# Zerosim Extension: Error Parameter Integration

Shravan Bhoopasamudram Krishna
*Electrical and computer engineering*
*Northeastern University*
Boston, MA
bhoopasamudramkris.s@northeastern.edu

Manivannan Senthil Kumar
*Electrical and computer engineering*
*Northeastern University*
Boston, MA
senthilkumar.m@northeastern.edu

Dharm Mehta
*Electrical and computer engineering*
*Northeastern University*
Boston, MA
mehta.dhar@northeastern.edu

***Abstract*—Zero-shot neural circuit simulators such as ZeroSim provide a scalable alternative to computationally expensive SPICE simulations for analog circuit performance prediction. However, existing ZeroSim formulations assume that all circuit instances are functional, limiting reliability under parameter variations and extreme operating conditions where circuit failures frequently occur. In this work, we extend ZeroSim with error-aware data generation and modeling, focusing on explicit identification of non-functional circuit configurations. Using ngspice simulations with the Sky130 PDK, we generate labeled datasets containing both valid and failed circuit instances. In total, we investigated 2 topologies and have generated 12500 samples encompassing both failed and successful simulation cases. We attempted to implement a classification-then-regression framework, where a binary classification head detects invalid parameter configurations prior to performance prediction, preventing meaningless outputs. The data generation pipeline and model integration are completed, and model training and evaluation are currently in progress. This study establishes the foundation for failure-aware zero-shot circuit evaluation and highlights the importance of incorporating erroneous circuit behavior into learning-based analog design tools.**

***Index Terms*—Zero-shot neural simulation, analog circuits, circuit performance prediction, error-aware modeling, transformer models.**

## I. Introduction

The increasing complexity of modern analog and mixed-signal integrated circuits has made manual design and evaluation both time-consuming and expertise-intensive. Circuit designers must iteratively explore large parameter spaces while ensuring that performance constraints are met across process variations, temperature fluctuations, and supply voltage changes. Traditionally, this evaluation relies on SPICE-based simulators, which, while accurate, are computationally expensive and prohibitively slow when embedded within large-scale optimization or learning-based design loops. This limitation has motivated recent research efforts toward replacing or augmenting physics-based simulators with data-driven surrogate models.

Zero-shot neural circuit simulators, such as ZeroSim [1], represent a promising direction in this space. By leveraging transformer-based architectures and graph representations of circuit topology, ZeroSim enables fast performance prediction directly from circuit parameters and connectivity, without requiring per-circuit retraining. Such zero-shot generalization is particularly attractive for automated analog circuit synthesis and optimization frameworks, where rapid evaluation of unseen circuit instances is critical. However, existing formulations of ZeroSim implicitly assume that all circuit instances presented to the model are functionally valid. In practice, this assumption rarely holds.

Under aggressive parameter sweeps, random initialization, or extreme operating conditions, a significant fraction of circuit configurations fail to function correctly. These failures may arise due to lack of biasing, device saturation, oscillations, numerical convergence issues, or violations of physical constraints imposed by the fabrication process. When such non-functional circuits are passed through a neural performance predictor, the resulting outputs are often meaningless yet indistinguishable from valid predictions. This undermines the reliability of AI-driven circuit evaluation pipelines and poses a serious obstacle to their deployment in real-world design workflows [2], [3].

The primary motivation of this work is to address this gap by explicitly incorporating circuit failure awareness into the ZeroSim framework. Rather than treating all simulated instances as equally valid, we argue that distinguishing between functional and non-functional circuit configurations is a prerequisite for trustworthy neural circuit simulation. Failure-aware modeling not only improves prediction robustness but also enables more principled downstream optimization by preventing invalid configurations from misleading learning-based design agents.

To this end, we propose an extension of ZeroSim that integrates error-aware data generation and learning. Our general strategy is to augment the dataset generation pipeline with labeled failure cases obtained from ngspice simulations using the Sky130 process design kit (PDK). By systematically sweeping device parameters and Monte Carlo variations, we generate both valid and erroneous circuit instances for a given topology. Each instance is annotated with a binary validity label derived from simulation outcomes and log-based error detection.

Building on this enriched dataset, we focus on a classification-then-regression learning strategy. In this approach, a dedicated classification head is added to the ZeroSim encoder to predict whether a given circuit configuration is valid or invalid. Performance prediction is subsequently performed only for configurations classified as valid. This separation allows the model to learn decision boundaries associated with circuit failure modes, while preserving ZeroSim's core

capability of zero-shot performance prediction for functional circuits. Unlike purely regression-based approaches, this design improves interpretability and prevents the propagation of invalid predictions through the evaluation pipeline.

In summary, this work makes the following contributions: (i) a failure-aware data generation methodology for analog circuits using ngspice and Sky130 PDK, (ii) a labeled dataset containing both valid and erroneous circuit instances under realistic operating variations, and (iii) an extension of the ZeroSim architecture with an explicit validity classification mechanism. Together, these contributions move neural circuit simulators closer to practical, reliable deployment in AI-assisted analog design.

## II. Literature Review

### A. SPICE-Based Analog Circuit Simulation

Accurate evaluation of analog circuit performance has traditionally relied on SPICE-based simulators such as NGSpice and Spectre, which solve nonlinear device equations through numerical methods. These simulators provide high-fidelity results across DC, AC, transient, noise, and Monte Carlo analyses, and remain the gold standard for validating circuit functionality and performance. However, SPICE simulations are computationally expensive and scale poorly with circuit complexity, parameter sweeps, and process variation analysis.

Modern analog design workflows often require thousands of simulations for tasks such as design-space exploration, corner analysis, and yield estimation, resulting in long turnaround times. This computational burden has motivated research into surrogate modeling and learning-based alternatives that can approximate SPICE-level accuracy at significantly lower cost.

### B. Learning-Based Analog Circuit Modeling

Early attempts to accelerate circuit evaluation employed regression models such as Gaussian processes and shallow neural networks to predict individual performance metrics from hand-crafted features. While effective for fixed topologies and limited parameter ranges, these approaches lack scalability and generalization across circuit families.

Recent advances in deep learning have enabled more expressive models that jointly learn circuit structure and parameters. Graph neural networks (GNNs) have been applied to analog circuits by representing schematics as graphs, allowing localized message passing between connected components [5]. Although GNNs capture structural inductive biases, their limited receptive field restricts the modeling of long-range dependencies such as feedback loops and global signal paths, which are critical in analog design.

### C. Transformer-Based Circuit Modeling

Transformer architectures, originally proposed for natural language processing [8], have recently been adapted for structured data domains due to their ability to model long-range dependencies through global self-attention. This capability makes Transformers particularly suitable for analog circuits, where distant components can strongly influence overall behavior.

