Review

Not peer-reviewed version

From Predict to Generate: A Cross-Layer Survey of Generative AI for IC Design and Manufacturing

Weiman Yan *, Ziyu Deng, Michael Molter, Ram Krishna, Nirjhor Rouf, Sohrab Sheikh Sofla, Ethan Thieme, Soumyadeep Chatterjee, Phoo Pyae Pyae Linn, Yuejiang Wen, Xavier Xiao, Zhuoer Zhang, Junsheng Huang, Yening Liu, Yujie Zhou, Paul Franzon, Madhavan Swaminathan, Elyse Rosenbaum, Aydin Aysu

Posted Date: 13 July 2026

doi: 10.20944/preprints202607.0874.v1

Keywords: generative AI; IC design; EDA; IC manufacturing

 doi:10.20944/preprints202607.0874.v1

Review

From Predict to Generate: A Cross-Layer Survey of Generative AI for IC Design and Manufacturing

Weiman Yan [1,*], Ziyu Deng [2], Michael Molter [1], Ram Krishna [1], Nirjhor Rouf [2], Sohrab Sheikh Sofla [3], Ethan Thieme [1], Soumyadeep Chatterjee [2], Phoo Pyae Pyae Linn [1], Yuejiang Wen [2], Xavier Xiao [2], Zhuoer Zhang [1], Junsheng Huang [1], Yening Liu [1], Yujie Zhou [1], Paul Franzon [2], Madhavan Swaminathan [3], Elyse Rosenbaum [1] and Aydin Aysu [2]

[1] Dept. of Electrical and Computer Engineering, University of Illinois Urbana-Champaign, Urbana, IL, USA
[2] Department of Electrical and Computer Engineering, North Carolina State University, Raleigh, NC,USA
[3] School of Electrical and Computer Engineering, Pennsylvania State University, University Park, PA, USA
* Correspondence: weimany2@illinois.edu

Abstract

AI tools used for chip design and manufacturing are moving from prediction to generation. This survey examines generative AI tools across the full silicon lifecycle. It covers analog/mixed-signal, logic and physical design, and manufacturing with yield and reliability. We organize recent work by what is generated—circuits, scripts, layouts, masks, and surrogates—and by how generation is grounded through retrieval, simulators, and physics. We give a clear taxonomy.We highlight emerging cross-stage dependencies linking early design choices to downstream manufacturability, and we identify where the literature has not yet closed those loops. We synthesize qualitative considerations for choosing retrieval-grounded LLM agents, task-specific generators, and physics-aware surrogates, while noting that controlled head-to-head evidence remains limited. We aim this work to be a broad entry point for researchers and practitioners who need a clear map and practical guidance on generative methods across the silicon lifecycle.

Keywords: generative AI; IC design; EDA; IC manufacturing

1. Introduction

Chip design is under strain. Designs are larger, rules are tighter, and time-to-market is unforgiving. First-time-right tape-outs are slipping, and verification loads keep rising [1,2]. Machine learning (ML) has emerged as a key tool for addressing these pressures, with demonstrated gains across the EDA stack in areas ranging from logic synthesis and placement to analog sizing and reliability analysis. This paper critically reviews the application of generative AI to chip design, covering work in analog/mixed-signal (AMS) design, logic design, physical design, and manufacturing reliability, and examines how these methods are evaluated and what gaps remain.

Prior literature reviews have been focused rather than comprehensive. Broad ML-in-EDA surveys document general progress but do not treat generative methods as a primary lens [3,4]. Focused reviews of ML for routing [5] and placement [6] cover those subproblems in depth but do not connect them to upstream design or downstream manufacturability. Surveys of AMS design tools emphasize sizing and modeling but predate or only briefly touch LLM- and agent-based generators [7,8]. Reviews of analog layout tools detail ML-assisted placement and routing but do not address multi-agent LLM workflows or knowledge-graph retrieval [9]. For digital layout generation, deep-learning overviews center on discriminative predictors rather than modern generative loops [10]. Generative surrogates and physics-integrated models are starting to be used for manufacturing analysis including hotspot correction, mask optimization, and reliability prediction—but those recent developments are not included in prior surveys [7,8,10]. In manufacturing, surveys of ML for semiconductor process optimization [11] and

 doi:10.20944/preprints202607.0874.v1

anomaly detection [12] synthesize broad methods, and computational lithography now has community datasets [13], yet a unifying account of generative models across defect modeling, mask optimization, and hotspot correction—and how they connect to earlier design stages—is missing. Recent surveys on LLMs for EDA [14] and circuit foundation models [15] address the rise of large pretrained models but do not span all layers of the design flow or tie coverage to datasets, evaluation practice, and reproducibility.

This review addresses that gap. We survey generative AI methods across all layers of the IC design flow, organizing the field by *what is generated* (circuits, code/flows, layouts, masks, surrogates) and *how generation is grounded* (retrieval/KG, simulators, and physics). We connect AMS, logic, physical design, and yield/reliability, and we highlight where generation must close the loop with analysis and verification. We also surface practical issues such as data curation, benchmarks, and reproducibility that limit deployment in industrial flows.

We claim the following contributions:

- **A unified taxonomy for generative EDA.** We classify model families—LLM agents, graph/sequence generators, diffusion and GAN pipelines, and physics-integrated autoencoders—by their artifacts and feedback signals. This complements LLM-specific and foundation-model surveys by placing them in a full-flow context [14,15].
- **Cross-stage integration.** We map how analog and digital netlists propagate into physical design, and illustrate the bidirectional interplay where EM/stress budget and DFM feedback from the manufacturing stage inform and refine earlier synthesis and layout decisions. Prior reviews treat these areas in isolation; we connect them with concrete interfaces and failure modes [10,13].
- **Evidence and evaluation.** We summarize representative datasets, benchmarks (e.g., LithoBench for computational lithography [13]), and recurring evaluation gaps, especially where cross-paper comparison is weak.
- **Data bottlenecks by stage.** We distinguish between corpus scarcity, label scarcity, proprietary tool-output dependence, and benchmark fragmentation, and show how each bottleneck shapes which generative methods are practical in logic, AMS, physical design, and manufacturing.
- **Guidance for practitioners.** We outline when to favor retrieval-grounded LLM agents versus task-specific generators, where physics priors are essential. We discuss practical data-availability constraints and their implications for model choice. We relate these choices to compute limits and toolchain realities [14].

Scope. We cover (i) AMS topology, sizing, layout, and document-to-netlist extraction; (ii) logic synthesis and flow orchestration; (iii) physical-design scripting, placement, and routability-aware generation; and (iv) manufacturing, yield, and reliability, including hotspot correction, mask optimization, and defect modeling. We emphasize recent work, while retaining influential earlier generative systems where they remain foundational. Representative prior surveys we build upon include Pan *et al.* on LLMs for EDA [14], Fang and Xie *et al.* on circuit foundation models [15], Shi *et al.* on intelligent layout generation [10], and Mina *et al.* and Martins on AMS design automation [7,8].

Why now? Design teams face mounting complexity while tool behavior grows harder to tune by hand. Generative methods can search design spaces that heuristics skip and can tie design choices to downstream manufacturing constraints. In recent years, we have seen increasingly generative applications in IC related paper (Figure 1). Without a cross-layer view, however, teams risk optimizing locally in ways that create problems later in the flow. This survey highlights representative examples and recurring risks [1,2].

Organization. Figure 2 maps the silicon design cycle as a connected system consisting of four main components. Analog IC design and logic synthesis provide the foundational netlists and constraints that dictate the requirements for physical design. This implementation phase yields the GDSII and masks necessary for IC manufacturing and lithography. The framework incorporates feedback loops for reliability corners and DFM/hotspot data, ensuring that downstream manufacturing realities refine upstream design choices and turning a linear sequence into an iterative, cross-layer workflow.

 doi:10.20944/preprints202607.0874.v1

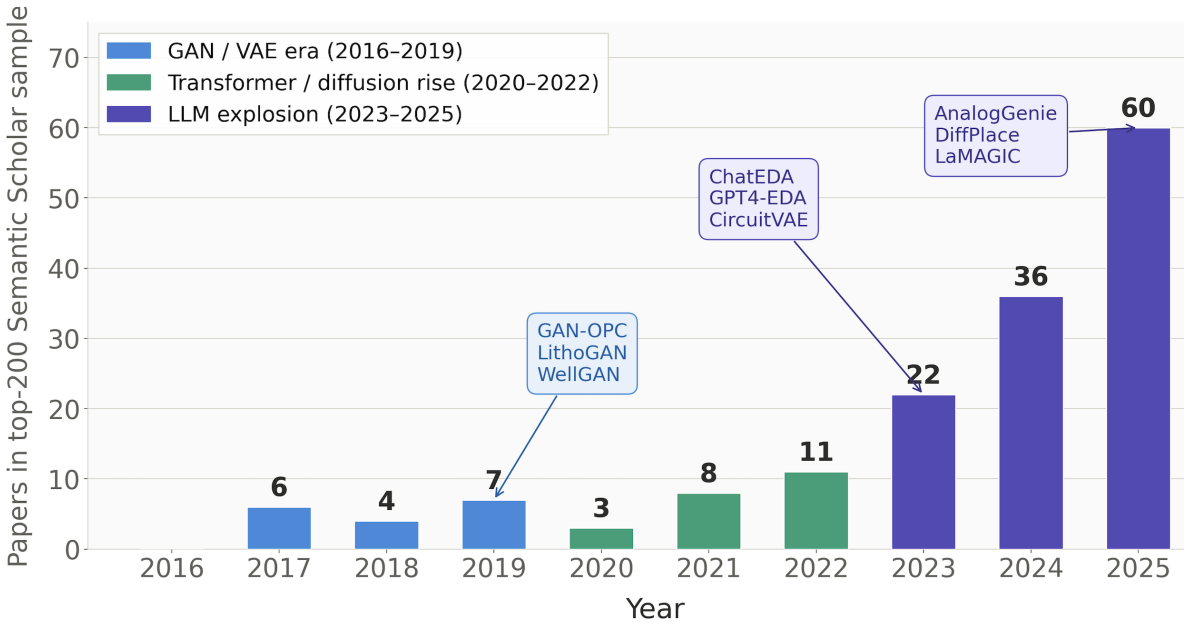

Figure 1. Generative model application in IC design related papers. Data source: Semantic Scholar API — top 200 most relevant results for "generative model integrated circuit design" (query total: 20,778 papers). Year distribution reflects relevance-ranked sample; relative trend is indicative.

