THUNDERGEN: A COMPREHENSIVE DATASET AND MULTI-TOPOLOGY SYNTHETIC ESTIMATOR FOR FAST ANALOG CIRCUIT SIMULATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Considering the design of analog integrated circuits (ICs), SPICE-level simulations remain essential for final verification, while their prohibitive computational requirements motivated the research towards synthetic performance estimators to accelerate early-stage design space exploration. However, these surrogates frequently suffer from a significant accuracy gap, failing to capture the non-linearities and high-dimensional complexities of analog circuits. This is mainly due to the lack of high-fidelity datasets covering vast architectural design spaces. Existing works primarily focus on fixed topologies and parameter sweeps, hindering the development of robust machine learning models capable of true topological generalization. This paper presents ThunderGen, a publicly available comprehensive framework and corresponding dataset conceived to bridge the gap between simulation speed and estimation accuracy. ThunderGen introduces a large-scale, multi-topology dataset generated through an automated pipeline that spans diverse circuit families, providing a rigorous ground truth. Alongside this dataset, we propose a performance estimator that serves as a high-fidelity benchmark to replace time consuming SPICE-level simulations and enabling AI-driven topology searches in early design phases. The experimental campaign evaluated ThunderGen against two state-of-the-art methodologies considering a realistic dataset consisting of more than $44,000$ circuits organized in five circuit families and thousands of circuit topologies per each. Experimental results demonstrate that ThunderGen outperforms existing estimators by $4.69\times$ and $6.07\times$ respectively, also showing an average relative percentage error compared to SPICE simulation limited to 16.11%.

1 INTRODUCTION

The design of analog integrated circuits (ICs) remains one of the most intellectually demanding phases of the semiconductor development cycle. Unlike digital design, which benefits from high levels of abstraction and automated tool-flows, the standard analog design flow is characterized by a meticulous, iterative process that relies heavily on expert intuition and exhaustive verification. At the heart of this workflow SPICE-level simulations represent the gold-standard to ensure silicon success, while their extreme computational cost has become a primary bottleneck in the era of rapid hardware innovation and AI-driven automation.

SPICE-level simulations are critical for circuit optimization and debugging across three pivotal stages. First, during pre-layout sizing, designers must navigate vast, non-linear parameter spaces to optimize for power, speed, and area Kochar et al. (2025); Ahmadzadeh et al. (2025). Second, the recent rise of AI-generated circuits matching provided performance specification has introduced a need for high-throughput validation tools to filter out non-functional topologies produced by generative models Chang et al. (2024); Gao et al. (2025). Third, at the post-layout stage, exhaustive simulations are the only way to validate pre-tapeout performance metrics against parasitic effects Xu et al. (2019). All in all, the shrink of the process nodes makes non-ideal silicon effects dominant, thus motivating the exponential growth in the number of simulations required to achieve circuit closure Liu et al. (2020).

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

To this end, seminal research works have started replacing time-consuming simulations with synthetic circuit evaluations. The power of these estimators lies in early-stage front-end design, not in final sign-off where zero-error SPICE validation remains mandatory. The final goal of delivering fast mathematical data-driven proxies that mimic the simulator to boost circuit-level optimization algorithms, e.g., sizing, and circuit generation Liu et al. (2025); Mehradfar et al. (2025).

However, existing state-of-the-art performance estimators frequently suffer from a significant accuracy gap compared to SPICE-level simulations. For instance, available estimators might predict a high gain for a specific amplifier sizing, while SPICE-level simulations reveal that the circuit is actually unstable or non-functional due to complex transistor-level interactions that simplified models learned by estimators fail to capture. Such failures are exacerbated by the lack of standardized, high-fidelity datasets that would allow for the development of more robust and generalized machine learning models. For example, the publicly available datasets are limited to a few circuit topologies per circuit family, and the size of the dataset is artificially increased by sweeping circuit parameters once the circuit topology is fixed. However such practice prevents the identification of high-fidelity, generic and robust performance estimators since the identification problem is demoted to a simple classification task selecting between few circuit topologies.

To address these limitations, this paper presents ThunderGen, a comprehensive framework and dataset designed to bridge the gap between simulation speed and performance estimation accuracy. ThunderGen delivers a new foundation for the community by standardizing how synthetic estimators are trained and evaluated with three major contributions to the state of the art:

- *Performance Estimator* - ThunderGen delivers a novel performance estimation flow designed to replace time-consuming SPICE-level simulations required in the analog design flow. By capturing higher-order nonlinearities more effectively than traditional lumped models, our estimator function serves as a high-fidelity proxy for circuit performance.
- *Extensive Experimental Campaign and Dataset* - ThunderGen introduces a new, automated dataset creation pipeline. Using this pipeline, we have generated a massive data set exceeding $44,000$ circuits, spanning five distinct circuit families, and thousands of topologies per circuit family. The diversity in the dataset ensures strong experimental evaluations also setting the stage for further investigations.
- *Effectivness Usecase* - ThunderGen demonstrates the validity of the proposed pipeline by employing the estimator to evaluate the performance of a large dataset of AI-generated analog circuit schematics from given performance specifications. In addition, ThunderGEN has been compared against multiple state-of-the-art methodologies, demonstrating significant improvements in generalizability across diverse topologies. Compared to SPICE-level simulations, ThunderGen shows an average Percentage Relative Error (PRE) limited to 16.11% with a net improvement against state-of-the-art performance estimators up to $6.07\times$.

By releasing the dataset, the generation pipeline, and the pre-trained estimators, ThunderGen facilitates reproducible research and pushes the boundaries of AI-driven electronic design automation (EDA) [1].

The rest of the paper is organized in four sections. Section 2 presents the state of the art. The methodology and the experimental results are discussed in Section 3 and Section 4, respectively. Conclusions are drawn in Section 5.

2 Related Work

Machine learning (ML) research in analog design primarily focuses on two tightly interconnected tasks: topology selection, the synthesis of a circuit schematic from functional specifications, and parameter selection, i.e., circuit sizing, the optimization of device sizing for a given schematic Mehradfar et al. (2025); Vijayaraghavan et al. (2025); Gao et al. (2024). Considering the performance estimation problem, Table 1 summarizes relevant literature based on computational approaches, optimization targets, dataset availability, and the use and validation of performance estimators. Seminal works in this domain largely focused on the sizing problem, using classical Reinforcement Learning and

[1] Repository: AnonymousUser00/thundergen

Table 1: Comparison of *ThunderGen* with similar ML CAD works targeting performance estimation and validation to support automatic topology and parameter selection problems.