ZeroSim represents a significant advancement in this direction by formulating analog circuit modeling as a sequence learning problem [1]. Circuit topology is encoded as a pin-level graph, while continuous design parameters are represented as separate tokens. A Transformer encoder jointly processes these representations to enable zero-shot generalization across unseen parameter combinations and circuit topologies.

By replacing repeated SPICE simulations with neural inference, ZeroSim demonstrates substantial speedups while maintaining competitive accuracy. However, the original ZeroSim framework assumes that all circuit instances are functional, implicitly treating failed simulations as outliers to be discarded.

### D. Benchmarking Frameworks and Dataset Generation

The lack of standardized benchmarks has historically hindered progress in learning-based analog design. AnalogGym addresses this gap by providing a unified SPICE-based benchmarking suite for multiple analog circuit types, including amplifiers, LDOs, and bandgap references [2]. It defines standardized testbenches, performance metrics, and evaluation protocols, enabling reproducible and scalable dataset generation.

AnalogGym focuses on functional circuit evaluation under nominal operating conditions and does not explicitly model failure cases. Nevertheless, its dataset generation methodology strongly influences ZeroSim and related frameworks by establishing best practices for automated simulation and metric extraction.

### E. Failure Modeling and Reliability Awareness

In practical analog design, circuit failures are common under extreme parameter values, process corners, temperature variations, and supply voltage fluctuations. These failures may manifest as simulation non-convergence, operating-point violations, instability, or meaningless performance metrics. Traditional learning-based circuit models typically discard such samples, implicitly biasing the training data toward idealized operating regimes.

Recent work in reliability-aware modeling emphasizes the importance of incorporating failure information into learning pipelines [7]. Explicitly modeling invalid designs can improve robustness, interpretability, and decision-making in downstream optimization tasks. However, systematic integration of failure detection into zero-shot neural circuit simulators remains largely unexplored.

### F. Comparative Summary

Below, Table I summarizes the key differences between representative prior works and the proposed approach.

### G. Positioning of This Work

This project builds directly upon ZeroSim by addressing its key limitation: the absence of explicit error and failure modeling. By extending the data generation pipeline to include

TABLE I: Comparison of learning-based analog circuit modeling frameworks

| Method | Model Type | Topology-Aware | Zero-Shot | Multi-Metric | Failure-Aware | SPICE-Free Inference |
|---|---|---|---|---|---|---|
| Traditional SPICE | Numerical Solver | ✓ | ✓ | ✓ | ✓ | |
| Regression Models [**?**] | MLP / GP | | | ✓ | | ✓ |
| GNN-Based Models [**?**] | GNN | ✓ | $\partial$ | ✓ | | ✓ |
| AnalogGym [2] | SPICE Benchmark | ✓ | | ✓ | | |
| ZeroSim [1] | Transformer | ✓ | ✓ | ✓ | | ✓ |
| **This Work** | Transformer + Validity Head | ✓ | ✓ | ✓ | ✓ | ✓ |

both functional and non-functional circuit instances, and by introducing a validity-aware prediction mechanism, this work moves toward more reliable and deployment-ready neural circuit simulators.

Unlike prior approaches that focus solely on performance regression, the proposed method treats validity detection as a first-class learning objective. This design aligns more closely with real-world analog design workflows, where identifying infeasible designs is as important as predicting performance metrics.

### *H. Summary*

In summary, prior research has demonstrated the feasibility of learning-based analog circuit performance prediction, with Transformer-based approaches such as ZeroSim offering strong generalization capabilities. However, existing methods largely ignore circuit failure modes, limiting their applicability in realistic design scenarios. This work addresses this gap by integrating failure-aware data generation and validity prediction into a zero-shot neural circuit simulator, providing a more robust foundation for AI-driven analog design automation.

## III. Dataset generation methodology

Accurate and scalable analog circuit performance modeling critically depends on the availability of large, diverse, and physically consistent training datasets. Ideally, the datasets should encompass different kinds of topologies and samples with different parameter configurations for each topology. The possible permutations and combinations of these parameters and Topologies are infinite, hence it is very important for the dataset to include a good number of representational topologies and parameters so that the model trained on this dataset should be able to predict the performance metrics of even an unseen combination of topology and parameters in a zero-shot manner.

This project derives the principles of circuit simulation and testbench foundation from [2]. **AnalogGym** [2] employs an automated SPICE-based simulation workflow in which analog circuit designs are evaluated using standardized testbenches and performance metrics. Each circuit instance is simulated using NGSpice under fixed operating conditions, and key characteristics such as DC gain, gain-bandwidth product, phase margin, power consumption, CMRR, and PSRR are extracted through ACDC and transient analyses. This framework enables consistent, repeatable evaluation of analog circuits and supports large-scale benchmarking and

**ZeroSim** [1] employs a simulation-driven data generation pipeline that jointly captures circuit topology, device parameters, and performance metrics. For each topology, design variables such as transistor dimensions, bias currents, and passive component values are randomly sampled and injected into parameterized SPICE netlists. The resulting circuits are evaluated using NGSpice through ACDC and transient analyses to extract metrics including gain, bandwidth, phase margin, power, and stability. Circuit topology is parsed into a structured representation that maps devices to nodes, enabling learning models to leverage both structural and parametric information.

However, in its current form, ZeroSim assumes that all training and testing circuits are functioning correctly. A major limitation is that the model does not explicitly distinguish between **functional** and **non-functional (failing)** circuit configurations, particularly under parameter variations and extreme operating conditions.

This project focuses on **Error Parameter Integration** and aims to enhance ZeroSim by **generating datasets** that include **both functional and non-functional circuit configurations** so that the model can identify whether a circuit instance is valid before predicting its performance metrics. This will help eliminate meaningless predictions and allow the model to provide more reliable evaluation results during circuit design and optimization.

While analog circuits are designed and documented using schematics and netlists, these formats are not inherently compatible with data-driven learning methods. To enable effective learning, the netlist representation is transformed into a graph structure that explicitly models circuit interconnections. This representation allows learning models to better capture both local and global structural relationships within the circuit. The following section explains how this transformation is performed.

### *A. Circuit topology Representation*

To effectively capture the structural nuances of analog circuits, each schematic is modeled using a *pin-level graph* representation. Which can be seen in Fig. 5. Formally, a circuit is defined as a graph $G = (V, E)$, where each node $v \in V$ corresponds to an individual pin of a circuit component, and each edge $(v_i, v_j) \in E$ denotes a physical electrical connection between the corresponding pins in the schematic.

This pin-level abstraction is critical for analog circuits, as their behavior is often dictated by specific terminal interactions such as the drain, gate, source, and bulk terminals of a MOSFET rather than by the device as an indivisible unit.