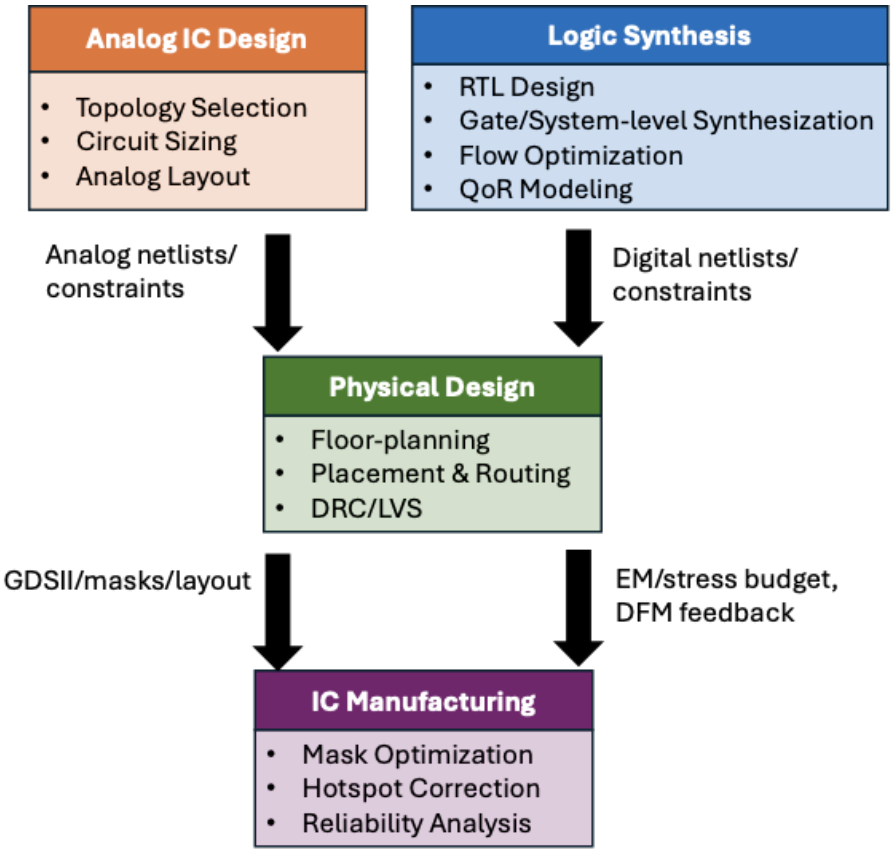

Figure 2. Overview of the integrated circuit design cycle, illustrating the interplay between analog design, logic synthesis, physical design, and the manufacturing process.

The remainder of this article follows the stages of this integrated flow. Section II establishes the taxonomy of generative models and grounding strategies. Section III treats digital logic synthesis and flow optimization. Section IV surveys analog design, from topology synthesis to netlist extraction. Section V covers physical design, focusing on script generation and placement. Section VI exam-

 doi:10.20944/preprints202607.0874.v1

ines manufacturing, yield, and reliability, including computational lithography and physics-aware surrogates. Section VII concludes the article.

2. Background on Generative AI

Generative models aim to approximate the underlying probability distribution $p_{\text{data}}(\mathbf{x})$ of observed data, where $\mathbf{x}$ denotes a data sample, and to synthesize new samples $\tilde{\mathbf{x}} \sim p_\theta(\mathbf{x})$ that preserve the statistical characteristics of the original dataset, where θ denotes the learned model parameters. Over the past decade, their architectures have evolved from latent-representation reconstruction to attention-driven reasoning capable of processing multimodal data, as shown in Figure 3. This section provides an overview of the four generative model families that underlie recent developments in EDA: autoencoder-based models [16], generative adversarial networks (GANs) [17], diffusion models [18], and transformer-based models [19] (including large language models, LLMs). The subsection headers list the year in which each model family was first introduced in the machine learning literature; EDA applications followed in subsequent years.

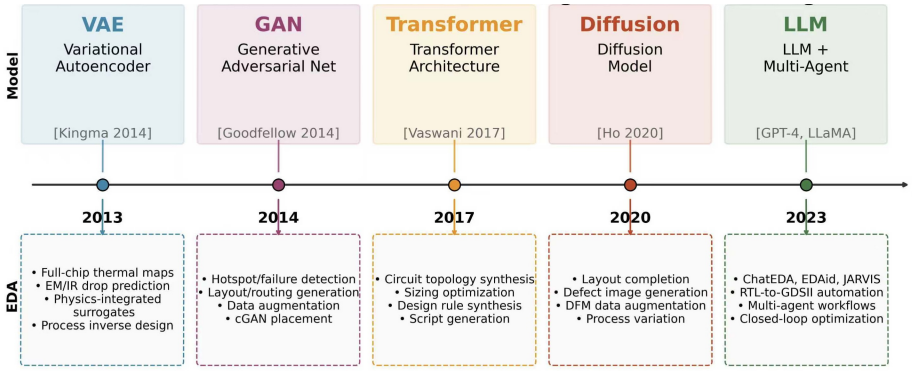

Figure 3. Evolution of Generative AI for IC Design and Manufacturing

2.1. Autoencoder-Based Models (2013–)

Autoencoders learn a low-dimensional representation $\mathbf{z}$ by jointly training an encoder E_ϕ, parameterized by ϕ, and a decoder D_θ, parameterized by θ:

$$\mathbf{z} = E_\phi(\mathbf{x}), \qquad \hat{\mathbf{x}} = D_\theta(\mathbf{z}),$$

with a reconstruction loss such as $\|\mathbf{x} - \hat{\mathbf{x}}\|_2^2$. The variational autoencoder (VAE) formulates this in a probabilistic framework and optimizes the evidence lower bound (ELBO) [16]:

$$\mathcal{L}_{\text{VAE}} = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[-\log p_\theta(\mathbf{x}|\mathbf{z})] + \text{KL}\big(q_\phi(\mathbf{z}|\mathbf{x}) \,\|\, p(\mathbf{z})\big),$$

where the Kullback-Leibler term regularizes the latent space toward a simple prior $p(\mathbf{z}) = \mathcal{N}(0, \mathbf{I})$. Variants such as physics-integrated autoencoders [20] further enhance robustness and interpretability by embedding domain constraints into the learned representations.

In EDA, autoencoder-based models have been applied to several tasks. They learn compact latent representations of chip-level power and thermal maps to accelerate sign-off analysis [21–23]. They also enable inverse design of device and process parameters through exploration of the latent space [24]. In addition, physics relationships can be embedded directly into the decoder so that generated outputs respect known physical laws, producing reliable surrogates for manufacturing and design optimization [20,24].

 doi:10.20944/preprints202607.0874.v1

2.2. Generative Adversarial Networks (2014–)

Generative adversarial networks (GANs) [17] introduce an adversarial optimization between a generator G_θ and a discriminator D_ϕ:

$$\min_\theta \max_\phi \Big[\mathbb{E}_{\mathbf{x}\sim p_{\text{data}}} \log D_\phi(\mathbf{x}) + \mathbb{E}_{\mathbf{z}\sim p(\mathbf{z})} \log\big(1 - D_\phi(G_\theta(\mathbf{z}))\big)\Big].$$

The generator G_θ maps a noise vector $\mathbf{z}$ to a synthetic sample, while the discriminator D_ϕ learns to distinguish real from generated samples. Two widely used variants are relevant to EDA applications. In the conditional GAN (cGAN), auxiliary information $\mathbf{y}$ (e.g., process conditions or design constraints) is incorporated to model $p(\mathbf{x}|\mathbf{y})$, enabling targeted generation for specific design scenarios. The Wasserstein GAN (WGAN) replaces the Jensen-Shannon divergence with the Earth-Mover distance to improve training stability, which is important when training data are scarce or imbalanced, as is common in EDA [17].

GANs are generally more expressive than autoencoders and tend to produce sharper, more realistic outputs because the adversarial discriminator penalizes blurry or averaged generations that reconstruction-based losses permit [17]. In EDA, GANs have been used to model hotspot distributions for lithography analysis [21], to learn failure distributions for reliability modeling [25], to augment datasets for rare-event learning [26,27], and to generate layouts and routing solutions conditioned on timing, power, and area constraints [28–30]. Conditional generation enables what-if exploration in early design stages and increases data diversity for downstream learning tasks.