	Method	Topology selection	Parameter selection	Public Dataset	Performance Prediction	Performance Validation
MACE Zhang et al. (2021)	ML	✗	✓	✗	✗	✗
GCN-RL Wang et al. (2020)	ML	✗	✓	✗	✗	✗
AnalogGym Li et al. (2024)	ML	✗	✓	✓	✗	✗
LaMAGIC Chang et al. (2024)	LLM	✓	✗	✗	✗	✗
AnalogCoder Lai et al. (2025)	LLM	✓	✗	✓	✗	✗
AnalogGenie Gao et al. (2025)	LLM	✓	✓	✓	✗	✗
AmpAgent Liu et al. (2024)	LLM	✓	✓	✗	✗	✗
CktGNN Dong et al. (2023)	GNN	✓	✓	✓	✗	✗
DiffCkt Liu et al. (2025)	GNN+ Diffusion	✓	✓	✓	✓	✗
Falcon Mehradfar et al. (2025)	GNN	✓	✓	✓	✓	✗
ThunderGen	GNN+Diffusion	✓	✓	✓	✓	✓

Bayesian Optimization, as seen in frameworks like MACE Zhang et al. (2021) and GCN-RL Wang et al. (2020), respectively. To facilitate standardized benchmarking, AnalogGym Li et al. (2024) introduced a framework encompassing diverse process design kits (PDKs) and topologies, establishing a foundation for validating parameter selection methods across different ML approaches. More recently, research has pivoted toward the generative capabilities of LLMs. LaMAGIC Chang et al. (2024) fine-tunes masked language models for small-scale circuit generation. To bypass the scarcity of specialized training data, AnalogCoder Lai et al. (2025) employs domain-specific prompt engineering with large-scale models like GPT-4 to generate PySpice code, effectively translating specifications into executable netlists, albeit without explicitly addressing parameter optimization. In contrast, AnalogGenie Gao et al. (2025) adopts a combination of LLM and sequence-based graph representation to improve scalability and circuit variety, contributing a dataset of $3,000$ designs, while AmpAgent Liu et al. (2024) leverages a multi-agent hierarchy to incorporate transfer function analysis into the design flow. Despite their generative promise, LLMs face two critical challenges: their sequential nature conflicts with the inherent graph structure of analog circuits, and they rely on slow SPICE simulations for validation. Addressing this structural mismatch, CktGNN Dong et al. (2023) uses Graph Neural Networks (GNNs) to automate topology and parameter selection, introducing an OpAmp dataset of $10,000$ circuits. DiffCkt Liu et al. (2025) and Falcon Mehradfar et al. (2025) further advance specification-driven generation. Falcon provides over a million samples; however, its dataset artificially augments a few topologies via parameter sweeps, limiting variety and hindering the design of robust performance estimators. Furthermore, while Liu et al. (2025); Mehradfar et al. (2025) pioneered ML-based estimators to accelerate the simulation bottleneck, they lack rigorous fidelity validation against SPICE ground truths. This missing validation remains a critical barrier preventing the trustworthy adoption of high-speed performance estimators in automated design.

3 METHODOLOGY

ThunderGen introduces (i) an automated generation pipeline to create a large-scale, multi-topology dataset spanning diverse analog circuit families and (ii) an identification framework for high-fidelity, robust, and generic performance estimators to replace SPICE-level simulations. Figure 1 depicts the infrastructure consisting of the two interactive components: the *Circuit Generative Pipeline* and the *Performance Estimation flow*. Starting from a set of publicly available hand-made analog circuits, the generative pipeline creates a novel dataset of analog circuits by generating new topologies and for each new topology the parameter selection is used to match circuit specifications. Moreover, each generated valid circuit is augmented with its SPICE-level simulation results. All in all, the generative pipeline produces a tuple for each created circuit consisting of (i) an annotated graph of the circuit, (ii) the netlist, and (iii) the testbench for SPICE-level simulation, and (iv) the performance results extracted from the SPICE-level simulations.

Notably, SPICE-level simulations are executed for each circuit within the generative pipeline to create the final public dataset for which the structure is shown in Figure 2. The generated dataset

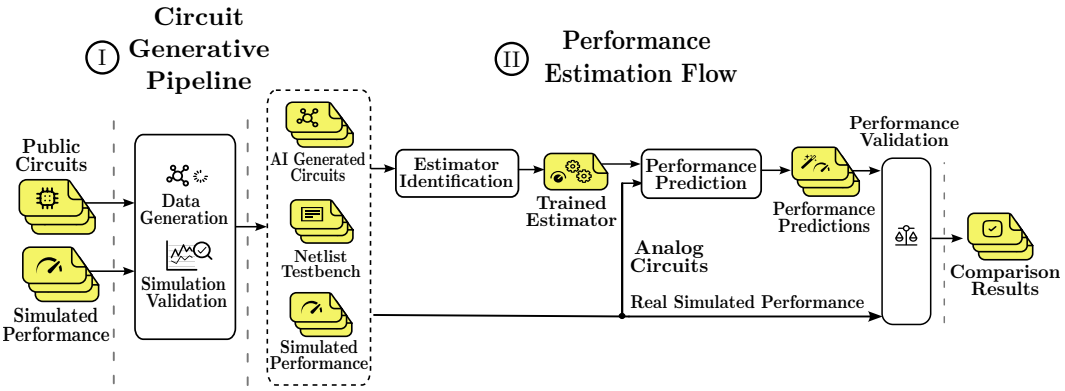


Figure 1: Circuit generative pipeline and performance estimator identification and validation.

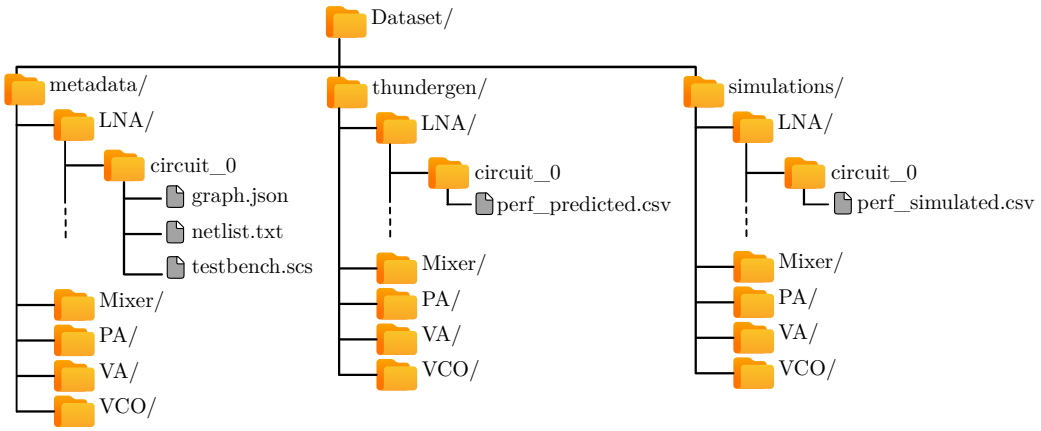


Figure 2: Structural view of the publicly available circuit dataset.

is partitioned between train, validation, and test sets to identify the performance estimators and for their validation. ThunderGen creates a circuit performance estimator for each circuit family under the domain-specific, experimental, and realistic assumption that the number of classes of analog circuits are limited Gao et al. (2025); Mehradfar et al. (2025). The Performance Estimation Flow creates the circuit performance estimators starting from the training and test parts of the dataset (see *Estimator Identification* in Figure 1), while the identified estimators are used to predict the performance for the circuits of the validation set (see *Performance Prediction* in Figure 1). Last, the *Performance Validation* block returns the quality of the estimated performance as the distance between the estimated quality metrics and the one collected from the SPICE simulation (more details are discussed in Section 4 and Appendix A). The rest of this section is organized in two parts. Section 3.1 discusses the generative pipeline, while Section 3.2 details the methodology to create and validate the performance predictors.