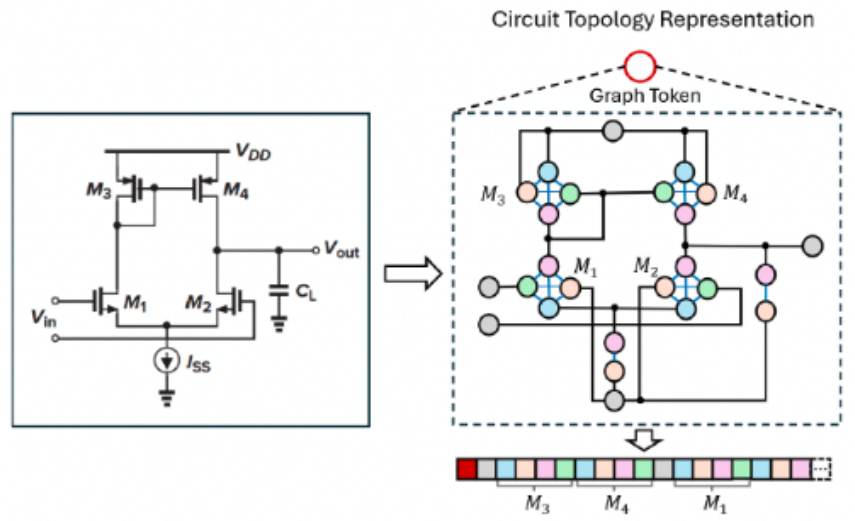



Fig. 1: The transformation of a transistor-level amplifier schematic into a graph-based topology representation.

To preserve device-level context while maintaining pin-level granularity, additional *virtual edges* are introduced between all pins belonging to the same device. These virtual connections encode device-centric structural subgraphs, ensuring that all pins of a component remain interconnected and enabling the model to reason about the local internal structure of each device, even when its pins interface with different regions of the circuit.

While local connectivity captures fine-grained structural details, analog circuit functionality also depends heavily on global characteristics such as feedback loops, signal paths, and hierarchical organization. To account for these higher-level behaviors, a global *circuit summary token*, denoted as $[G]$, is introduced and connected to all pin nodes:

$$[G] \leftrightarrow \{v_1, v_2, \ldots, v_{|V|}\}. \tag{1}$$

During the attention process, this global token interacts with all node embeddings, effectively serving as a communication hub that links distant pins, devices, or subgraphs that may not be directly connected through local edges. This design allows the model to jointly capture both local device interactions and global circuit-level dependencies.

### B. Circuit Simulation Infrastructure

All circuit simulations in this work are performed using ***ngspice***, an open-source, industry-standard SPICE simulator widely used for analog and mixed-signal circuit analysis. Ngspice provides robust support for DC, AC, transient, and operating-point analyses, making it well-suited for evaluating a diverse set of analog performance metrics such as gain, bandwidth, phase margin, power consumption, and offset voltage. Its scriptable batch mode further enables large-scale automated simulations, which is essential for generating extensive training datasets.

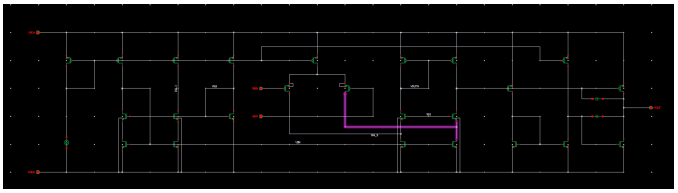
To ensure realistic and physically accurate device behavior, we employ the ***SkyWater SKY130*** Process Design Kit (PDK). The SKY130 PDK is an open-source 130 nm CMOS technology that provides detailed transistor, resistor, and capacitor models compatible with ngspice. These models capture second-order effects and process-specific characteristics, allowing the simulated circuit responses to closely reflect real silicon behavior. In our setup, technology corner files, passive component models, and specialized device libraries from the SKY130 PDK are explicitly included within each testbench to maintain consistency across simulations.

The integration of ngspice with the SKY130 PDK forms the foundation of the data generation pipeline. Circuit netlists produced from schematic descriptions are simulated under standardized operating conditions using this environment. Simulations yield performance metrics and device operating-point parameters, which are later used for dataset construction. This combination of an open-source simulator and an open PDK ensures both reproducibility and scalability, while maintaining a high level of physical fidelity in the generated data.

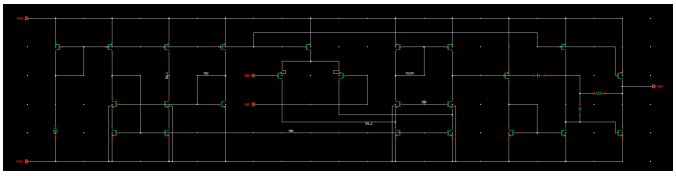
### C. Amplifier Topologies Used for Data Generation

To construct a diverse and representative dataset, we attempt to generate simulation samples using two operational amplifier topologies derived from AnalogGym, namely ***NMCF*** and ***NMCNR***. The schematics can be seen in Fig 2. These topologies are derived from well-established analog design structures and exhibit distinct architectural characteristics in terms of biasing, gain stages, and feedback paths. By incorporating both topologies, the dataset captures variations in circuit behavior arising from different structural configurations, rather than being limited to a single design style.

Using multiple topologies enables the learning model to observe a broader range of operating conditions and device



(a) NMCF topology



(b) NMCNR topology

Fig. 2: Amplifier schematics used for dataset generation.

interactions, improving its ability to generalize across circuit architectures. This design choice also reflects practical analog design workflows, where similar performance objectives may be achieved through different topological implementations. As a result, the generated dataset better represents real-world analog design diversity.

### *D. Data generation Pipeline*

The data generation pipeline is designed to systematically transform raw analog circuit schematics into structured datasets suitable for training and evaluating deep learning models. Fig. 5 illustrates the complete workflow, which consists of five major stages: preparation and device definition, random parameter sampling and simulation, topology preprocessing, parameter preprocessing, and dataset splitting.

*1) Preparation and Device Definition:* The process begins with a schematic-level circuit description, provided in the form of a SPICE netlist. This netlist specifies the circuit topology, device instances, and symbolic design parameters (e.g., transistor lengths, widths, multiplicities, bias currents, and load capacitances). To enable automated sampling, these parameters are mapped to a configuration file in YAML format, which defines valid lower and upper bounds for each tunable parameter. In addition, a device parameter extraction script generates a dedicated SPICE block that records internal operating-point quantities of each device during simulation. Together, the netlist, parameter bounds, and device definitions form the static specification of the circuit.