2.3. Diffusion Models (2020–)

Diffusion models learn to reverse a gradual noising process. The forward process adds Gaussian noise over T steps, producing a sequence of increasingly noisy versions of a data sample $\mathbf{x}_0$. At each step t, the noisy sample $\mathbf{x}_t$ is obtained from the previous step via:

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}\Big(\sqrt{1-\beta_t}\,\mathbf{x}_{t-1},\, \beta_t\mathbf{I}\Big),$$

where β_t is a variance schedule coefficient that controls how much noise is added at step t. A neural network ϵ_θ is trained to predict and remove the noise:

$$\mathcal{L}_{\text{diff}} = \mathbb{E}_{t,\mathbf{x}_0,\epsilon}\Big[\|\epsilon - \epsilon_\theta(\mathbf{x}_t, t)\|_2^2\Big].$$

New samples are generated by starting from pure noise and iteratively applying the learned denoising network, yielding stable and diverse outputs [18]. Extensions such as conditional diffusion, latent diffusion, and classifier-guided diffusion allow generation to be steered toward desired properties.

Because the forward and reverse processes operate on fixed-size inputs, diffusion models are naturally suited to image-like data, and most EDA applications treat layout patterns, lithographic contours, or defect maps as images. In EDA, diffusion models have been applied to layout pattern completion, where a partial routing is extended to a complete solution [10]; to the synthesis of realistic defect images, which augments inspection datasets that are otherwise limited by the rarity of real defects [31]; and to the generation of design-rule-compliant layout patterns for design-for-manufacturability (DFM) analysis [32]. Diffusion models have also been used to emulate process variation and device degradation, providing a way to model physical uncertainties without running full physics-based simulations.

2.4. Transformer-Based Models (2017–)

The Transformer architecture captures long-range dependencies through a self-attention mechanism [19]:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V,$$

 doi:10.20944/preprints202607.0874.v1

where Q, K, and V are query, key, and value matrices formed by learned linear projections of input tokens, and d_k is the dimension of the key vectors. Under the autoregressive objective

$$\mathcal{L}_{\text{Transf}} = -\sum_t \log p_\theta(x_t|x_{<t}),$$

the model learns to predict each token given all preceding tokens, capturing the joint distribution of sequential or graph-structured data. Large language models (LLMs) are transformers trained on large text and code corpora, which gives them broad language understanding, reasoning capability, and the ability to generate and interpret structured formats such as netlists and EDA tool scripts.

Transformer models are currently considered state-of-the-art among generative architectures, though their large parameter counts require correspondingly large training datasets. In EDA, transformers and LLMs have been applied to analog circuit topology selection and sizing [33–35], as well as to design rule synthesis and EDA script generation [36,37]. Their attention mechanism captures hierarchical dependencies among circuit components and design constraints, while the broader language understanding of LLMs enables conversational interfaces, automatic code generation, and cross-tool reasoning.

2.5. Taxonomy of Generative Methods and Data Constraints

To place those model families in the context of the IC design cycle, Table 1 classifies specific generative tools and methods across logic synthesis, AMS design, physical design, and manufacturing. Each entry is characterized by its generation target (what the model produces), its grounding methodology (how generation is constrained or informed), and the feedback signal used to evaluate or refine the output.

While Table 1 organizes generative methods by their outputs, grounding mechanisms, and feedback signals, an equally important dimension is the availability and structure of data that support those methods. In practice, the feasibility of different generative approaches is often constrained not by model architecture, but by the type and quality of data available at each stage of the design flow. In Table 2, we summarized data limitations in generative EDA into four recurring categories: corpus scarcity, label scarcity, proprietary tool-output dependence, and benchmark fragmentation. Those bottlenecks manifest differently across domains. In logic synthesis, the main issue is the limited availability of broad, clean RTL corpora with reliable functional supervision. In AMS, the bottleneck is structured topology and simulation data, which are expensive to curate and often tied to expert design knowledge. In physical design, large datasets exist in principle, but many are locked behind proprietary tools, flows, and rule decks, limiting reproducibility. In manufacturing and reliability, the challenge is often not corpus size but label imbalance and restricted access to rare failure cases.

Those differences are not merely descriptive; they directly determine which generative strategies are practical in each domain. Foundation-style scaling is more viable where corpora are abundant, while retrieval, simulation feedback, and domain-specific models become more effective when data are sparse, structured, or confidential.

With growing interest in applying ML models to IC design in recent years, we have observed an increasing number of dataset-focused papers [38–40] aimed at benchmarking ML performance in this space. Adopting one or more of these datasets as standardized baselines can benefit the community by providing a common basis for comparison across different ML approaches.

In the following sections, we revisit each stage of the design flow—logic synthesis (Section III), AMS design (Section IV), physical design (Section V), and manufacturing and reliability (Section VI)—highlighting how data constraints shape both model design and evaluation practice.

 doi:10.20944/preprints202607.0874.v1

Table 1. Summary of generative AI tools and methods organized by design stage.

Design Stage and Task	Method / Tool	Model and Grounding	Feedback / Signal
Logic Synthesis: RTL Generation and Circuit Construction	ChipChat [41]	LLM + interactive prompting	Verilog code, simulation logs
	DAVE / RTLcoder [42,43]	Fine-tuned LLM	Syntax checks, testbenches
	ShortCircuit [44]	Transformer + MCTS	AIG structure, Boolean equivalence
	T-Net [45]	Triangular network (Shannon decomp.)	Truth tables, logic correctness
	GPT-LS [46]	Decision transformer (offline RL)	Synthesis recipes, QoR metrics
	AISYN [47]	RL-based cut selection	AIG rewriting, area/power
	SOformer [48]	Causal transformer	QoR prediction, trajectory pruning
	Vaegan [26]	VAE-GAN hybrid	Synthetic HLS data, MMD
AMS Design: Topology Selection	AMSnet-KG [49]	LLM + KG-RAG retrieval	Simulation-driven sizing (BO)
	ADO-LLM [33]	LLM-guided Bayesian optimization	Sampling efficiency, FoM
	AnalogCoder [50,51]	LLM and MLLM	SPICE simulation, BO, functionality checks
	LaMAGIC [35]	LLM with supervised fine-tuning	SPICE simulation
	Artisan [34]	LLM with supervised fine-tuning	SPICE simulation + BO
	Atelier [52]	LLM with RAG	SPICE simulation + BO
	AnalogGenie [53]	Domain-specific GPT trained on analog netlists	SPICE simulation
AMS Design: Layout	LayoutCopilot [54]	Multi-agent LLM (script generation)	Layout scripts (SKILL / Python)
	Masala-CHAI [55]	Vision model + LLM	Schematic-to-netlist accuracy
Physical Design: Script Generation	ChatEDA [56]	LLM + LangChain	Syntax correctness, logical accuracy
	EDAid [57]	LLM + multi-agent (RAG / history)	Debugging logs, interaction history
	MCP4EDA [58]	LLM + backend-aware loop	Tool execution logs, PPA metrics
	JARVIS [59]	LLM + synthetic data augmentation	Synthetic training samples
Physical Design: Placement and Routing	Painting on Place [60]	cGAN, image-to-image	Congestion heatmaps, overflow estimates
	Policy-Gradient [61]	RL + DREAMPlace simulator	Total wirelength (HPWL), DRC violations
	GAN-Place [28]	GAN with style transfer	Macro locations, geometric similarity
	PatternPaint [32]	Diffusion model + litho simulation	Lithography hotspots, DRC rules
Manufacturing: Mask Optimization, Hotspot Correction, and Reliability Modeling	LithoGAN [62]	cGAN	Resist contours, EPE metrics
	EM-GAN [25]	GAN	Electromigration stress maps, IR drop
	Physics-VAE [20]	Physics-integrated autoencoder	Breakdown time, lifetime distribution
	CycleGAN-Hotspot [63]	CycleGAN	Hotspot images, wafer yield
	GridVAE [64]	VAE	IR drop
	GAN-SRAF [65]	GAN	Assist feature placement on layout
	GAN-OPC [66]	GAN	Corrected mask, simulated wafer image

 doi:10.20944/preprints202607.0874.v1

Table 2. Data bottlenecks across generative EDA domains and their implications for model design.

Domain	Data Availability	Primary Data Bottlenecks	Model Implications
Logic Synthesis	Moderate	Public RTL datasets are small and narrow in design style, making it difficult to train models that generalize across specification types. Functional correctness labels require testbenches that are expensive to write and rarely open-sourced. Evaluation benchmarks vary across works, preventing direct comparison of reported results.	Models rely on self-generated or synthetic supervision, which limits functional correctness guarantees. Generalization across design families and specification formats is poor. Reported accuracy figures are sensitive to benchmark choice and are difficult to reproduce.
AMS Design	Low	Structured datasets of analog circuit topologies are small and expensive to curate, as each entry requires expert annotation and time-consuming SPICE simulation. Process-specific design knowledge is largely proprietary and not shared across institutions or companies.	Models depend heavily on retrieval-augmented generation (RAG) or domain-specific fine-tuning to compensate for limited training data. Foundation-model scaling is impractical without much larger open datasets. Human expert involvement remains necessary throughout the design loop.
Physical Design	Moderate	Many large-scale placement and routing datasets are generated by proprietary EDA tools under non-disclosure agreements, restricting public access. Open benchmarks cover only a subset of real industrial flows, and the labels (e.g., congestion maps, timing reports) are tool-specific and do not transfer across toolchains.	Models trained on open datasets transfer poorly to industrial toolchains. Academic metrics such as half-perimeter wirelength (HPWL) do not always correlate with real PPA outcomes. Results reported on open benchmarks may not reflect performance on proprietary designs.
Manufacturing, Yield, and Reliability	Low	Defect and failure events are rare in production, so labeled inspection datasets are heavily imbalanced: non-defective samples vastly outnumber defective ones. Fab-internal process data and failure signatures are proprietary and not released publicly. The diversity of failure modes across process nodes and design styles is rarely captured in available datasets.	Severe class imbalance limits the ability of models to detect and characterize rare but critical failure modes. Data augmentation and physics-based simulation are commonly used to compensate, but augmented data may not capture the full range of real failure signatures. Physics-informed or hybrid models are often necessary to produce outputs that are physically meaningful under process variation.