3.1 Circuit Generative Pipeline

Figure 3 shows the Circuit Generative Pipeline as organized in two stages: (i) Data Generation, which trains the diffusion stack and synthesizes new circuits using a Graph Internal Representation (GIR), and (ii) Simulation and Validation, which translates the GIR-encoded circuits into physical netlists, wraps them with testbenches, and filters them after their simulation to provide a final validated dataset.

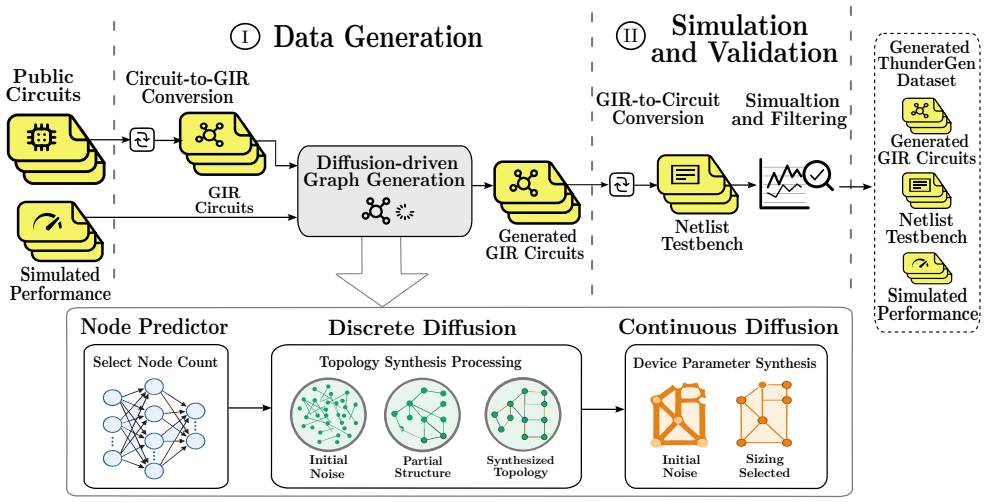


Figure 3: Circuit generative Pipeline. Circuits from public datasets are converted into a unified graph intermediate representation to train the diffusive generative pipeline. Generated circuits are filtered by the Simulation and Validation stage that leverages on SPICE-level simulations to retain only physically viable circuits for the final Validated Dataset.

Data Generation - Starting from the public datasets Gao et al. (2025); Liu et al. (2025); Mehradfar et al. (2025), the Data Generation stage creates a new dataset of graph-encoded circuits with the objective of creating high variability in the number of topologies and parameter selection. At first, the circuits in the public datasets are converted into ThunderGen GIR to be fed into the Diffusion-Driven Circuit Generation (DDCG) stage. GIR offers an informative, abstract, and efficient, i.e., matrix-based, representation of the circuit, encoding each device as a node and circuit connections as links in the graph. For each circuit in the publicly available datasets, the Diffusion-Driven Circuit Generation (DDCG) stage takes in input the GIR and the performance measured from SPICE-level simulations and outputs a new set of GIR-encoded circuits varying the topology circuit and the parameters of each device (see DDCG block in Figure 3). Notably, the two inputs are used to train the models according to the standard diffusion-based training strategy Liu et al. (2025); Ho et al. (2020); Song et al. (2022); Vignac et al. (2022). Then, the trained DDCG receives the SPICE-level performance, used as circuit specification, in input to generate a new circuit. The DDCG pipeline consists of three blocks. First, the Multi-Layer Perceptron (MLP) network receives the SPICE-level performance vector and predicts the device-count of the circuit to be generated. Second, the discrete diffusion model creates the circuit topology from the predicted number of devices and the circuit performance specification. Last, the continuous diffusion model takes the SPICE-level performance vector and the created circuit topology to solve the parameter selection problem, i.e., to size each device in the newly created circuit. The final output is a new GIR-encoded circuit. Appendix A.4 discusses further details of the generative model architecture as well as training and inference phases.

Simulation and Validation - This stage filters the GIR-encoded circuits from the Data Generation stage to produce the final ThunderGen dataset. Notably, AI-generated GIR-encoded circuits can contain electrically unfeasible structures or the circuit performance can be far off with respect to the specification. To this end, the Simulation and Validation stage implements a 3-stage, automated, and algorithmic-based pipeline. At first, GIR-encoded circuits are converted into SPICE netlists for simulation. Then, SPICE-level simulation are executed to filter out non-working circuits. Last, the GIR-encoded circuit, its netlist and testbench, as well as the simulated performance are appended to the final ThunderGen dataset.

3.2 Performance Estimation Flow

Figure 4 shows the Performance Estimation Flow as organized in two stages: Performance Estimation, which trains the performance estimator for each circuit family and extracts circuit performance estimates, and Estimation Validation, which validates the performance estimates against SPICE-level

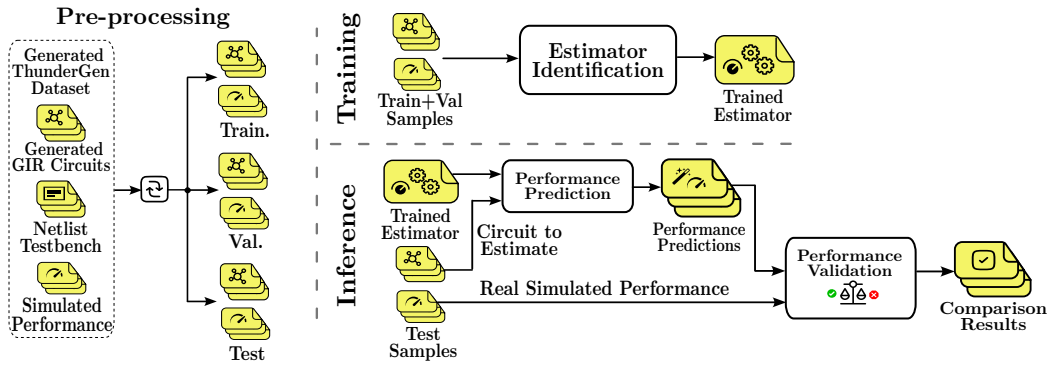


Figure 4: Performance estimation flow. For each circuit family, the generated dataset is split in train, test, and validation sets to train the performance estimator, compute the estimates and validate them against SPICE-level results.