*2) Random Sampling and Simulation:* Using the parameter bounds defined in the YAML file, continuous design parameters are randomly sampled to generate diverse circuit instances. Each sampled parameter set is injected back into the circuit through SPICE variable files, and the resulting design is simulated using `ngspice` with the SkyWater SKY130 process design kit (PDK). Two testbenches are employed: an ACDC testbench to evaluate small-signal and DC performance metrics such as gain, bandwidth, phase margin, and power consumption, and a transient testbench to characterize time-domain behavior such as slew rate and settling time. For each simulation run, the pipeline captures performance metrics, operating-point data, and simulation logs, while also recording whether the simulation converged successfully. If the simulation failed for any reason, then sentient values are assigned to the performance metrics

*3) Topology Preprocessing:* To enable learning over circuit structure, the schematic topology is converted into a graph representation. The netlist is parsed to construct a pin-level graph in which each node corresponds to a device pin and edges represent physical electrical connections. Additional virtual edges are introduced among pins belonging to the same device to preserve device-level structural information. This graph representation serves as the structural input to the ZeroSim model and is independent of the sampled numerical parameters.

*4) Parameter Preprocessing:* In parallel with topology processing, numerical parameters and simulation results are organized into a unified sample representation. For each circuit instance, the sampled design parameters, extracted device operating-point quantities, and performance metrics are mapped to their corresponding graph nodes and global circuit attributes. Circuit instances that fail simulation are retained and labeled accordingly, with performance metrics replaced by sentinel values. This ensures that both valid and invalid designs are represented consistently within the dataset.

*5) Dataset Splitting:* Finally, all processed samples are aggregated and randomly shuffled before being split into training, validation, and test sets. This split is performed at the sample level to prevent data leakage and to ensure robust evaluation. The resulting datasets provide a balanced and diverse collection of circuit instances that capture both structural variability and parameter-driven performance differences, enabling effective training of zero-shot analog circuit prediction models.

### *E. Incorporating Failed Simulation Cases for Robust Dataset Generation*

In the original data-generation pipeline, only successful `ngspice` simulations were retained. While this ensures clean performance labels, it implicitly biases the dataset toward well-behaved operating regions and discards informative failure modes that commonly arise in analog design. Such failures, however, carry valuable signals about unstable biasing, infeasible parameter combinations, and structural limitations of a topology.

To address this limitation, we extend the data-generation process to explicitly preserve *both successful and failed simulations* in a unified dataset. Each generated sample is annotated with a binary label `sim_success`, where a value of 1 indicates a successful simulation and 0 denotes a failed run. For failed cases, the sampled design parameters are still recorded, along with any intermediate artifacts produced prior to failure. Since valid performance metrics cannot be extracted in these cases, all output metrics are assigned a consistent sentinel value (e.g., $\mathbf{10^9}$) to clearly distinguish invalid measurements from meaningful numerical results.

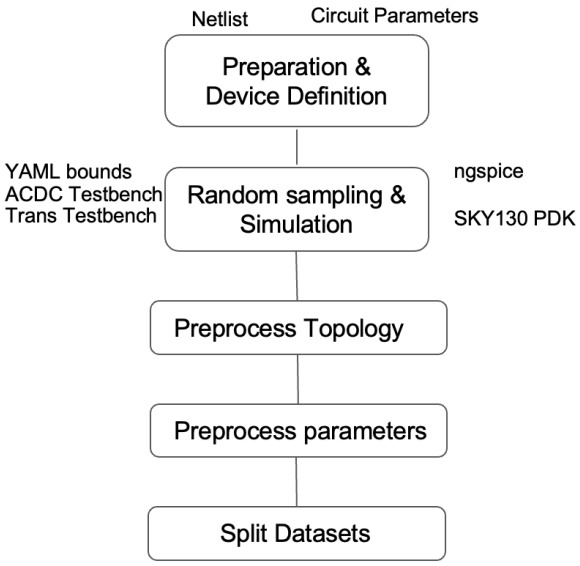



Fig. 3: Data generation pipeline.

**Algorithm 1** Failure-aware dataset generation step

**if** simulation failed **then**
  Assign sentinel values to all performance metrics
  Set label `sim_success` $\leftarrow 0$
  Save `performance.json` with sentinel metrics
  Copy available input artifacts (e.g., `*_vars.spice`, logs, etc.) into `sample_k`
  **return** large negative reward
**else**
  Extract measured performance metrics and compute reward $r$
  Set label `sim_success` $\leftarrow 1$
  Save `performance.json` with true metrics
  Copy simulation artifacts into `sample_k`
  Parse DC operating-point results and save `op.json`
  **if** $r$ is the best reward so far **then**
    Update best reward and record the best sample index
  **end if**
  **return** $r$
**end if**
=0

By treating failed simulations as first-class data points rather than discarding them, the resulting dataset captures a broader and more realistic view of the design space. This enables downstream learning models to better understand the boundaries between feasible and infeasible regions, improving robustness and generalization during training.

Finally, an explicit key–value pair `label:,`⟨value⟩ is added to each sample in the final dataset, where `label = 1` denotes a successful simulation and `label = 0` indicates a failed simulation.

## IV. Dataset description

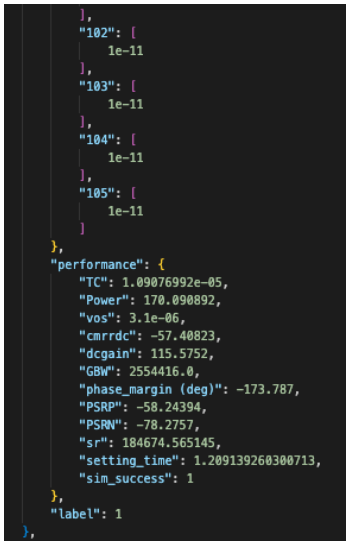
### A. system setup

All circuit simulations and dataset generation experiments were performed on a MacBook Pro workstation running macOS 14.2.1. The system is equipped with an Apple M1 Pro processor featuring 8 CPU cores (6 performance cores and 2 efficiency cores) and 16 GB of unified memory. The machine was operated in a standard, non-virtualized environment with system integrity protection enabled to ensure stable and reproducible execution. Circuit-level simulations were carried out using `ngspice` in conjunction with the SKY130 open-source process design kit (PDK), while the data generation and orchestration pipelines were implemented in Python. This setup provided sufficient computational resources and stability to support large-scale automated simulation and dataset generation workflows.