3. Logic Synthesis

Logic synthesis is among the most mature and automated stages of the digital design flow. Modern tools efficiently transform RTL into optimized gate-level implementations thanks to decades of progress in Boolean optimization, technology mapping, and heuristic-driven search. Compared with placement, routing, or verification, the baseline level of automation in logic synthesis is significantly higher. However, recent advances in generative AI—particularly large language models (LLMs), graph transformers, and reinforcement learning—enable forms of automation that extend well beyond traditional synthesis capabilities. These models aim to infer RTL directly from natural language, construct circuits from functional specifications, learn high-quality synthesis recipes from data, and accelerate solver-based optimization. While early demonstrations show substantial potential, the practical integration of generative models into production flows remains an active research challenge.

3.1. RTL and High-Level Code Generation

LLMs have been widely studied for translating natural language or structured specifications into synthesizable RTL. Early works, such as DAVE [42], demonstrated natural-language-to-Verilog conversion, followed by a range of fine-tuned or prompt-engineered models including [43,67–73].

 doi:10.20944/preprints202607.0874.v1

These efforts explore hierarchical prompting, multi-stage refinement, and domain-specific fine-tuning to improve correctness.

Conversational design frameworks such as ChipChat [41] demonstrate the feasibility of co-designing real processors using interactive prompting. Figure 4 details the complete flow of ChipChat, a seminal contribution to LLM-assisted hardware design. In ChipChat, an engineer provides natural language specifications and constraints to GPT-4, which generates Verilog and a testbench. A compilation and simulation tool then returns structured feedback at one of four levels—no-failing-nodes (NFN), testbench fail (TF), simulation hard fail (SHF), or always-hard fail (AHF)—and the loop iterates until the design passes. As a case study, the framework was used to co-design QTCore, an 8-bit accumulator microprocessor with a 256-instruction ISA, which was successfully taped out in Skywater 130 nm CMOS. This represents the first reported tapeout of an AI-written hardware design. A benchmark study across eight Verilog tasks showed that GPT-4 performs best among the models tested, GPT-3.5 remains usable, and Bard and HuggingChat prove unreliable. A key finding is that LLMs are most effective when used conversationally with human-provided testbenches and verification feedback, rather than in a one-shot generation mode. ChatEDA [56] takes a complementary approach by attempting to drive a full RTL-to-GDSII flow from natural-language queries, grounding a fine-tuned LLaMA 2 model in EDA tool API documentation to decompose user intent into executable Tcl scripts. Despite promising results, both works reveal the need for stronger functional guarantees and broader generalization across design families.

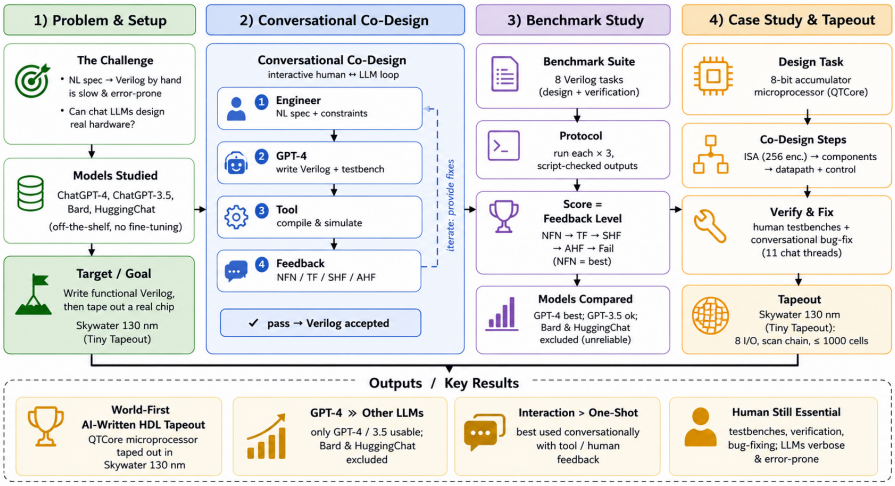

Figure 4. Overview of the ChipChat conversational co-design framework for LLM-assisted RTL generation [41]. An engineer provides a natural language specification and constraints to GPT-4, which writes Verilog and a testbench. A compilation and simulation tool returns one of four feedback levels—no-failing-nodes (NFN), testbench fail (TF), simulation hard fail (SHF), or always-hard fail (AHF)—and the loop iterates until the design passes. The framework is evaluated on a benchmark suite of eight Verilog tasks run three times each with script-checked outputs, where GPT-4 achieves the best results and GPT-3.5 remains usable, while Bard and HuggingChat prove unreliable. As a case study, the framework is used to co-design QTCore, an 8-bit accumulator microprocessor with a 256-instruction ISA, successfully taped out in Skywater 130 nm CMOS—representing the first reported tapeout of an AI-written hardware design. The study concludes that LLMs are most effective when used conversationally with human-provided testbenches and verification, rather than in a one-shot generation mode.

3.2. Generative Circuit Construction

Beyond code generation, several works aim to directly construct gate-level circuits. GAN-based structures [29] produced varied but often non-functional topologies, motivating more structured approaches. T-Net [45] performs end-to-end synthesis from truth tables using a triangular network informed by Shannon decomposition, achieving high correctness on circuits up to 1200 nodes. ShortCircuit [44] frames synthesis as sequential And-Inverter Graph (AIG) construction using a structurally

 doi:10.20944/preprints202607.0874.v1

aware transformer refined through Monte Carlo Tree Search. These approaches illustrate that generative models can learn structural biases intrinsic to the logic networks they target.

While these systems produce promising results on curated benchmarks, scalability to multi-output functions and industrial designs remains an active research direction. LLM-based RTL generators treat synthesis as a language-to-code translation problem, where the main challenge is semantic correctness and alignment with specifications. By contrast, methods such as T-Net and ShortCircuit treat synthesis as structured circuit construction; validity is easier to enforce but scalability becomes the dominant bottleneck, because these models must search an exponentially large space of valid circuit structures and have so far been demonstrated only on circuits with a limited number of inputs and outputs. This distinction matters because the two types of models fail in different ways: language models tend to generate syntactically plausible but functionally incomplete RTL, whereas graph-based models preserve structural validity more naturally but remain confined to smaller or more stylized benchmarks. A useful takeaway is that these methods should not be compared purely by headline correctness numbers; they solve different subproblems under different assumptions.

3.3. Synthesis Flow Optimization

Synthesis flow optimization refers to the automated selection and sequencing of logic optimization passes—such as technology mapping, rewriting, and refactoring—to maximize quality of results (QoR) metrics including area, delay, and power. GPT-LS [46] uses an offline reinforcement learning (RL) Decision Transformer trained on OpenABC-D trajectories, achieving QoR competitive with Bayesian optimization while reducing sequence generation time. AISYN [47] embeds RL-based cut selection within AIG rewriting to target area and power improvements, demonstrating substantial gains on ISCAS'85 circuits and modest improvements on industrial designs.

These works highlight the potential of data-driven policy learning to complement or replace hand-crafted heuristics for synthesis flow optimization.

3.4. Predictive Modeling and Solver Acceleration

Predictive models aim to accelerate synthesis by forecasting QoR outcomes or improving solver efficiency. SOformer [48] employs a causal transformer with cross-attention between circuit graphs and optimization sequences to predict synthesis trajectories, enabling early pruning of low-quality flows. Transformer-based MaxSAT solvers [74] represent conjunctive normal form (CNF) formulas as bipartite graphs and use a graph attention mechanism with multi-relational (meta-path) edges [74] to achieve significant speedups over classical SAT/MaxSAT engines.

Such models illustrate how generative and predictive architectures can augment existing optimization engines and reduce computational overhead across flow iterations.

3.5. System-Level Generative Synthesis

Using GPT-4, the authors of [75] demonstrated an end-to-end generative design of a vector processor SoC, encompassing RTL, firmware, and verification infrastructure; the design was fabricated successfully in 130-nm CMOS. Although a compelling proof of concept, context-length and modularity constraints currently limit generalization beyond designs of this scale and regularity.

Generative approaches have also been explored for data augmentation and design-space exploration. Vaegan [26] produces high-fidelity HLS data using a VAE–GAN hybrid, reducing Maximum Mean Discrepancy by 44% compared to Gaussian or ABC-based generators. Phybrid [76] employs transfer learning to predict ASIC metrics from FPGA prototypes, achieving an average error of 2.6%. These works highlight how generative models can mitigate data scarcity in architecture- and HLS-level exploration.

