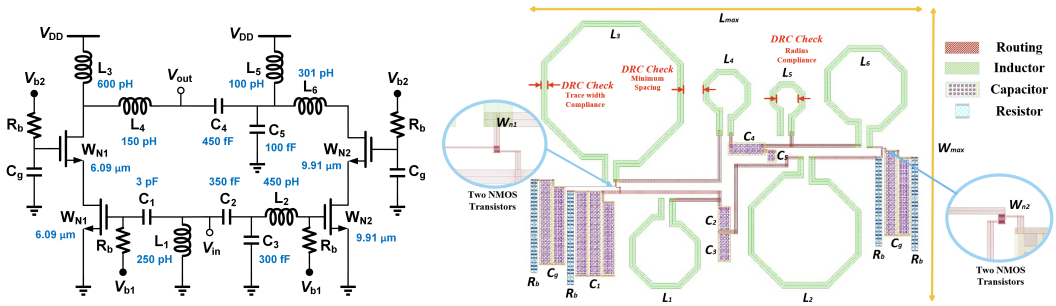
General Extensibility. More generally, the total loss $\mathcal{L}_{\text{total}}$ can be replaced with any user-defined formulation, allowing both $\mathcal{L}_{\text{perf}}$ and $\mathcal{L}_{\text{layout}}$ to be substituted with customized objectives. Additional physical constraints, multi-objective trade-offs, or alternative layout penalties can be incorporated with only a few lines of code. This extensibility underscores the flexibility of FALCON and enables the framework to adapt to diverse design objectives.

D Layout Examples of Synthesized Circuits

To illustrate the correspondence between schematic and layout representations, we present two synthesized circuits: DohPA and IFVCO, shown in Figures 12 and 13, respectively.

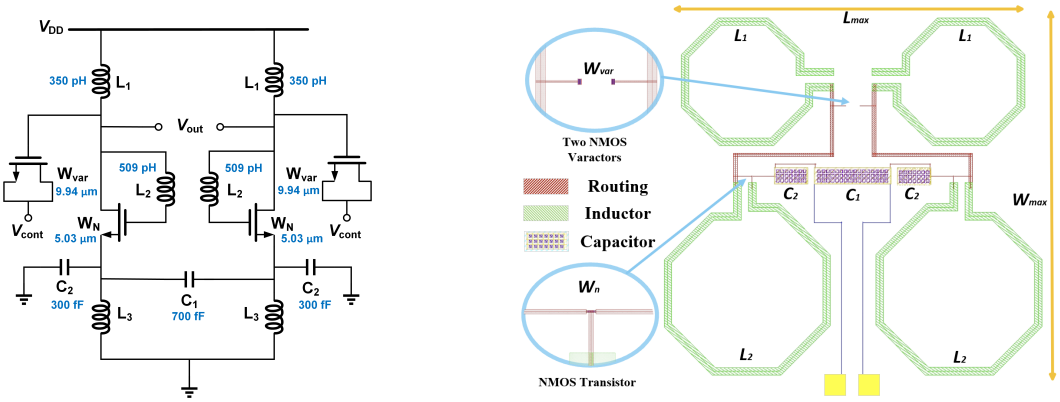
In the IFVCO example, the inductor labeled L_3 functions as an RF choke and is excluded from the on-chip layout due to its large area requirement. Instead, it is intended for off-chip implementation at the PCB level and connected to the die via wire bonding. This external connection is indicated by the yellow pad in Figure 13(b), which serves as the wire-bonding interface.

Since the current stage of system lacks automated routing, all interconnects in the layout were manually drawn to ensure accurate correspondence with the schematic connectivity. These examples demonstrate that synthesized circuit parameters can be successfully translated into DRC-compliant, physically realizable layouts, bridging the gap between high-level optimization and tapeout-ready design.



(a) Designed DohPA schematic (b) Layout of designed DohPA

Figure 12: Stage 3 results for a synthesized DohPA. The schematic (a) reflects optimized parameters to meet the target specification. The layout (b) is DRC-compliant and physically realizable. The final design achieves a mean relative error of 5.4% compared to the target performance.



(a) Designed IFVCO schematic (b) Layout of designed IFVCO

Figure 13: Stage 3 results for a synthesized IFVCO. The schematic (a) reflects optimized parameters to meet the target specification. The layout (b) is DRC-compliant and physically realizable. The final design achieves a mean relative error of 1.3% compared to the target performance.

E Practical Considerations and Limitations

E.1 Training and Inference Efficiency

Although our codebase supports GPU acceleration, all experiments in this work—excluding initial dataset generation—were conducted entirely on a MacBook CPU. This highlights the efficiency and accessibility of the FALCON pipeline, which can be executed on modest hardware without specialized infrastructure. Our MLP and GNN models contain 207k and 1.4M trainable parameters, respectively, with memory footprints of just 831 KB and 5.6 MB.

In Stage 1, the MLP classifier trains in approximately 30 minutes with a batch size of 256 and performs inference in the order of milliseconds per batch. Stage 2's GNN model takes around 3 days to train on the full dataset using the same batch size and hardware. Fine-tuning on an unseen topology (e.g., RVCO) using $\sim$30,000 samples completes in under 30 minutes.

In Stage 3, the pretrained GNN is used without retraining to perform layout-constrained parameter inference via gradient-based optimization. Inference is conducted one instance at a time (batch size 1), with typical runtimes under 1 second per circuit. Runtime varies based on the convergence threshold and circuit complexity but remains below 2–3 seconds in the worst case across the full test set.

A solution is considered successful if the predicted performance meets the target within a specified relative error threshold. While tighter thresholds (e.g., 5%) improve accuracy, they require more optimization steps—particularly over large datasets. As a result, both success rate and inference time in Stage 3 are directly influenced by this tolerance, which can be tuned based on design fidelity requirements.

E.2 Limitations

This work focuses on a representative set of 20 curated analog topologies spanning five circuit families. Consequently, the topology selection stage is limited to suggesting only among the designs present in the training set and cannot synthesize novel circuits. A natural future direction is to either extend the training library to a broader set of topologies or replace the classifier with a generative model capable of directly proposing new netlists conditioned on input specifications. In contrast, the GNN-based forward modeling stage is designed to operate on arbitrary circuit graphs and has already demonstrated strong generalization to unseen architectures (e.g., RVCO), indicating that no modification to this stage is required to support novel circuits.

Beyond topology considerations, the dataset is constructed at a fixed operating frequency of 30 GHz, which ensures consistency across circuit families but constrains frequency generalization. Although the framework can, in principle, extend to other operating points—for example, the voltage amplifier topologies already demonstrate scalability across varying gain–bandwidth trade-offs—systematic validation across diverse frequency bands is beyond the scope of this work. Extending the dataset to cover multiple operating frequencies, or incorporating frequency as an explicit conditioning variable during training, represents an important direction for broadening applicability.

Finally, the differentiable layout model in FALCON captures parasitic effects through analytical approximations of passive components, which is effective for guiding parameter optimization within the learning framework. However, this approach does not fully replace electromagnetic (EM) simulations or post-layout verification, and electromigration constraints are not explicitly incorporated. Incorporating learned parasitic estimators, EM-informed models, and reliability constraints, therefore, remains an important extension toward bridging schematic-level optimization and silicon-proven robustness. In addition, all interconnect routing in the current flow is performed manually to ensure precise control over parasitic management and DRC compliance. While this provides accuracy for the studied designs, it limits scalability for more complex circuits, motivating future integration with automated analog routing tools.