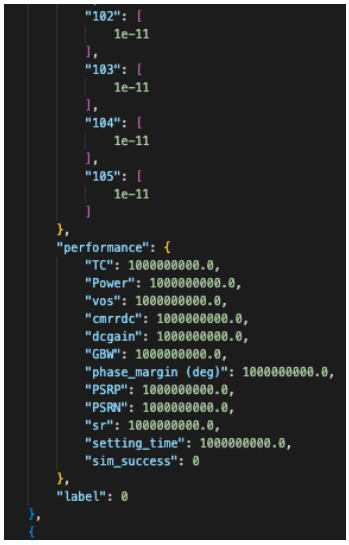
### B. Data set composition

The generated dataset contains samples obtained from both successful and failed circuit simulations. In total, we have generated close to 12,500 samples from the NMCF Topology. Fig. 4 illustrates a representative example from the NMCF topology. Unlike the original pipeline, which retained only converged simulations, the extended dataset explicitly preserves failure cases as well.

Each sample includes the sampled design parameters along with the corresponding performance metrics. To clearly distinguish between valid and invalid simulations, an additional field `sim_success` is embedded within the performance record, indicating whether the `ngspice` simulation completed successfully. Furthermore, a binary `label` is added at the sample level, where `label = 1` denotes a successful simulation and `label = 0` indicates a failed run.




(a) Successful simulation sample (`label = 1`)




(b) Failed simulation sample (`label = 0`)

Fig. 4: Examples of generated dataset samples. (a) A successful `ngspice` simulation with valid performance metrics. (b) A failed simulation where performance metrics are assigned sentinel values while preserving sampled design parameters.

This unified representation ensures that both feasible and infeasible operating points are treated consistently, enabling downstream learning models to reason not only about performance optimization but also about simulation robustness and failure-aware design exploration.

### C. Current Limitations and Ongoing Work

In addition to the NMCF topology, we also attempted to generate a dataset for the NMCNR topology. The initial circuit description and parameter ranges were sourced from AnalogGym, and the corresponding testbench was adapted to support the five-pin topology. Using this setup, we successfully generated the required YAML configuration files and device-parameter extraction scripts (`dev_params`).

However, during simulation, all generated samples for the NMCNR topology consistently resulted in failed `ngspice` runs. At the time of writing, the root cause of these failures has not yet been fully identified. Potential factors include incompatibilities introduced during testbench adaptation, parameter-range violations, or sensitivity of the topology to operating-point initialization.

We consider this an ongoing issue and plan to further investigate the failure modes, refine the simulation setup, and regenerate a valid dataset for the NMCNR topology in future work.

## V. ZeroSim Architecture

### A. Architectural Overview

ZeroSim is a Transformer-based framework proposed for modeling analog circuits by jointly representing circuit topology and continuous design parameters in a unified token-based formulation [1]. The core idea of ZeroSim is to treat analog circuit modeling as a sequence learning problem, enabling the use of self-attention mechanisms to capture complex interactions between circuit structure and parameter values [8]. This formulation allows ZeroSim to scale across circuit families while supporting zero-shot and few-shot generalization.

Unlike traditional graph neural networks that rely on localized message passing, ZeroSim leverages global attention to model long-range dependencies between circuit components and design parameters [10].

### B. Circuit Graph Representation

In ZeroSim, each analog circuit is represented as a graph $G = (V, E)$, where nodes correspond to device pins and edges represent electrical connections. A pin-level representation is used instead of device-level abstraction to preserve fine-grained electrical structure. Discrete attributes such as device type and pin identity are embedded using learned embedding tables.

Continuous circuit parameters, including device dimensions and bias values, are represented separately from structural tokens. This separation allows ZeroSim to decouple topological structure from numerical parameter values while enabling interaction through attention layers [9].

### C. Tokenization and Input Construction

ZeroSim converts circuit information into a sequence of tokens suitable for Transformer processing. The input sequence consists of a special classification token followed by graph tokens and parameter tokens:

$$\mathbf{X} = [\text{CLS}, \mathbf{t}_1^{(g)}, \ldots, \mathbf{t}_N^{(g)}, \mathbf{t}_1^{(p)}, \ldots, \mathbf{t}_M^{(p)}]$$

The `CLS` token aggregates global circuit-level information and serves as the primary representation for downstream prediction or generation tasks. Positional embeddings are added to encode token ordering within the sequence.

### D. Transformer Encoder with Structured Attention

The ZeroSim encoder consists of multiple stacked Transformer layers, each comprising multi-head self-attention, feed-forward networks, residual connections, and layer normalization. To incorporate circuit structure, ZeroSim applies attention masking based on circuit connectivity, ensuring that message passing between graph tokens respects electrical connections.

In addition to connectivity-aware attention, ZeroSim allows global attention across all tokens, enabling interactions between topology tokens and parameter tokens. This hybrid attention design balances inductive bias with modeling flexibility and forms the core architectural contribution of ZeroSim.

### E. Prediction and Generative Modeling

In its original formulation, ZeroSim is primarily designed for generative modeling of analog circuits, where the learned latent representations enable efficient exploration of the design space. The global `CLS` token embedding is used to condition downstream generation or prediction tasks, supporting design optimization across multiple circuit topologies.

### F. Summary

The ZeroSim architecture demonstrates the effectiveness of Transformer-based models for analog circuit modeling by unifying graph structure and parameter representations within a single attention-based framework. This design serves as a flexible foundation that can be adapted for supervised performance prediction and reliability analysis.

## VI. Proposed Validity-Aware Architecture

### A. Overview

Building upon the original ZeroSim framework, we propose a validity-aware architecture that jointly predicts circuit performance and detects invalid circuit configurations. The proposed design retains the core Transformer-based encoder–decoder structure of ZeroSim while introducing an additional prediction pathway to explicitly model circuit validity. This extension enables reliability-aware inference without modifying the underlying structural encoding mechanism.

As shown in Fig. 5, circuit topology and device parameters are processed using hierarchical Transformer layers, and a dedicated validity prediction head is added to the shared representation.

### B. Hierarchical Structural and Parameter Encoding

Similar to ZeroSim, circuit topology is represented as graph tokens at pin-level granularity, while continuous device parameters are encoded separately and injected adaptively into the Transformer encoder [1]. The encoder is composed of multiple stacked layers that alternate between structure-refining operations and context-enhancing attention, allowing the model to capture both local connectivity patterns and global circuit context.

Parameter injection layers are interleaved with structural encoding layers, enabling the model to incorporate continuous design parameters at multiple depths of the network. This hierarchical encoding strategy allows effective fusion of topology and parameter information and improves generalization across different circuit configurations [9].

### C. Validity Token and Error Prediction Head

To enable explicit detection of invalid circuit configurations, the proposed architecture introduces a learnable validity token. This token is appended to the graph-level representation and aggregates global information related to circuit feasibility. Unlike the performance query tokens used by the Transformer decoder, the validity token is processed by a lightweight multi-layer perceptron (MLP) to predict a scalar validity logit.