3.6. Limitations of Current Models and Resource Requirements

Despite rapid progress, substantial challenges remain before generative models can be deployed reliably in industrial logic synthesis flows. Current LLM-based RTL generators often exhibit perfor-

 doi:10.20944/preprints202607.0874.v1

mance degradation as design complexity increases. Generative circuit construction models such as T-Net and ShortCircuit show promising results on small or medium-scale benchmarks but struggle to scale to larger multi-output functions because the search space grows exponentially with the number of inputs and outputs; training these models also requires considerable GPU resources.

A second challenge concerns data scarcity and limited access to high-quality training data. Public RTL datasets are small, narrow in domain coverage, and often contain inconsistencies. As a result, many works including [42,43,67,69,71,73] assembled their own datasets, which were typically limited in circuit variety and design style, restricting the generalization of the trained models to circuit families not well represented in those datasets. Proprietary datasets, such as those used in ChipNeMo [77], demonstrate the benefits of domain specialization but are not publicly accessible, limiting reproducibility and community-wide progress.

A third challenge is the lack of common evaluation methods across works. Current studies use disparate metrics ranging from Likert-scale human assessments [77] to syntax checks and partial functional testbenches [67,73]. Because researchers are not using a common set of evaluation methods, the results reported across works cannot be fairly compared. The absence of shared benchmarks for RTL generation, circuit construction, and synthesis flow optimization makes it difficult to measure progress or identify which methods represent genuine advances.

Finally, the heavy dependence on proprietary large models and the prohibitive cost of high-quality fine-tuning pose practical barriers. GPT-4 [78] and related closed models continue to outperform existing open-source alternatives, yet their weights are inaccessible for domain-specific refinement. Conversely, strong open-source results such as those achieved by ChipNeMo rely on massive compute budgets—on the order of 128 A100 GPUs—which are far beyond the reach of most academic labs and smaller companies.

A further limitation is that most logic-generation studies are evaluated in isolation from downstream physical design. An RTL or synthesis recipe that appears beneficial under local QoR metrics may still lead to poor floorplanning, congestion, or timing closure later in the flow. This disconnect is especially problematic for LLM-based generators, whose outputs may preserve functional intent while introducing structural patterns that are difficult to route or optimize physically. A more realistic evaluation framework would track whether improvements at the logic stage are preserved through placement, routing, and signoff.

Collectively, these limitations highlight that, although generative AI has introduced new opportunities for logic synthesis, significant challenges related to scalability, dataset quality, evaluation standards, and computational resources must be addressed before such methods can be integrated into industrial design flows.

4. Analog/Mixed-Signal Circuit Design

In contrast to digital design, which has seen maturation of automated design flows, analog/mixed-signal (AMS) circuit design still relies on significant manual effort. Designers iteratively refine and tune their circuits, a process that is not only time-consuming but also dependent on expert intuition, prior experience, and a deep understanding of performance trade-offs when selecting topologies.

Researchers have applied generative AI and machine learning to AMS design with the aim of reducing manual design effort and improving scalability. There are four primary applications of generative AI to analog/mixed-signal design: sizing, topology synthesis, layout, and schematic and document parsing. Analog sizing refers to the optimization of design variables, such as MOSFET channel widths and resistance values, to meet a set of performance specifications. Topology synthesis refers to the selection or generation of a circuit topology. Layout refers to the physical arrangement of circuit components and their connections, performed after topology synthesis and sizing. Schematic and document parsing uses generative models to extract circuit descriptions, component values, and performance specifications from written documents, including data sheets, application notes, and journal papers. Beyond these categories, recent work has also explored the use of generative models

 doi:10.20944/preprints202607.0874.v1

for behavioral modeling [79–81] and testbench generation [82]. However, the use of generative AI for these purposes remains relatively limited; therefore, we do not discuss them in detail in this survey.

This section surveys the current uses of generative AI in topology synthesis, sizing, layout, and schematic and document parsing, while also discussing current limitations, open issues, and future trends.

4.1. AMS Topology Synthesis

Circuit synthesis is among the biggest hurdles to AMS design automation due to its complex design space containing numerous discrete components of different types and a strong reliance on human expertise. Recent advances in generative AI have opened promising pathways toward automating this process by enabling models to learn high-level structural patterns and generate valid circuit topologies directly from high-level specifications. Broadly, research efforts in generative AI for AMS topology synthesis fall into two categories: (1) use of widely available pre-trained models such as large language models (LLMs), and (2) development of custom generative models such as generative adversarial networks, variational autoencoders, and domain-specific pre-trained models.

4.1.1. LLM-Based Topology Synthesis

Recent progress in LLM-assisted EDA has revealed a key bottleneck for AMS circuits: the lack of high-quality, structured topology data to support model reasoning and prevent hallucination. Shi et al. address this issue with AMSnet-KG, a curated dataset linking AMS netlists to a knowledge graph that encodes architectural semantics, pin functions, and expert annotations [49]. Global annotations guide topology selection, while local annotations mark components that must be treated as matched pairs during sizing, as well as the net connections subject to matching constraints. AMSnet-KG integrates LLMs through Knowledge-Graph Retrieval-Augmented Generation (KG-RAG) to enable automated AMS circuit generation. Given performance specifications, the LLM proposes a high-level design strategy, which is used to retrieve matched components and testbenches from AMSnet-KG. The system then assembles a candidate topology and performs simulation-driven sizing using Bayesian optimization. If specifications are unmet or the estimated number and size of transistors exceeds a user-specified threshold, the framework iteratively refines and regenerates the topology, closing the loop with minimal human intervention.

Rather than taking a RAG approach, the developers of LaMAGIC opted to fine-tune an LLM using a custom dataset of power converter circuits [35]. The objective of LaMAGIC is to generate power converter topologies in a single forward pass. To improve generation quality, LaMAGIC introduces a structured prompt format referred to as the canonical form. These enhancements enable LaMAGIC to generate correct topologies with a 96% success rate, outperforming prior reinforcement learning-based methods.

Similarly, Artisan employs a purpose-built dataset of three-stage operational amplifier topologies to fine-tune a 7B-parameter LLM [34]. In addition to supervised fine-tuning, Artisan incorporates a custom chain-of-thought and tree-of-thought prompting scheme for three-stage amplifier topology generation. It also introduces a textual circuit representation called NetlistTuple, which converts the conventional graph-based circuit description into a text format more suitable for LLM processing. A key limitation of fine-tuning, as demonstrated in both Artisan and LaMAGIC, is the substantial computational cost required for training. Fine-tuning also produces models that are narrow in scope: a model trained on three-stage amplifiers cannot readily be applied to other circuit families without retraining on a new dataset.

Atelier also targets the synthesis of three-stage operational amplifiers but takes a different approach to address the data problem [52]. Rather than fine-tuning, the developers of Atelier constructed a custom database of circuit topologies extracted from published circuit papers and data sheets using LLMs, and integrated it with RAG and open-source LLMs. Atelier further employs a multi-agent, feedback-based design flow with graph-of-thought prompting, achieving better performance in three-stage amplifier topology generation than Artisan. The comparison between Artisan and Atelier

 doi:10.20944/preprints202607.0874.v1

suggests that RAG-based approaches can match or exceed the accuracy of fine-tuned models on specialized tasks while avoiding the high training cost and narrow generalization of fine-tuning. However, RAG-based methods depend on the quality and coverage of the retrieval database, and their performance may degrade on circuit families that are not well represented in the database. AnalogCoder [50] and AnalogCoderPro [51] take a different direction from all of the previous methods. Instead of building a knowledge database or fine-tuning a model, these tools reframe topology selection as a Python code generation problem. This strategy leverages the LLM's code-generation capability while avoiding the time-consuming database construction required by RAG-based methods and the computational cost of fine-tuning. A feedback loop informed by circuit simulation is used to verify that the generated topologies are functional. Figure 5 presents the full AnalogCoderPro framework, which extends this by using a multimodal LLM that can accept simulation waveform images as input, feeding visual simulation results back into the multi-agent design loop [51].