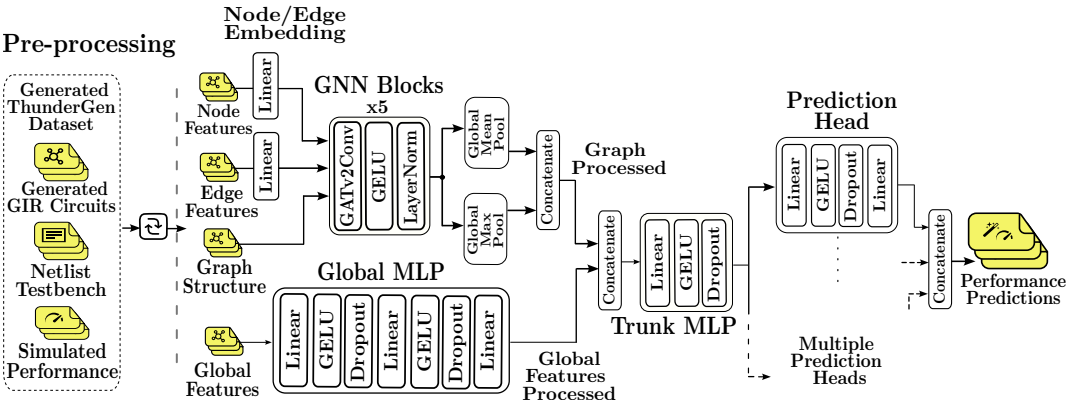


Figure 5: HybridGNN model architecture.

simulations. Notably, the performance estimation flow organizes the generated dataset into train, test, and validation sets.

Performance Estimation - For each circuit family, the stage takes the train and validation sets of the generated dataset to train the performance estimator (see *Estimator Identification* in Figure 4). In particular, the *Estimator Identification* block trains the HybridGNN which architecture is depicted in Figure 5. The dataset is preprocessed before being fed into the HybridGNN architecture and organized as four features as detailed in Table 2. Notably, the HybridGNN architecture is generic, but it allows to train performance predictors considering a variable number of quality metrics. To this end, the number of *Prediction Head* in the trained HybridGNN can be adjusted to be equal to the desired number of predicted performance metrics. Indeed, ThunderGen trains a separate performance estimator for each circuit family. The complete parametrization of the proposed HybridGNN is detailed in Table 10 in Appendix A.5.

Estimation Validation - For each circuit, the stage uses the trained performance estimator and the GIR-encoded representation of the circuit itself to predict its performance (see *Performance Prediction* in Figure 4). Then, the predicted performance are validated against the results extracted from the SPICE simulations and the Percentage Relative Error is calculated (see *Performance Validation* in Figure 4).

Table 2: Pre-processing summary for the inputs of the performance estimator. The HybridGNN model splits the problem into two parts: GNN blocks branch studies the exact, pin-by-pin wiring of the circuit, while Global MLP looks at a high-level features of the entire design.

Input Type	Description
GNN Blocks Branch: Graph Processing	
Node features	Identifies each device type (e.g., a transistor or capacitor) and its physical size.
Edge features	Labels the specific pins used for a connection (e.g., the gate, drain, or source of a transistor). This ensures the model understands exactly how a device is plugged into the circuit, rather than just knowing it is connected.
Graph structure	The fundamental wiring diagram that maps physical components to their connecting electrical wires (nets), forming the pathways the neural network uses to analyze the circuit.
Global MLP Branch: Global Features Processing	
Global features	An overview of the entire circuit to represent the overall design by combining simple device counts, averages and ranges of component sizes, general measurements of wiring complexity, and counters tracking how often specific components share a wire. This gives the model an immediate, high-level understanding of the circuit's physical scale and common analog design patterns.

Table 3: Global dataset pipeline: from the initial dataset used for data generation ($\mathcal{D}_{\text{init}}$) to the raw generated pool ($\mathcal{D}_{\text{gen}}$) and the final Spectre-validated dataset ($\mathcal{D}_{\text{val}}$). Filtering required successful Spectre convergence and physically in-range metrics.

	Initial Dataset	**Data Generation**			**Sim Validation**			
Family	$\lvert\mathcal{D}_{\text{init}}\rvert$	$\lvert\mathcal{D}_{\text{gen}}\rvert$	Training Time	Inference Time	$\lvert\mathcal{D}_{\text{val}}\rvert$	Filtered	Time	Unique Top.
LNA	244 927	20 000	5h29m	13h03m	**8 835**	68%	4h08m	5 953 (67%)
Mixer	179 760	20 000	1h33m	5h17m	**8 895**	44%	47m48s	6 267 (70%)
VA	104 842	19 405	34m02s	1h24m	**6 281**	32%	49m35s	479 (8%)
PA	281 171	20 000	3h01s	11h08m	**14 554**	73%	1h26m	11 088 (76%)
VCO	233 100	20 000	57m21s	4h57m	**5 990**	30%	4h26m	4 225 (71%)
Total	1 043 800	99 405	11h23m	35h49m	**44 555**	51%	11h22m	28 012 (63%)

4 Experimental Results

This section presents the experimental results of ThunderGen in terms of the multi-topology circuit generation pipeline and the performance estimator benchmark. The rest of this section is organized in two parts. Section 4.1 presents the hardware/software experimental setup, defines the quality metrics used to validated the proposed performance estimation flow, and presents the generated dataset also showing the time spent in each stage. Section 4.2 presents the experimental results for the performance estimation flow considering different circuit families and two state-of-the-art performance estimators. Performance predictions are compared with SPICE-level simulations to assess the fidelity of the proposed estimators.

4.1 Experimental Setup and Generated Dataset

Hardware and Software Setup - Training and inference of all models related to the automated circuit generation pipeline and the circuit performance estimator as well as the SPICE-level simulations to create the dataset were conducted on a cloud workstation running Ubuntu 24.04 with 480 GB of RAM and an NVIDIA H100 (96 GB). The detailed software configuration, including all libraries and tools used for the experimental campaign, is reported in Appendix A.3.

Generated Dataset - The automated circuit generation pipeline takes in input the publicly available circuit dataset consisting of one million samples organized across 5 circuit families with 4 topologies

each Gao et al. (2025); Mehradfar et al. (2025). The output of the proposed automated circuit generation pipeline is a new dataset where, for each baseline topology, a new set of circuit topologies coupled with the required parameter selection. Table 3 reports for each stage of the automated circuit generation pipeline the evolution in terms of the number of circuits for each circuit family as well as the time spent in each stage of the circuit generative pipeline. According to the pipeline presented in Section 3.1, the Data Generation phase takes circuits from the publicly available dataset for the diffusive generation stack training and inference. In the conducted experimental evaluation, we limited Data Generation to 20-thousand circuits for each circuit family varying the topology and the parameter selection. At last, Simulation and Validation phase validates the generated circuits using exhaustive SPICE-level simulations, dismissing non-working and non-compliant circuits due to the underlined generative process. The final dataset exceeds 44-thousand circuits featuring thousands of new topologies for each circuit family, thus ensuring dataset diversity and generality. Notably, the high number of topologies is essential to train robust and generic performance estimators without demoting the problem of training the performance estimator to a classification problem within the few available circuit topologies.