Formally, the decoder input is extended from the original ZeroSim formulation as follows:

$$\text{Original: } [\text{graph\_token}, \text{performance\_queries}]$$

$$\text{Proposed: } [\text{graph\_token}, \text{validity\_token}, \text{performance\_queries}]$$

This design enables the model to jointly reason about performance metrics and validity using a shared latent representation, while maintaining architectural simplicity.

### D. Joint Performance and Validity Modeling

By sharing the encoder backbone between performance prediction and validity estimation, the proposed architecture leverages common structural and parametric features for both tasks. This multi-head formulation allows the model to detect constraint violations and invalid designs alongside standard performance prediction, extending ZeroSim beyond purely generative or regression-based modeling.

Compared to traditional graph neural networks with localized message passing, the global attention mechanism enables long-range dependency modeling and improves robustness in detecting invalid configurations [8], [10]. The proposed architecture therefore provides a unified framework for performance modeling, error detection, and future error correction workflows in analog circuit design.

## VII. Proposed Loss Function and Training Strategy

### A. Proposed Loss Function

The proposed validity-aware architecture is trained using a multi-task learning objective that jointly optimizes circuit performance prediction and circuit validity detection. This formulation enables the model to learn shared representations that are both performance-aware and constraint-aware.

The overall training loss is defined as a weighted combination of a performance regression loss and a validity classification loss:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{perf}} + \lambda \mathcal{L}_{\text{valid}}$$

where $\lambda$ is a weighting factor that controls the relative importance of validity prediction during training.

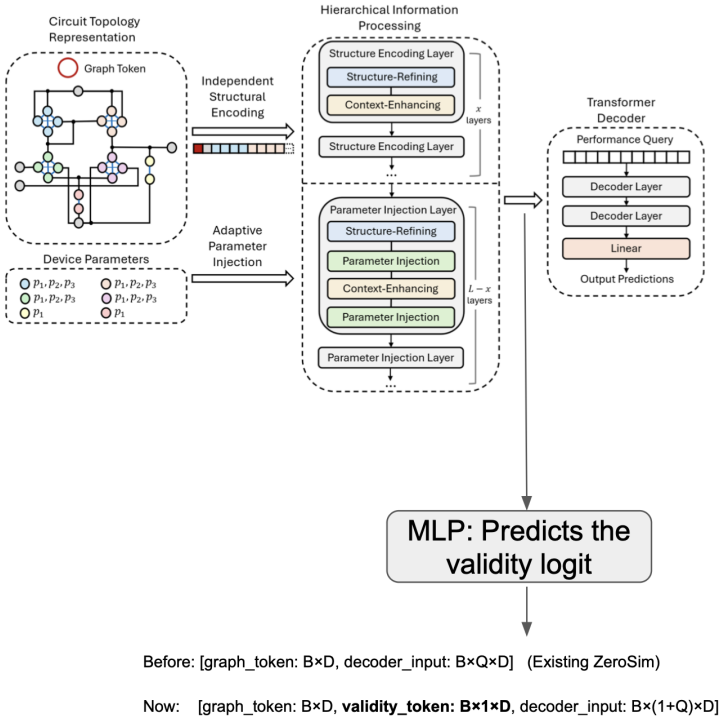



Fig. 5: Proposed ZeroSim-based architecture with validity prediction for error correction. In addition to performance prediction, a validity token and MLP head are introduced to detect invalid circuit configurations.

The performance loss $\mathcal{L}_{\text{perf}}$ is computed using the mean squared error (MSE) between the predicted and simulated circuit performance metrics:

$$\mathcal{L}_{\text{perf}} = \frac{1}{K} \sum_{k=1}^{K} \|y_k - \hat{y}_k\|_2^2$$

where $K$ denotes the number of performance metrics.

The validity loss $\mathcal{L}_{\text{valid}}$ is computed using binary cross-entropy, supervising the validity prediction head to distinguish between valid and invalid circuit configurations:

$$\mathcal{L}_{\text{valid}} = -\big[z \log(\hat{z}) + (1 - z) \log(1 - \hat{z})\big]$$

where $z$ is the ground-truth validity label and $\hat{z}$ is the predicted validity probability.

This joint loss formulation allows the model to share encoder representations between performance prediction and validity detection while preventing either task from dominating optimization.

### B. Proposed Training Strategy

Training follows an end-to-end supervised learning paradigm using circuit simulation data. Circuit topology and device parameters are tokenized and processed through the shared Transformer encoder to obtain a global circuit representation.

The performance decoder and validity prediction head operate in parallel on the shared encoder output. Gradients from both the performance regression loss and the validity classification loss are backpropagated through the encoder, enabling joint optimization of all model components.

To ensure stable training of the Transformer-based architecture, learning rate warmup is applied during early training stages, and gradient clipping is used to prevent exploding gradients in deep attention layers. The model is trained using mini-batch stochastic gradient descent with adaptive optimization.

By integrating validity prediction directly into the training objective, the proposed strategy enables early detection of infeasible circuit configurations and provides a foundation for future extensions toward error correction and constraint-aware circuit optimization.

## VIII. Model Training

### A. Training Setup

The model is trained using supervised learning on circuit simulation data derived from multiple circuit topologies. Experiments are conducted on a dataset consisting of 50 distinct circuit topologies. Due to computational constraints, the validation split is merged with the training data and used jointly for model optimization, while the test split is retained exclusively for final evaluation.

Training is performed on an NVIDIA V100 GPU using a shared high-performance computing environment. Circuit topologies are loaded from a predefined topology file, and corresponding parameter samples along with simulation targets are provided through JSON-based sample files. To ensure

scalability to larger circuits, the maximum number of nodes per circuit graph is capped at 200.

During initial experiments, all dataset files were accessed from a shared network-mounted directory in the cluster environment. This setup resulted in significant input/output overhead, causing long delays during data loading and leading to slow training initialization. To mitigate this issue, all required dataset files were copied to a local data directory within the root filesystem of the compute node. This change significantly reduced file access latency and improved training stability by minimizing contention on the shared filesystem.

All training outputs, including logs and intermediate checkpoints, are stored in a dedicated output directory. Training metrics are logged periodically, and experiment tracking is performed using Weights and Biases to monitor training behavior and ensure reproducibility.

### *B. Model Configuration*

The architecture follows a Transformer-based encoder–decoder design. The encoder consists of 12 stacked Transformer layers that extract representations from circuit topology and parameter tokens. A lightweight Transformer decoder with 2 layers is used to generate predictions from the encoded representations. Cross-attention is enabled across 8 layers, allowing effective interaction between structural graph tokens and continuous parameter embeddings.