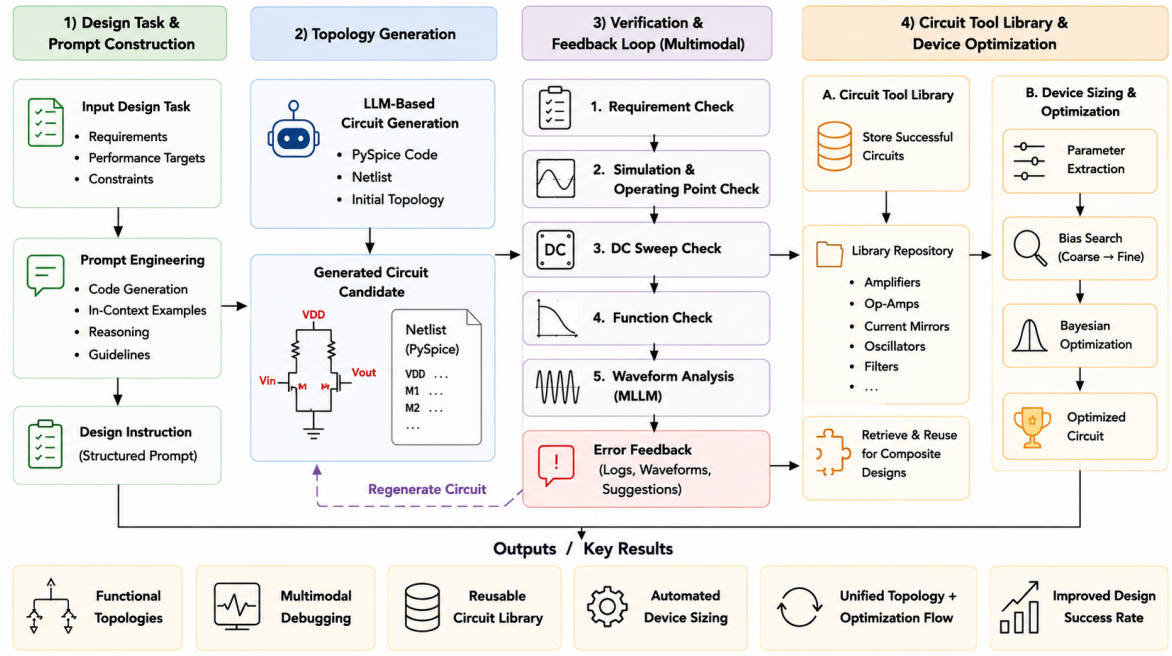

Figure 5. Overview of the AnalogCoderPro framework for LLM-driven analog circuit topology generation and sizing [51]. The framework handles two task types: circuit design tasks, which require component-level netlist generation from scratch, and composite design tasks, which can leverage a library of pre-generated subcircuits. For both task types, a multimodal LLM generates a Python-based circuit description (PySpice code) and iterates through design prompt generation and subcircuit retrieval as needed. The generated netlist is then validated through simulation-based rule checks, including netlist, operating point, DC sweep, and functional checks. A key distinguishing feature of AnalogCoderPro is its multimodal simulation feedback loop: a vision-capable LLM analyzes simulation waveforms and generates text-based feedback, which is fed back into the topology generation stage to guide regeneration. Once a valid topology is confirmed, component sizing is performed in three steps: tunable design parameters and their search ranges are extracted, bias voltages are found through a two-level coarse-to-fine sweep, and device sizes are optimized using Bayesian optimization for a target figure of merit.

Collectively, these LLM-based approaches illustrate a trade-off between breadth and specialization in AMS topology synthesis. Methods such as AnalogCoder and AMSnet-KG support a wide variety of circuit types, but often struggle to produce highly optimized circuits. In contrast, Artisan, Atelier, and LaMAGIC focus on narrow circuit classes such as three-stage operational amplifiers or power converters, achieving higher success rates within those domains but lacking generality. Because these two groups of methods are evaluated on different tasks and by different metrics, their reported results are not directly comparable. What is clear is that both directions are being actively explored; what remains unknown is which approach will ultimately yield better results as circuit complexity increases and evaluation methods become more standardized.

 doi:10.20944/preprints202607.0874.v1

4.1.2. Custom Generative Models

In contrast to approaches that rely on widely available LLMs, the works discussed in this section employ custom deep-generative models. The first class integrates generative models within the optimization loop, as in [83] and [84], to generate amplifier topologies from specifications. The second class uses generative models to directly synthesize topologies.

The work in [83] generates amplifier topologies using a graph variational autoencoder (G-VAE) combined with a Bayesian optimizer. Since many Bayesian optimization algorithms are designed for continuous design spaces, the authors first learn a G-VAE that embeds the discrete, graph-encoded circuit description of a three-stage amplifier into a continuous latent space. After optimizing the topology, they map the result to a transistor-level implementation using a library of building blocks and the g_m/I_D methodology. Similarly, ATOM [84] represents the three-stage amplifier as a directed acyclic graph (DAG) and uses a DAG-VAE to embed the graph into latent space. By combining this embedding with an improved freeze-thaw Bayesian optimization algorithm, ATOM achieves higher success rates while requiring up to 8.5$\times$ fewer simulations than prior state-of-the-art methods.

AnalogGenie [53] takes a more general approach by training a domain-specific GPT model on a dataset of over 3,000 analog circuit topologies. It introduces a novel circuit representation based on an Eulerian circuit traversal and a custom tokenizer designed for circuit netlists. After fine-tuning, AnalogGenie produces valid topologies for 93.2% of generation attempts across a diverse range of circuit types.

In summary, the G-VAE and ATOM approaches demonstrate the effectiveness of embedding generative models within optimization frameworks for topology generation, but remain tailored to specific circuit types. AnalogGenie represents a step toward more general and scalable topology generation by training on a broader and more diverse dataset.

4.2. AMS Circuit Sizing

Machine learning-based tools have been widely used for analog circuit sizing—optimizing design variables to meet specific performance requirements. Prior approaches using black-box optimization techniques such as Bayesian optimization (BO), reinforcement learning, and deep learning often lack awareness of circuit topology or domain knowledge. The papers in this section address this limitation by incorporating LLMs into the optimization process to inject circuit-specific knowledge.

ADO-LLM explores the integration of LLMs with BO to improve sampling efficiency in analog circuit design [33]. The approach uses the in-context learning ability of LLMs to inject domain knowledge into the BO process, refining candidate samples generated by the optimizer. The method produces high-quality design points efficiently across two different types of analog circuits, achieving improvements in both design efficiency and solution quality. However, the framework is constrained by its reliance on external LLM APIs and the inherent curse of dimensionality faced by BO in high-dimensional design spaces.

LEDRO introduces a different strategy by combining LLMs with BO for design space reduction rather than direct point selection [85]. The framework employs a 70B-parameter LLM to identify the regions of the design space most likely to contain high-performance solutions, thereby guiding subsequent BO searches. The approach achieves superior figure of merit (FoM) performance and requires fewer iterations to converge for both simple and complex op-amp designs.

EasySize adopts a different strategy compared to LEDRO and ADO-LLM [86]. Instead of narrowing the design space, it uses an LLM to determine the weighting of performance metrics within an FoM-based optimizer. The method first ranks performance metrics based on how easily their requirements can be met. A set of design specifications is then generated, and corresponding loss functions are created with weightings derived from ease of attainability and human feedback. This dataset is used to fine-tune the LLM, which then generates optimized FoM weights for the sizing optimizer. The authors report superior sizing performance compared to prior BO-based methods

 doi:10.20944/preprints202607.0874.v1

on a set of op-amp benchmarks; however, comparisons to other state-of-the-art sizing methods and evaluations on more complex circuit types have not yet been reported.

Despite these promising results, the generalizability of LLM-guided sizing remains unclear, and further comparison with other state-of-the-art methods across a broader range of circuit types is needed. An interesting future direction is the integration of sizing feedback into the topology generation tools discussed in the previous section, which would help close the loop between topology selection and parameter optimization.

4.3. Analog Layout

Analog layout design remains a largely manual process due to complex constraints on symmetry, matching, and parasitic sensitivity that strongly influence post-layout performance. Automating this process is challenging, as the quality of results from tools like MAGICAL or ALIGN depends heavily on parameter tuning and the correct definition of layout constraints. Even interactive layout environments suffer from similar limitations, as user-defined constraints and parameter choices can significantly affect layout quality. Recent research has begun exploring how machine learning and LLMs can enhance analog layout automation and interactivity by improving constraint specification and designer–tool collaboration.

Chen et al. integrate LLMs with Bayesian optimization to address the problem of net weighting constraints in the MAGICAL layout tool [87]. In MAGICAL, net weights control the priority assigned to different signal nets during placement and routing; setting these weights correctly is critical to achieving good layout quality but requires expert knowledge. Their method uses the in-context learning ability of an LLM to perform key BO steps, including initializing design points, modeling the design space, and selecting new candidate points. The approach achieves performance comparable to state-of-the-art BO algorithms while enabling more effective design space exploration. However, repeated LLM inference introduces significant computational overhead, raising concerns about scalability to larger designs.

LayoutCopilot takes a complementary approach by improving the usability of interactive layout tools rather than pursuing full automation [54]. It generates layout scripts from high-level natural language commands, handling tasks such as constraint generation and parameter adjustment for device placements. LayoutCopilot employs a multi-agent system that decomposes user commands into executable layout scripts, with specialized agents handling subtasks such as code generation, task decomposition, and abstraction classification.

Other generative approaches target specific layout subproblems. Wang et al. propose a generative learning framework for analog building-block layout that leverages prior design data [88]. Layouts from a design repository are used to train a variational autoencoder (VAE) that captures designer layout preferences across multiple block types. The trained model then generates routing guidance for new layouts, improving circuit performance by 1.2% in gain and 2.1% in bandwidth compared to traditional migration-based methods.

Analog layout optimization also requires joint consideration of N-well/P-well region generation and device placement to prevent area inefficiency, parasitic coupling, and the Well Proximity Effect (WPE). Traditional placers treat wells as post-processing constraints, but Zhu et al. propose a well-aware placement approach that integrates well generation directly into placement by modeling wells as fence-region constraints [89]. A conditional GAN generates well guidance masks from initial placements, steering the placer and enforcing the WPE constraint during legalization to maintain compact layouts and low half-perimeter wirelength (HPWL). The method achieves up to 82% area reduction and 46% HPWL reduction over baselines. Remaining challenges include scalability to larger circuits, adaptability across process nodes, and dependence on manually optimized training data.

4.4. Schematic and Document Parsing

Recent work in analog design automation has emphasized the importance of scalable, structured data generation to support data-driven design methods. Much of the most valuable information about

 doi:10.20944/preprints202607.0874.v1

AMS circuits has been presented in the form of schematic images, which are not suitable inputs for many automation tools. Similarly, performance specifications and design parameters are often buried in written documents such as data sheets and journal papers, making them difficult to ingest into automated design flows.