Quality Metrics - Starting from the performance specifications of the circuit generated by means of the ThunderGen automated generative pipeline ($Perf_{spec,t,i}$), the SPICE simulation returns the actual performance of the circuit ($Perf_{SPICE,t,i}$) where t and i identify the circuit family and the specific performance metric, respectively. Similarly, we defined the estimated circuit performance as $Perf_{est,t,i}$. In addition to the estimated performance from ThunderGen ($Perf_{ThunderGen,t,i}$), we also considered the estimated performance from two state-of-the-art contributions, i.e., DiffCkt ($Perf_{DiffCkt,t,i}$) Liu et al. (2025) and Falcon ($Perf_{Falcon,t,i}$) Mehradfar et al. (2025). To this end, the mean absolute error (MAE) between the specification performance and SPICE or estimated performance for each circuit family t, quality metric i, across the evaluated circuit in the test set ($circ_t$) is defined by Equation (1), where $X \in \{SPICE, ThunderGen, DiffCkt, Falcon\}$. The description of considered quality metrics (i) is detailed in Appendix A.1.

$$MAE_{spec,X,t,i} = \frac{\sum_{n=0}^{circ_t} (Perf_{spec,t,i} - Perf_{X,t,i})^2}{cirt_t} \tag{1}$$

Notably, the actual performance collected from SPICE simulations are expected to slightly differ from the circuit specification due to the diffusive architecture of the automated generative pipeline. It is worth noticing that even in the traditional analog design flow, a certain tolerance between circuit specification and actual performance is allowed. Considering the scope of ThunderGen of demonstrating the creation of robust and accurate performance estimators, SPICE data are used as the reference circuit performance.

The quality of the performance estimators is evaluated in terms of the Mean Percentage Relative Error (MPRE) obtained as the relative error between the estimated and SPICE MAEs, as defined by Equation (2), where $X \in \{ThunderGen, DiffCkt, Falcon\}$, $t \in \{LNA, Mixer, POA, VA, VCO\}$ and i within the addressed quality metrics (see Appendix A.1). For each performance estimator, the MPRE measures the distance between the estimated performance of the specific quality metric with respect to the value collected from SPICE simulations.

$$MPRE_{X,t,i} = \frac{ABS(Perf_{X,t,i} - Perf_{SPICE,t,i})}{Perf_{SPICE,t,i}} * 100 \tag{2}$$

4.2 Performance Estimation Benchmark

For each circuit family and quality metric, Table 4 reports the Mean Percentage Relative Error (MPRE) for ThunderGen as well as for two state-of-the-art estimators, i.e., DiffCkt, and Falcon (see MAE results in Appendix A.2). Considering average results across all circuit families and quality metrics, ThunderGen shows an MPRE limited to 16.11% while DiffCkt and Falcon have an MPRE of 75.56% and 97.89%, respectively. To this end, ThunderGen improves the performance estimation accuracy over the state of the art by 4.69× and 6.07×. All in all, compared to state-of-the-art performance estimators, ThunderGen offers a higher fidelity in the estimates exhibiting an MPRE below 30% for 19 out of 24 quality metrics.

Table 4: MPRE for ThunderGen, DiffCkt, and Falcon. Smaller numbers mean better accuracy.

Circuit Family	Metrics	ThunderGen	DiffCkt	Falcon
LNA	PowerGain	2.44%	84.27%	57.56%
	NoiseFigure	4.95%	89.63%	43.94%
	S11	20.78%	59.04%	125.54%
	Bandwidth	20.50%	87.30%	64.60%
	DCPowerConsumption	28.37%	79.48%	29.10%
	Average	**15.41%**	**79.94%**	**64.15%**
Mixer	ConversionGain	1.39%	92.65%	73.63%
	NoiseFigure	2.56%	90.25%	70.55%
	VoltageSwing	33.85%	97.76%	95.10%
	DCPowerConsumption	30.03%	99.42%	9.42%
	Average	**16.96%**	**95.02%**	**62.18%**
PA	PowerGain	1.07%	90.54%	66.19%
	PSAT	7.31%	56.44%	96.88%
	DrainEfficiency	43.14%	93.20%	207.03%
	PAE	72.55%	99.61%	83.45%
	S11	35.59%	75.44%	623.42%
	S22	8.44%	63.00%	168.05%
	DCPowerConsumption	5.56%	71.67%	12.32%
	Average	**24.81%**	**78.56%**	**179.62%**
VA	VoltageGain	6.57%	33.13%	65.26%
	Bandwidth	18.11%	58.96%	59.15%
	DCPowerConsumption	7.65%	92.22%	147.17%
	Average	**10.78%**	**61.44%**	**90.53%**
VCO	OscillationFrequency	2.13%	88.25%	78.56%
	OutputPower	5.84%	83.95%	28.76%
	PhaseNoise	10.13%	67.52%	278.69%
	TuningRange	39.05%	51.10%	50.00%
	DCPowerConsumption	5.82%	23.47%	28.89%
	Average	**12.59%**	**62.86%**	**92.98%**
All circuits	**Total Average**	**16.11%**	**75.56%**	**97.89%**

5 CONCLUSIONS

This paper presented ThunderGen, a comprehensive dataset and multi-topology performance estimator to accelerate analog circuit simulations. Shifting focus from simple parameter sweeps to true architectural diversity, ThunderGen addresses the lack of rigorous ground-truth validation across varied topologies. The five chosen families span distinct physical regimes (small/large-signal, mixing, and oscillatory behavior); while broader coverage would strengthen the benchmark, our family-agnostic GIR accommodates additions without structural changes, constrained only by SPICE simulation costs. Experimentally, ThunderGen yields accurate surrogates with a 16.11% average Percentage Relative Error against SPICE, improving upon DiffCkt and Falcon by $4.69\times$ and $6.07\times$, respectively. Crucially, ThunderGen enforces the power of estimators in the early stages of front-end design. The ability to accurately and rapidly estimate topological performance paves the way for a more robust and automated analog design flow.

Limitations - While ThunderGen exhibits an MPRE below 30% for 19 out of 24 quality metrics, we acknowledge that for few quality metrics, e.g., PAE and DrainEfficiency metrics in PA circuit family, ThunderGen delivers inaccurate predictions (see Table 4). Notably, state-of-the-art performance estimators perform far worse. Further investigations highlighted that better results can be achieved by improving the training of the performance estimators with (i) a larger dataset and (ii) additional information sources, e.g., the use of circuit-level transfer functions as in AmpAgent Liu et al. (2024), while such an investigation is left as future work.