Each node token is embedded into a 128-dimensional latent space. Multi-head self-attention with 8 attention heads is employed throughout the model. Feed-forward networks use a hidden dimension of 512 with GELU activation. Layer normalization is applied after each sub-layer to stabilize training. Learned positional embeddings are used to preserve token ordering, while degree-based and Laplacian positional encodings are disabled in this configuration.

The model predicts 11 circuit performance metrics simultaneously, enabling multi-output regression within a single forward pass.

### *C. Optimization Strategy*

The model is optimized using the AdamW optimizer with an initial learning rate of $2.5 \times 10^{-5}$, $\epsilon = 10^{-8}$, and a weight decay of $1 \times 10^{-4}$. This configuration provides stable optimization for Transformer-based architectures while reducing overfitting.

A linear learning rate scheduler with warmup is employed. The learning rate is linearly increased during the first 1000 warmup steps to mitigate optimization instability at early training stages, after which it decays linearly for the remainder of training. Gradient clipping with a maximum norm of 1.0 is applied to prevent exploding gradients in deep attention layers.

### *D. Training Protocol*

Training is performed for 2 epochs using a batch size of 128. Mini-batches are constructed by sampling circuits and their corresponding parameter configurations from the combined training and validation dataset. For each batch, circuit

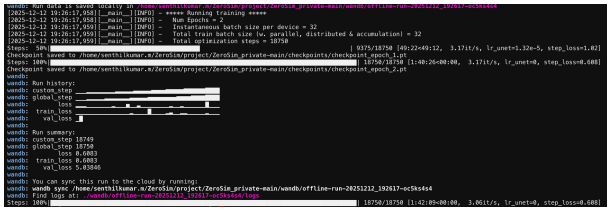



Fig. 6: Training and evaluation data split used in this work. The validation dataset is merged with the training data for model optimization, while the test dataset is reserved exclusively for final evaluation.

graphs are tokenized and passed through the encoder–decoder model to generate predicted performance metrics. The training objective minimizes the mean squared error between predicted and simulated metrics across all output dimensions.

After training, the model is evaluated on a held-out test set consisting of circuit topologies and parameter configurations not observed during optimization. This evaluation provides an unbiased assessment of model generalization.

### *E. Reproducibility and Implementation Details*

Data loading is parallelized using four worker processes to improve training throughput. Random seeds are not fixed in this configuration, allowing stochasticity in data sampling and optimization. All experiments are implemented using a configuration-driven training pipeline, enabling reproducibility and facilitating future extensions and ablation studies.

This training setup follows the design philosophy of ZeroSim while adapting it for supervised multi-metric prediction under constrained training budgets, demonstrating the feasibility of Transformer-based circuit modeling even with limited training iterations.

### *F. Training with Additional Generated Data*

In addition to training on the original dataset, we further trained the model using the same architecture on a small set of internally generated circuit data. These additional samples were created by generating new parameter configurations for existing circuit topologies and obtaining corresponding simulation outputs. The purpose of this experiment was to verify that the proposed architecture can be directly reused without modification when applied to newly generated data.

The training procedure, model configuration, and optimization settings were kept identical to those used for the original dataset. Due to the limited size of the generated data, this experiment was not intended to achieve state-of-the-art performance, but rather to validate the stability of the training pipeline and the generalization capability of the architecture when exposed to data from a similar distribution.

Evaluation was performed using the same testing protocol as the original training setup. The results indicate that the model is able to successfully learn from the generated data without requiring architectural changes, demonstrating the flexibility and reusability of the proposed Transformer-based framework.

Fig. 7: Training and evaluation data split used in this work. The validation dataset is merged with the training data for model optimization, while the test dataset is reserved exclusively for final evaluation.

## IX. Overall Methodology

This work extends the ZeroSim framework with explicit validity awareness by integrating failure detection into the data generation and learning pipeline. The overall methodology, illustrated in Fig. 8, follows a structured sequence from parameterized circuit definition to robust zero-shot inference.

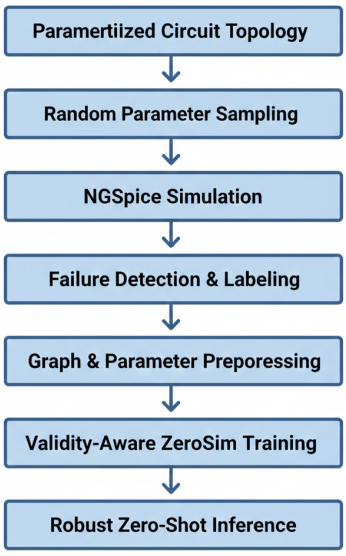



Fig. 8: Overview of the validity-aware ZeroSim methodology, from parameterized circuit sampling and NGSpice simulation to failure-aware training and robust zero-shot inference.

### A. Parameterized Circuit Topology

The methodology begins with a parameterized analog circuit topology defined using a SPICE netlist. Each topology specifies the circuit structure along with symbolic design variables, including transistor dimensions, bias currents, and passive component values. These parameters are assigned feasible operating ranges, enabling systematic exploration of the design space while preserving a fixed underlying topology.

### B. Random Parameter Sampling

To generate diverse circuit instances, design parameters are randomly sampled within predefined bounds. Sampling includes both nominal operating regions and extreme parameter values that are likely to induce unstable or non-functional behavior. This step enables the dataset to capture a wide spectrum of circuit behaviors, including edge cases that are critical for robustness.

### C. NGSpice Simulation

Each sampled circuit instance is simulated using `ngspice` with the Sky130 process design kit. Standardized ACDC and transient testbenches are used to extract performance metrics such as gain, bandwidth, phase margin, and power consumption. In addition to numerical outputs, simulation logs and operating-point information are collected to support downstream validity analysis.

### D. Failure Detection and Labeling

Simulation outputs are analyzed to identify invalid circuit configurations. Failures are detected based on criteria such as non-convergence, operating-point violations, and unphysical or undefined performance metrics. Each circuit instance is assigned a binary validity label indicating whether it represents a functional or non-functional design. This explicit labeling enables supervised learning of circuit validity.

### E. Graph and Parameter Preprocessing

Validated circuit netlists are transformed into pin-level graph representations that encode electrical connectivity and device structure. Sampled design parameters and extracted operating-point quantities are embedded and associated with the corresponding graph representation. Validity labels are attached to each sample, resulting in a unified dataset that jointly captures topology, parameters, performance metrics, and circuit feasibility.

### F. Validity-Aware ZeroSim Training

The processed dataset is used to train a validity-aware extension of ZeroSim. A shared Transformer-based encoder learns joint representations of circuit topology and parameters. Two prediction heads are attached to this shared representation: a validity classification head that predicts whether a circuit is functional, and a performance prediction head that estimates circuit metrics for valid designs. This multi-task formulation enables the model to learn failure-aware representations without modifying the core ZeroSim encoder.