Masala-CHAI addresses the schematic parsing problem through the creation of a large-scale dataset that bridges schematic images and circuit-level representations via netlists [55]. By combining a YOLOv8-based vision pipeline with an LLM, the tool automatically identifies circuit components in schematic diagrams and translates them into SPICE netlists. It employs advanced prompt tuning to enhance netlist extraction accuracy, thereby reducing component classification errors, incorrect node assignments, and missing elements. Note that AMSnet-KG similarly automates the process of deriving netlists from schematic images; however, a different vision pipeline was used to identify and label components in the schematic images [49].

A complementary tool, DocEDA, translates human-authored documents, such as conference papers and data sheets, into formalized specifications suitable for automated design workflows [90]. DocEDA combines computer vision with chain-of-thought LLM prompting to parse figures, tables, and text and extract circuit specifications such as gain, bandwidth, and supply voltage. This capability supports dataset expansion and improves interoperability between written design documents and machine learning-based analog design tools.

4.5. Trends, Limitations, and Future Directions

Recent progress in generative AI for analog circuit design highlights promising directions, but several key challenges remain. One limitation is the lack of methods that close the loop between topology, sizing, and layout. In practice, layout failures are often discovered late in the design flow, and current tools do not provide mechanisms to feed layout outcomes back into topology synthesis or sizing decisions. Similarly, few works utilize feedback between sizing and topology synthesis.

Another barrier is the scarcity of high-quality training data. Constructing circuit datasets remains labor-intensive, requiring extensive domain expertise, and few efforts have focused on systematically open-sourcing such resources. Broader community-driven initiatives in data collection and sharing could accelerate progress, lower the entry barrier for new researchers, and promote reproducibility.

The field also lacks widely adopted open benchmarks, making it difficult to compare results across works. Many current methods are evaluated on narrow circuit classes or relatively simple design problems, making it hard to assess scalability to more complex scenarios. Establishing standardized benchmarks would enable more rigorous evaluation of generative models in AMS design. Some initial steps have been taken: AnalogGym provides an open sizing benchmark based on open-source PDKs [91], and AMSnet-KG has been partially open-sourced [49].

Finally, robustness to process, voltage, and temperature (PVT) variation and device mismatch remains largely unaddressed. Most existing methods focus on nominal design conditions, leaving a gap in ensuring reliable performance under real-world variability. Addressing this limitation is essential for transitioning these techniques into practical design automation tools.

5. Physical Design

In this section, we focus on the domain of IC Physical Design. Recent advances in Generative AI have also led to a number of innovative studies in this field. We categorize its major applications into two main areas.

5.0.1. Tool Script Generation

Generative models have achieved notable breakthroughs in recent years. Large Language Models (LLMs) are rapidly transforming their ability to comprehend tool documentation and autonomously generate scripts into substantial productivity improvements in EDA tool automation.

 doi:10.20944/preprints202607.0874.v1

5.0.2. Placement and Routing

Generative models are increasingly employed to capture spatial and topological correlations in physical layouts. Approaches such as conditional GANs, diffusion models, and reinforcement–generative hybrids have been leveraged to enhance placement quality, predict routing congestion, and generate DRC-compliant layout patterns, demonstrating the capability of generative modeling to explore diverse, physically consistent design solutions beyond traditional heuristic optimization.

5.1. Tool Script Generation

One of the key challenges in Physical Design (PD) is the efficient utilization of Electronic Design Automation (EDA) tools, which often requires engineers to interpret and translate complex user guides, command references, and tool manuals into executable scripts. Leveraging their strong language comprehension and code-generation capabilities, Large Language Models (LLMs) have recently demonstrated remarkable effectiveness in this area—particularly in automating and optimizing EDA scripting workflows.

5.1.1. First Stand alone

Early progress was made by Bei Yu et al., who introduced ChatEDA [56] at MLCAD 2023. Figure 6 illustrates the complete ChatEDA framework for LLM-driven RTL-to-GDSII automation. ChatEDA employed LLaMA 2 as the base model and applied LoRA fine-tuning to adapt it to OpenROAD [92] commands and APIs. The OpenROAD ecosystem itself integrates several key open-source components—including Yosys [93] for RTL synthesis and OpenLane [94] for full RTL-to-GDSII flow automation—providing a complete, open testing environment for LLM-driven physical-design automation. By fine-tuning on domain-specific scripting data, ChatEDA aligned the generative capabilities of the model with the structured syntax and hierarchical logic of EDA commands. This made LLaMA 2 particularly suitable for the task: its large contextual window and natural-language reasoning ability allowed the model to interpret high-level design intents and translate them into executable Tcl scripts, effectively bridging the gap between user intent and tool operation.

5.1.2. Multi-agent collaboration

Building upon this foundation, the same research group proposed EDAid [57], a multi-agent collaborative framework that integrates several domain-specialized LLM agents. Each agent focuses on a different stage of the EDA flow—such as synthesis setup, placement control, or verification script generation—and cooperates through message passing and task decomposition. This multi-agent architecture fits the problem of multi-stage flow orchestration naturally, as it mirrors how human experts coordinate across specialized tool domains. By distributing reasoning responsibilities among specialized LLMs, EDAid improved both the reliability and contextual accuracy of generated scripts, achieving higher success rates in complex hierarchical flows.

5.1.3. Close-loop feedback

More recently, MCP4EDA [58] further advanced this paradigm by introducing result-aware automation and parameter optimization. Unlike earlier models that generated scripts in a static, one-shot manner, MCP4EDA established a closed-loop feedback mechanism between the LLM and the EDA toolchain. It incorporated real backend feedback—such as timing, area, and congestion metrics—into iterative synthesis optimization, allowing the model to refine its script generation based on physical outcomes. This framework employed Retrieval-Augmented Generation (RAG) to ground the model in tool documentation and synthesis heuristics, while leveraging its generative capabilities to explore new combinations of optimization parameters. The integration of retrieval grounding with backend feedback made the model both contextually aware and physically meaningful, achieving measurable improvements in timing and area across multiple RTL-to-GDSII benchmarks.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 13 July 2026 doi:10.20944/preprints202607.0874.v1

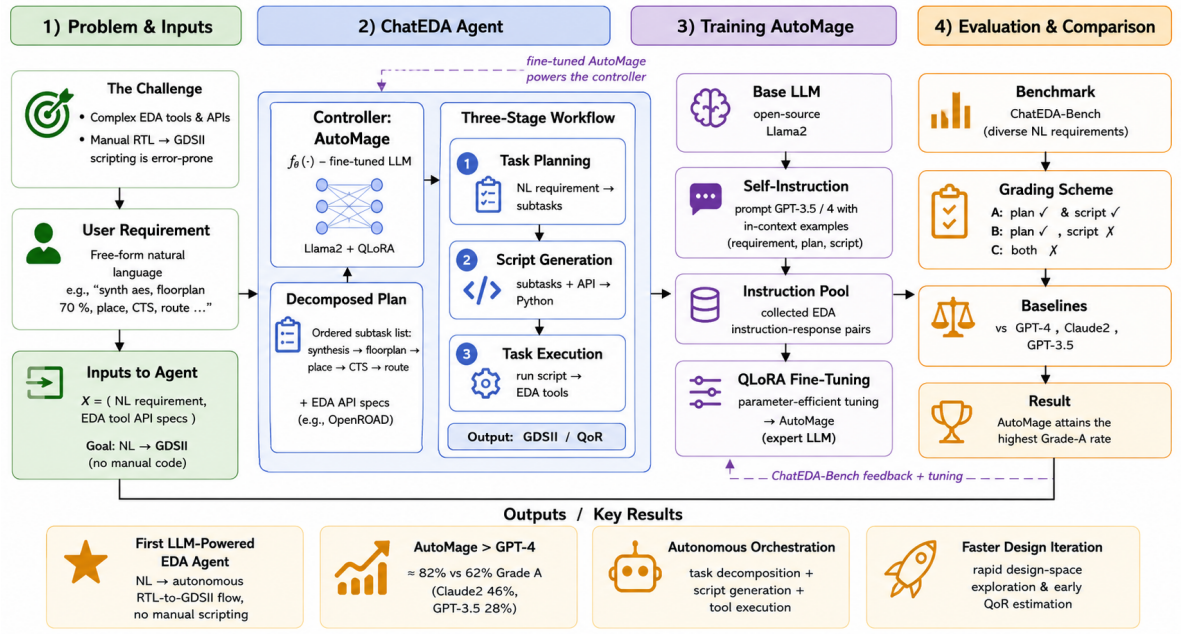

Figure 6. Overview of the ChatEDA framework for LLM-driven RTL-to-GDSII automation [56]. Given a free-form natural language requirement and EDA tool API specifications as inputs, the fine-tuned LLM controller AutoMage decomposes the requirement into an ordered subtask plan and executes a three-stage workflow: task planning (natural language to subtasks), script generation (subtasks and API calls to Python), and task execution (running scripts through EDA tools such as OpenROAD). AutoMage is trained via self-instruction, in which GPT-3.5/4 generates requirement–plan–script triplets that are used to fine-tune LLaMA 2 with QLoRA. The framework is evaluated on ChatEDA-Bench using a three-level grading scheme (Grade A: correct plan and script; Grade B: correct plan, incorrect script; Grade C: both incorrect), where AutoMage achieves approximately 82% Grade-A accuracy, outperforming GPT-4 (62%), Claude 2 (46%), and GPT-3.5 (28%).