6 AI USE STATEMENT

We used gnerative AI tools during this work as general-purpose assistants. They helped revise the text and grammar of the manuscript, supported code development, and helped identify and fix bugs. We have not used generative AI tools to gather, process, or analyze data, to develop the methodology, to conduct experiments, or to generate research claims. All AI-assisted text and suggestions were reviewed, verified, and revised by the authors for technical accuracy, fidelity to the implemented methodology, and consistency with the experimental results. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

7 ETHICS STATEMENT

This work does not involve human subjects or personal data. ThunderGen targets analog integrated-circuit design automation. The methodology is intended to reduce manual design effort and does not make decisions affecting individuals or protected groups. We report the experimental setup and evaluation criteria to support reproducibility and responsible interpretation of the results. The main contributions of the paper clearly stated both in the Abstract and the Introduction (Section 1) and study conducted in Related Work (Section 2) demonstrate how our work can foster the research toward new frontiers of analog circuit design and optimization.

8 REPRODUCIBILITY STATEMENT

We support reproducibility through the methodological description in Section 3 and experimental campaign in Section 4. Detailed sections in Appendix A) describe the performance metrics considered, performance results, software configuration, generative pipeline architecture (with training and inference parameters) and performance estimators architecture (with training and inference parameters). All code, data and trained models are released at released Hugging Face (Section 1). The ThunderGen dataset contains 44,555 validated circuits across five families (LNA, Mixer, PA, VA, VCO). Each circuit comes with its component-level graph, a gpdk045 Spectre netlist and its family-specific testbench, and each test circuit also has its Spectre-simulated metrics. For every family, we provide the performance estimator checkpoint together with its exact train, validation and test splits, the training script, and an inference script that reproduces the reported test-set predictions. For the generative pipeline, we release the trained node-count predictor, discrete-diffusion and continuous-diffusion checkpoints, the precomputed per-family data, and a single script that runs the full three-stage generation and evaluation on the test set.

REFERENCES

Mohsen Ahmadzadeh, Kaichang Chen, and Georges Gielen. (invited paper) anaflow: Agentic llm-based workflow for reasoning-driven explainable and sample-efficient analog circuit sizing. In *2025 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, pp. 1–7. IEEE, October 2025. doi: 10.1109/iccad66269.2025.11240818. URL http://dx.doi.org/10.1109/ICCAD66269.2025.11240818.

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.

Zehao Dong, Weidong Cao, Muhan Zhang, Dacheng Tao, Yixin Chen, and Xuan Zhang. Cktgnn: Circuit graph neural network for electronic design automation. *arXiv preprint arXiv:2308.16406*, 2023.

Jian Gao, Weidong Cao, and Xuan Zhang. Analoggenie-lite: Enhancing scalability and precision in circuit topology discovery through lightweight graph modeling. In *Forty-second International Conference on Machine Learning*, 2025.

Xiaohan Gao, Haoyi Zhang, Siyuan Ye, Mingjie Liu, David Z Pan, Linxiao Shen, Runsheng Wang, Yibo Lin, and Ru Huang. Post-layout simulation driven analog circuit sizing. *Science China Information Sciences*, 67(4):142401, 2024.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models, 2020. URL https://arxiv.org/abs/2006.11239.

Dimple Vijay Kochar, Hanrui Wang, Anantha Chandrakasan, and Xin Zhang. Ledro: Llm-enhanced design space reduction and optimization for analog circuits, 2025. URL https://arxiv.org/abs/2411.12930.

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, pp. 379–387, 2025.

Jintao Li, Haochang Zhi, Ruiyu Lyu, Wangzhen Li, Zhaori Bi, Keren Zhu, Yanhan Zeng, Weiwei Shan, Changhao Yan, Fan Yang, et al. Analoggym: An open and practical testing suite for analog circuit synthesis. In *Proceedings of the 43rd IEEE/ACM International Conference on Computer-Aided Design*, pp. 1–9, 2024.

Chengjie Liu, Weiyu Chen, Anlan Peng, Yuan Du, Li Du, and Jun Yang. Ampagent: An llm-based multi-agent system for multi-stage amplifier schematic design from literature for process and performance porting. *arXiv preprint arXiv:2409.14739*, 2024.

Chengjie Liu, Jiajia Li, Yabing Feng, Wenhao Huang, Weiyu Chen, Yuan Du, Jun Yang, and Li Du. Diffckt: A diffusion model-based hybrid neural network framework for automatic transistor-level generation of analog circuits, 2025. URL https://arxiv.org/abs/2507.00444.

Mingjie Liu, Keren Zhu, Xiyuan Tang, Biying Xu, Wei Shi, Nan Sun, and David Z. Pan. Closing the design loop: Bayesian optimization assisted hierarchical analog layout synthesis. In *2020 57th ACM/IEEE Design Automation Conference (DAC)*, pp. 1–6, 2020. doi: 10.1109/DAC18072.2020.9218621.

Asal Mehradfar, Xuzhe Zhao, Yilun Huang, Emir Ceyani, Yankai Yang, Shihao Han, Hamidreza Aghasi, and Salman Avestimehr. Falcon: An ml framework for fully automated layout-constrained analog circuit design, 2025. URL https://arxiv.org/abs/2505.21923.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models, 2022. URL https://arxiv.org/abs/2010.02502.

Clement Vignac, Igor Krawczuk, Antoine Siraudin, Bohan Wang, Volkan Cevher, and Pascal Frossard. Digress: Discrete denoising diffusion for graph generation. *arXiv preprint arXiv:2209.14734*, 2022.

Prashanth Vijayaraghavan, Luyao Shi, Ehsan Degan, Vandana Mukherjee, and Xin Zhang. Autocircuit-rl: Reinforcement learning-driven llm for automated circuit topology generation, 2025. URL https://arxiv.org/abs/2506.03122.

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)*, pp. 1–6. IEEE, 2020.

Biying Xu, Keren Zhu, Mingjie Liu, Yibo Lin, Shaolan Li, Xiyuan Tang, Nan Sun, and David Z Pan. Magical: Toward fully automated analog ic layout leveraging human and machine intelligence. In *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 2019.

Shuhan Zhang, Fan Yang, Changhao Yan, Dian Zhou, and Xuan Zeng. An efficient batch-constrained bayesian optimization approach for analog circuit synthesis via multiobjective acquisition ensemble. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 41(1):1–14, 2021.

A Appendix

A.1 Performance Metrics

To comprehensively evaluate the generated circuit topologies across all considered families, we utilize a diverse set of standard analog and radio-frequency (RF) metrics. Table 5 provides a complete list of the 16 performance metrics used during our experiments, along with a brief description of the physical characteristics they measure.