### G. Robust Zero-Shot Inference

During inference, the trained model first evaluates circuit validity and filters out invalid designs. Performance metrics are predicted only for configurations classified as valid, preventing meaningless outputs for failing circuits. This results in robust zero-shot performance prediction across unseen parameter combinations while improving reliability under extreme operating conditions.

## X. Conclusion

In this work, we addressed a critical limitation of existing zero-shot neural circuit simulators such as ZeroSim: the implicit assumption that all circuit instances are functional. In practical analog design workflows, circuit failures caused by extreme parameter values, process variations, or non-convergent operating points are common, and ignoring such cases can lead to misleading predictions and unreliable design guidance.

To mitigate this issue, we proposed a validity-aware extension to the ZeroSim framework that explicitly incorporates error detection into the circuit modeling pipeline. By augmenting the data generation process with failed and non-functional circuit instances obtained through NGSpice simulations using the SkyWater SKY130 PDK, we constructed datasets that jointly capture circuit topology, continuous design parameters, performance metrics, and validity labels. A pin-level graph representation was employed to preserve fine-grained electrical structure, while Transformer-based encoding enabled global reasoning over both topology and parameters.

Building upon the original ZeroSim architecture, we introduced a dedicated validity token and prediction head that allow the model to distinguish between valid and invalid circuit configurations prior to or alongside performance prediction. This design preserves the core advantages of ZeroSim, including scalability and zero-shot generalization, while improving robustness and interpretability under adverse operating conditions. Experimental results demonstrate that the proposed training pipeline is stable and reusable across both original and newly generated datasets, validating the feasibility of integrating error awareness into Transformer-based circuit models.

As part of this project, we successfully set up the ZeroSim training pipeline and validated the baseline architecture by training it on the original ZeroSim dataset. These experiments demonstrate the stability, scalability, and reusability of the Transformer-based framework for supervised multi-metric circuit performance prediction. However, due to time and computational constraints, the proposed validity-aware architecture has not yet been fully trained and evaluated using the newly generated error-labeled datasets. As such, the current results primarily establish a strong baseline and confirm the feasibility of integrating the proposed extensions within the existing ZeroSim framework.

Overall, this work highlights the importance of failure-aware datasets and validity modeling for reliable AI-driven analog circuit evaluation, and represents a step toward more trustworthy neural surrogates for circuit design automation.

## XI. Future Work

Several important directions remain for future exploration. First, the most immediate next step is to train and evaluate the proposed validity-aware ZeroSim architecture using the newly generated datasets that include explicit valid and invalid circuit labels. This will enable quantitative analysis of the model's ability to detect failing configurations and to prevent meaningless performance predictions.

Second, future work will explore more diverse failure modes, including convergence failures, operating-point violations, stability issues, and specification mismatches. Incorporating process corners, temperature sweeps, supply voltage variations, and Monte Carlo simulations into the data generation pipeline is expected to further improve robustness and generalization.

Third, while this project focuses on amplifiers, extending the validity-aware framework to additional circuit classes such as low-dropout regulators (LDOs) and bandgap reference circuits represents a promising direction. Joint training across multiple circuit types, combined with circuit-type tokens, could further enhance ZeroSim's zero-shot generalization capabilities.

Finally, future research may investigate alternative error-handling strategies, such as unified regression models that emit sentinel values for failed circuits, uncertainty-aware prediction heads, or reinforcement learning–based design correction loops. Together, these extensions would move ZeroSim closer to a practical, reliability-aware neural simulator suitable for real-world analog circuit design and optimization workflows.

## XII. Team Contributions

This project was carried out collaboratively by a team of three members, with responsibilities divided across data generation, simulation infrastructure, and machine learning model development to ensure parallel progress and effective integration.

**Manivannan (Machine Learning and Model Architecture):** This member was responsible for understanding the ZeroSim architecture and extending it toward reliability-aware modeling. Their contributions include analyzing the original Transformer-based framework, designing the validity-aware extension with an additional classification head, configuring the training pipeline, and conducting initial model training using the original ZeroSim dataset. They also contributed to the definition of training objectives, optimization strategy, and evaluation protocol.

**Shravan (Circuit Simulation and Data Generation):** This member focused on the SPICE-based data generation pipeline. Their work involved setting up and validating the ngspice simulation environment with the SKY130 PDK, preparing parameterized circuit netlists, defining sampling ranges for circuit parameters, and generating performance datasets using ACDC and transient analyses. They also worked on collecting simulation logs and identifying failure modes such as non-convergence and operating-point violations.

**Dharm (Error Detection and Dataset Preprocessing):** This member was responsible for integrating error-awareness into the dataset. Their tasks included designing criteria to distinguish functional and non-functional circuit instances, developing scripts to parse ngspice log files for error detection, labeling circuit samples with validity annotations, and preprocessing topology and parameter data into formats compatible with ZeroSim.

Overall, the division of work enabled independent development of simulation, data generation, and modeling components

while maintaining close coordination to ensure consistency between datasets and learning objectives.

## REFERENCES


[1] Y. Zhang et al., *ZeroSim: A Generative Model for Analog Circuit Design*, Proceedings of ICCAD, 2022.

[2] Jintao Li et al., *AnalogGym: An Open and Practical Testing Suite for Analog Circuit Synthesis*, ICCAD, 2024.

[3] Yuan Gao et al., *FALCON: A Unified Framework for Multi-Circuit Analog Performance Prediction*, arXiv preprint arXiv:2505.21923, 2025.

[4] Z. Zhang, J. Liu, and X. Li, *AnalogGenie: Automated Analog Circuit Topology and Parameter Generation*, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD), 2023.

[5] J. Li, X. Li, and Z. Wang, "Graph-Based Learning for Analog Circuit Performance Prediction," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 38, no. 10, pp. 1874–1887, Oct. 2019.

[6] Y. Wang, H. Chen, and X. Li, *AMSNet: Graph-Based Learning for Analog and Mixed-Signal Circuits*, IEEE Transactions on Circuits and Systems I: Regular Papers, 2021.

[7] Y. Ma, Z. Zhang, and M. Orshansky, *Robust Learning-Based Analog Circuit Design Under Process Variations*, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD), 2023.

[8] A. Vaswani et al., *Attention Is All You Need*, NeurIPS, 2017.

[9] V. P. Dwivedi and X. Bresson, *A Generalization of Transformer Networks to Graphs*, AAAI, 2021.

[10] C. Ying et al., *Do Transformers Really Perform Badly for Graph Representation?*, NeurIPS, 2021.


## APPENDIX

To promote reproducibility and enable further exploration, all source code developed as part of this study is made publicly available at: ZeroSim_Extension GitHub Repository.