Table 5: Overview of the 16 performance metrics

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
$\mathbf{P_{SAT}}$	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

Table 6 summarizes the benchmark circuits evaluated in our study, categorized into five distinct analog and radio-frequency (RF) families. For each family, we consider four common circuit topologies to ensure topological diversity. The Low-Noise Amplifier (LNA) family includes Common-Gate (CGLNA), Cascode (CLNA), Common-Source (CSLNA), and Differential (DLNA) architectures. The Mixer family comprises Double-Balanced Active (DBA), Double-Balanced Passive (DBP), Single-Balanced Active (SBA), and Single-Balanced Passive (SBP) designs. For Power Amplifiers (PA), we evaluate Class-B (ClassBPA), Class-E (ClassEPA), Doherty (DohPA), and Distributed (DPA) topologies. The Voltage Amplifier (VA) class consists of Common-Gate (CGVA), Common-Source (CSVA), Cascode (CVA), and Split-Phase (SPVA) variants. Finally, the Voltage-Controlled

Oscillator (VCO) group covers Intermediate-Frequency (IFVCO), Cross-Coupled (CCVCO), Colpitts (ColVCO), and Ring (RVCO) oscillators. The table also lists the specific performance metrics extracted and evaluated for each respective family.

Table 6: Benchmark circuits present in the initial dataset used by the ThunderGen generative pipeline.

Family	#Topo	Topologies	Performance metrics
LNA	4	CGLNA, CLNA, CSLNA, DLNA	DCP, PGain, S11, NF, BF
Mixer	4	DBA, DBP, SBA, SBP	DCP, CGain, NF, VSwg
PA	4	ClassBPA, ClassEPA, DohPA, DPA	DCP, PGain, PSAT, DE, S11, S22, PAE
VA	4	CGVA, CSVA, CVA, SPVA	DCP, VGain, BW
VCO	4	IFVCO, CCVCO, ColVCO, RVCO	DCP, OscF, TR, OutP, PN

A.2 Performance MAE: measured and estimated

For all the circuit families and for all the considered quality metrics, Table 7 part reports the MAE, as defined by Equation (1) in Section 4.1, for SPICE simulations, ThunderGen, DiffCkt Liu et al. (2025), and Falcon Mehradfar et al. (2025).

Table 7: Measured (Cadence Spectre) and estimated performance (ThunderGen, DiffCkt, and Falcon) across different circuit families and quality metrics. Estimates closer to Cadence Spectre results indicate higher fidelity of the performance estimator.

Family	**Metric**	**Cadence Spectre**	**ThunderGen**	**DiffCkt**	**Falcon**
LNA (442 circuits)	PowerGain	15.15	14.78	2.38	6.43
	NoiseFigure	8.46	8.04	0.88	4.74
	S11	4.41	3.50	1.81	9.95
	Bandwidth	3.22×10^{10}	2.56×10^{10}	4.09×10^{9}	1.14×10^{10}
	DCPowerConsumption	0.0289	0.0208	0.0059	0.0205
Mixer (445 circuits)	ConversionGain	97.73	96.37	7.19	25.77
	NoiseFigure	77.32	75.34	7.54	22.77
	VoltageSwing	0.3643	0.2410	0.0082	0.0178
	DCPowerConsumption	0.0466	0.0326	0.0003	0.0510
PA (728 circuits)	PowerGain	30.94	30.61	2.93	10.46
	PSAT	14.09	13.06	6.14	27.74
	DrainEfficiency	4.51	2.57	0.31	13.85
	PAE	57.56	15.80	0.22	9.53
	S11	2.08	1.34	0.51	15.04
	S22	4.67	4.27	1.73	12.51
	DCPowerConsumption	0.0309	0.0292	0.0088	0.0271
VA (314 circuits)	VoltageGain	6.78	6.33	4.53	11.20
	Bandwidth	5.19×10^{10}	4.25×10^{10}	2.13×10^{10}	2.12×10^{10}
	DCPowerConsumption	0.0141	0.0130	0.0011	0.0349
VCO (300 circuits)	OscillationFrequency	7.08×10^{10}	7.23×10^{10}	8.32×10^{9}	1.52×10^{10}
	OutputPower	12.32	11.60	1.98	8.78
	PhaseNoise	15.39	13.83	5.00	58.28
	TuningRange	3.99×10^{9}	2.43×10^{9}	1.95×10^{9}	2.00×10^{9}
	DCPowerConsumption	0.0345	0.0365	0.0264	0.0445

A.3 SOFTWARE CONFIGURATION

This appendix summarizes the software libraries and tools utilized across the ThunderGen methodolgy and experiments setup discussed in Section 4.1. The entire pipeline relies on a unified Python runtime, with the validation stages executing external calls to commercial EDA tools.

Table 8: Hardware, software, and commercial tool configuration used for all training, generation, and validation experiments.

Category	**Version / Configuration**
Commercial EDA Tools	
Simulator	Cadence Spectre 23.10.242
Process Design Kit (PDK)	GPDK045 v6.0 (45 nm)
Python Runtime & Core Libraries	
Python	3.10.4
PyTorch	2.10.0 (CUDA 12.8)
PyTorch Geometric	2.7.0
NumPy	2.2.6
Pandas	2.3.3
SciPy	1.15.3
scikit-learn	1.7.2
NetworkX	3.4.2

A.4 CIRCUIT GENERATIVE PIPELINE: DETAILED ARCHITECTURE

This appendix provides an in-depth look at the neural architectures that make up the generative pipeline used to create the multi-topology dataset within ThunderGen. To handle the complex, mixed-modality nature of electronic circuits, the automated generative pipeline decouples the generation process into distinct sequential stages: predicting the circuit scale, generating the discrete graph topology, and assigning continuous component sizings. The following paragraphs describe the internal architecture of each sub-module, as depicted in Figure 3 of Section 3.1, while Table 9 reports the complete hyperparameter configurations for reproducibility.

MLP Node Predictor - This network predicts the number of components the generated circuit will contain based on a 16-dimensional target performance vector $\mathbf{y}$. It consists of three fully connected blocks with hidden widths of $[256, 256, 128]$. Each block applies a `Linear` $\rightarrow$ `ReLU` $\rightarrow$ `Dropout` sequence. The final layer is a classification head producing a softmax distribution over $\{1, \ldots, 32\}$ possible node counts.

Denoising Network Backbone - Both the discrete and continuous diffusion networks share an identical graph-based denoising backbone. Sharing this architecture across fundamentally different tasks (categorical classification vs. numerical regression) is made possible by decoupling the representation learning from the final diffusion objective. The backbone acts solely as a structural feature extractor: it processes the noisy graph state at step t to produce rich, high-dimensional latent embeddings for each node and edge. Task specialization is then handled exclusively by the respective output heads.

This backbone is constructed from four `GraphAttentionLayer` blocks with a hidden dimension of 256. Each block utilizes 4-head self-attention with edge-conditioned attention bias, followed by a feed-forward network (4×256), residual connections, `LayerNorm`, and 0.1 dropout. The backbone employs Feature-wise Linear Modulation (FiLM) to inject the conditioning signal and Principal Neighbourhood Aggregation (PNA) for read-outs. The conditioning signal itself is a concatenation of a 128-dimensional sinusoidal time embedding and a performance-vector embedding generated by a 3-layer MLP ($16 \rightarrow 256 \rightarrow 256 \rightarrow 256$).

Discrete Diffusion - This network wraps the shared denoising backbone to generate node types and edge connectivity for a graph of size n. It features two output heads: one outputting per-node type logits across 9 component classes, and another outputting per-edge attribute logits over a 9-

dimensional edge feature space. It operates on a cosine β-schedule over $T{=}500$ steps and applies a net-count regularization auxiliary loss with a weight of 0.1.

Continuous Diffusion - This network denoises the 4-dimensional per-node continuous parameter block (device sizing). Using the same shared backbone, its regression head outputs a 4-d tensor per node. An active-dimension mask ensures the loss is only applied to valid sizing parameters for the specific component type. It also uses a cosine β-schedule over $T{=}500$ steps and applies an auxiliary performance-prediction loss with a weight of 0.1.

Table 9: Training and inference hyperparameters for the Generative Pipeline. Values are identical across all five circuit families.

Parameter	**Value**
Common Parameters	
Random seed	42
Train / val / test split	70 %/15 %/15 % (deterministic)
#component types	9 (MOSFET, cap, ind, res, balun, port, vsource, isource, ...)
max_nodes / max_ports / edge_dim	32 / 3 / 9
Diffusion steps T	500 (cosine schedule, $\beta \in [10^{-4}, 2{\cdot}10^{-2}]$)
Hidden dim / GNN layers	256 / 4
MLP Node Predictor	
Epochs / Batch size	100 / 1024
Optimizer / Loss	AdamW / Cross-entropy
Learning rate / Weight decay	$1{\cdot}10^{-4}$ / $1{\cdot}10^{-4}$
Discrete Denoising Diffusion	
Epochs / Batch size	200 / 768
Learning rate / Weight decay	$1.4{\cdot}10^{-3}$ / $1{\cdot}10^{-4}$
LR scheduler	ReduceLROnPlateau (factor 0.5, patience 5)
Early stopping patience	15 epochs
Auxiliary loss weight	0.1 (Net-count regularization)
Inference Batch size	512
Continuous Denoising Diffusion	
Epochs / Batch size	300 / 512
Learning rate / Weight decay	$2{\cdot}10^{-4}$ / $1{\cdot}10^{-3}$
LR scheduler	CosineAnnealingWarmRestarts ($T_0{=}20$, $T_{\text{mult}}{=}2$)
Early stopping patience	30 epochs
Auxiliary loss weight	0.1 (Performance prediction)
Inference Batch size	512

A.5 Performance Estimator Architecture

This section details the architectural design and training configuration of the performance estimator, which plays a critical role in providing rapid, differentiable feedback during the diffusion sampling process. Because running full SPICE simulations can be very expensive, this estimator acts as a highly efficient surrogate model. It is designed to be both fast enough to evaluate thousands of state transitions per second and accurate enough to pose as a replacement candidate of commercial tools.

To achieve this, we represent the intermediate circuits as attributed graphs where nodes represent electrical components and edges represent the topological nets connecting them. The input to the estimator encompasses node features (component types and continuous parameters), edge features (pin configurations), and global circuit-level descriptors. The structural and hyperparameter details for this module are summarized in Table 10 and Table 11, respectively.

HybridGNN - This network predicts the 3 to 7 family-specific performance metrics directly from the component-level circuit graph. To capture the complex topological dependencies of analog circuits, the component-level encoder employs five `GATv2Conv` layers (hidden dimension 192, 4 attention heads, edge-feature gating, and 0.1 dropout). This sequence culminates in a global mean + max

pooling read-out, which captures both average node properties and distinct, high-magnitude bottleneck features.

In parallel with the graph message passing, a global-feature MLP processes 224-dimensional structural and operating-point descriptors through a hidden width sequence of $320 \to 320 \to 160$. By processing these global features concurrently, the model ensures that critical macro-level constraints are not lost to over-smoothing. The read-outs from both the graph and global pathways are then concatenated into a 544-dimensional vector and projected by a $544 \to 224$ linear trunk. Finally, to mitigate negative transfer between competing electrical metrics (e.g., gain versus bandwidth), the latent representation is decoded by independent, per-metric MLP heads ($224 \to 112 \to 1$), with one distinct head allocated per family-specific target.

Table 10: Layer-by-layer architecture of the HybridGNN performance estimator. Shapes use N: vertices, M: directed edges, B: circuits per batch. The summary is split into two halves to save vertical space. Parameter counts are shown for the LNA estimator ($n_{\text{out}} = 5$).

Layer (type)	Output shape	Param #	Layer (type)	Output shape	Param #
input_x (Node)	$(N, 27)$	0	gat_block4 (GATv2)	$(N, 192)$	111,360
input_e (Edge)	$(M, 5)$	0	gat_block4_act	$(N, 192)$	0
input_ei (Idx)	$(2, M)$	0	gat_block4_norm	$(N, 192)$	384
input_g (Global)	$(B, 224)$	0	gat_block5 (GATv2)	$(N, 192)$	111,360
in_proj (Linear)	$(N, 192)$	5,376	gat_block5_act	$(N, 192)$	0
edge_proj (Linear)	$(M, 192)$	1,152	gat_block5_norm	$(N, 192)$	384
gat_block1 (GATv2)	$(N, 192)$	111,360	mean_pool	$(B, 192)$	0
gat_block1_act	$(N, 192)$	0	max_pool	$(B, 192)$	0
gat_block1_norm	$(N, 192)$	384	concat_pool	$(B, 384)$	0
gat_block2 (GATv2)	$(N, 192)$	111,360	gmlp_fc1 ($\to$320)	$(B, 320)$	72,000
gat_block2_act	$(N, 192)$	0	gmlp_fc2 ($\to$320)	$(B, 320)$	102,720
gat_block2_norm	$(N, 192)$	384	gmlp_fc3 ($\to$160)	$(B, 160)$	51,360
gat_block3 (GATv2)	$(N, 192)$	111,360	trunk_fc ($\to$224)	$(B, 224)$	122,080
gat_block3_act	$(N, 192)$	0	head_fc1 ($\times$5)	$(B, 112)$ ea.	25,200
gat_block3_norm	$(N, 192)$	384	head_fc2 ($\times$5)	$(B, 1)$ ea.	113
Total Params (LNA Family):			**1,039,973**		

Table 11: Training and inference hyperparameters for the Performance Estimator. Values are identical across all five circuit families.

Parameter	Value
Training	
Train / val / test split	$90\%/5\%/5\%$ (deterministic, seed 42)
Epochs	150 (early stopping, patience 15)
Batch size	64
Optimizer	AdamW
Learning rate / Weight decay	$2 \times 10^{-3}/1 \times 10^{-5}$
Loss	Huber ($\delta = 1$)
Auxiliary multi-task weight	0.1
Target normalization	QuantileTransformer (Gaussian output)
Inference	
Batch size	64
Pre-processing	log-transform (for log-trained metrics)
Post-processing	Inverse quantile normalization