5.1.4. Data augmentation

While these academic contributions demonstrate the feasibility and promise of LLM-based EDA automation, they are largely constrained by their reliance on open-source tools and datasets. This dependence limits their evaluation scope and generalizability to proprietary industrial environments, where tool ecosystems, parameter spaces, and design rules are significantly more complex. To tackle this, NVIDIA's research team introduced JARVIS [59] in 2025—a multi-agent code assistant designed for high-quality EDA script generation. JARVIS advances the field by employing Synthetic Data Generation (SDG) to augment limited training data and performing both Domain-Adaptive Pre-Training (DAPT) and Domain-Supervised Fine-Tuning (DSFT) to enhance the model's understanding of specialized API structures. It further integrates the strengths of prior academic frameworks—multi-agent collaboration and feedback-guided correction—within an industrial-grade setting, achieving more robust and verifiable script generation. Through the combination of data synthesis, retrieval-based grounding, and structured multi-agent reasoning, JARVIS represents a pivotal step toward scalable, enterprise-level deployment of generative EDA assistants.

Collectively, these works address the challenge of automating EDA tool control by employing generative LLM architectures that are structurally aligned with the problem itself: their text-to-code reasoning enables high-level design specifications to be translated into executable flow scripts, while their generative adaptability supports iterative optimization and tool parameter exploration. This evolution—from static script generation to context-aware, feedback-driven automation—marks a significant step toward intelligent and adaptive physical-design flows.

5.2. Placement and Routing

Within the overall Integrated Circuit (IC) Physical Design flow, generative models have indeed demonstrated remarkable potential in tool control and script generation. However, due to the in-

 doi:10.20944/preprints202607.0874.v1

herently diverse nature of tasks across different stages, the impact of generative modeling remains concentrated in only a subset of domains—most notably in standard cell library generation, floorplanning, placement, and routing. These stages naturally involve design-space exploration and multi-solution search, where the capacity of generative models to synthesize diverse and physically consistent alternatives becomes particularly advantageous.

5.2.1. Congestion awareness Placement

One of the earliest applications of generative modeling in physical design appeared in Painting on Placement [60], which addressed the problem of routing congestion prediction. Traditional analytical or regression-based congestion estimators require handcrafted features and lack spatial generalization. To overcome this, the authors reformulated congestion estimation as a conditional image-to-image generation task and employed a conditional Generative Adversarial Network (cGAN) to translate placement maps into predicted congestion heatmaps. This model fits the problem naturally, since both placement layouts and congestion maps are spatially structured data. By learning a direct mapping from spatial cell-density distributions to congestion patterns, the conditional GAN provided a fast, differentiable, and layout-aware alternative to classical estimators, significantly accelerating early-stage routability analysis.

5.2.2. Routability awareness Placement

Building on this concept of generative spatial prediction, Cheng et al. proposed the Policy-Gradient Placement and Generative Routing framework [61], which jointly tackled macro placement and global routing. They introduced a hybrid architecture that combined a policy-gradient reinforcement learning (RL) network for macro placement with a conditional generative routing model that produced one-shot routing paths. The RL component was used to sequentially decide macro positions under multi-objective physical constraints, while the generative routing model directly synthesized valid interconnect paths conditioned on pin locations and net topology. This design tightly couples the model structure with the physical problem: the sequential decision nature of placement aligns with reinforcement learning, while the spatially structured routing generation benefits from the conditional generative approach. Such synergy enabled improved wirelength, routability, and congestion balance compared with traditional heuristic or sequential routing methods.

5.2.3. Style Transfer for Layout Distribution

More recently, GAN-Place [28] extended adversarial learning to the standard-cell placement problem. Rather than generating new layouts from scratch, GAN-Place adopted a style-transfer–based adversarial framework to inject the spatial distribution characteristics of high-quality placements into existing open-source placers. Here, the generator learns the implicit "style" or statistical distribution of expert-designed layouts, while the discriminator enforces structural plausibility and density constraints. This adversarial setup fits the nature of placement optimization, where the goal is not purely reconstruction but distribution matching between generated and optimal placements. By aligning the generative model's output distribution with high-performance design samples, GAN-Place demonstrated consistent improvements in placement quality and PPA (Power–Performance–Area) metrics.

5.2.4. DRC Rule Compliance Placement and Data Augmentation

At the pattern and manufacturability level, PatternPaint [32] addressed the problem of layout pattern augmentation for Design for Manufacturability (DFM) and lithography hotspot analysis. The authors used a combination of inpainting and diffusion-based generative models to synthesize new, rule-compliant layout patterns that expand the coverage of existing pattern libraries. Diffusion models were particularly suitable for this task, as their iterative denoising process allows the generation of fine-grained geometric details while preserving DRC-compliant spatial relationships. This work demonstrated that generative modeling can complement traditional pattern-matching or rule-based

 doi:10.20944/preprints202607.0874.v1

verification methods by producing realistic, manufacturable variants that enrich training datasets for DFM and lithography analysis.

In contrast, stages such as clock tree synthesis (CTS) [95], signoff analysis, power integrity evaluation, and DRC/LVS verification have seen relatively limited adoption of generative modeling. These tasks are inherently deterministic and constraint-driven, focusing on producing a single correct and convergent solution that satisfies strict physical and electrical rules. Consequently, rule-based and discriminative learning approaches remain dominant in these domains, as they emphasize prediction accuracy and constraint satisfaction over design-space exploration. Thus, while generative frameworks excel in creative, exploratory, and data-scarce tasks within the PD flow, deterministic models continue to serve as the backbone for analysis-oriented stages such as timing closure and physical signoff.

5.3. Trends, limitations, and future directions

While generative modeling has shown considerable promise across multiple domains of Physical Design, significant challenges remain before these methods can achieve practical and generalizable solutions. In the area of EDA tool script generation, recent works have demonstrated impressive progress in automating tool operation and parameter tuning. However, most academic efforts still rely heavily on open-source tools and datasets, which inevitably limits their transferability and compatibility with proprietary industrial flows. Moreover, the absence of a standardized, third-party benchmark for script generation and tool control makes it difficult to perform fair and reproducible comparisons—many studies report strong results under inconsistent or domain-specific evaluation settings. The establishment of open, unified benchmarks and metrics for evaluating generative EDA assistants will be essential to move beyond proof-of-concept results toward systematic performance assessment.

Within the Physical Design workflow, generative models have proven particularly effective in placement and routing, where design-space exploration and distribution learning are central to the problem structure. Nevertheless, the direction of modern EDA development is increasingly toward cross-stage integration, in which early steps such as logic synthesis and floorplanning are executed with full awareness of downstream objectives including timing closure, power optimization, and signal integrity. This trend amplifies the computational and modeling burdens of early PD stages, as these tasks must implicitly account for complex multi-physics and cross-layer dependencies. In this context, generative AI stands out as a key enabler: by learning comprehensive, multi-domain representations and fine-tuning on high-quality, cross-stage datasets, generative models can serve as powerful surrogates and design-space explorers that capture inter-stage interactions more effectively. Such integration has the potential to not only accelerate early-stage decision-making but also propagate improvements downstream, ultimately enhancing the overall efficiency and intelligence of the physical design flow.

The same argument applies downstream. Placement and routing are often evaluated using HPWL, congestion, or timing-oriented proxies, yet these objectives do not fully capture manufacturability risk. A placement policy that improves routability may still increase hotspot density or complicate mask correction. This suggests that cross-stage integration in generative PD should not stop at logic-aware optimization, but should also incorporate lithography and reliability feedback earlier than is common today.

6. IC Yield and Reliability

Generative AI has emerged as a valuable tool in yield and reliability analysis, addressing persistent challenges in defect detection, reliability prediction, and lithography optimization. These three areas are traditionally among the most data- and compute-intensive tasks in semiconductor manufacturing. At their core, they share common bottlenecks: (i) labeled data are scarce and highly imbalanced, particularly for rare but critical defect and failure modes; (ii) physical parameters such as electric fields, stress maps, or resist contours must satisfy underlying physics constraints to be meaningful; and (iii) physics-based simulation flows, while accurate, are prohibitively slow for industrial-scale

 doi:10.20944/preprints202607.0874.v1

deployment. Generative frameworks are typically employed to model the distributions of these key physical parameters—such as voltage drop maps, defect patterns, or lithographic contours—and then integrate these with domain equations or heuristic models for downstream analysis. This dual role of generative modeling, as both data augmenter and physics surrogate, explains much of its growing appeal in IC yield and reliability research.

6.1. Data Augmentation

One of the most universal challenges across IC design and manufacturing is the lack of labeled data. In wafer inspection and defect classification, images of defective wafers form only a tiny fraction of all collected inspection data, leading to heavily skewed datasets. Traditional augmentation strategies—rotations, flips, random crops—expand dataset size but do little to diversify underlying defect signatures. Generative AI provides richer augmentation by synthesizing entirely new images of wafer defect patterns. In [96] and [97], VAEs were trained to generate synthetic wafer defect images. VAEs learn a continuous latent representation of the defect distribution and sample from it to produce new examples; this makes them well suited to generating diverse variations of rare defect types, but the reconstruction-based training objective can lead to blurry outputs that may not capture fine-grained defect morphology. By augmenting training data with these synthetic samples, the authors achieved significant gains in fault classification accuracy, with improvements of up to 40% in some defect categories [97].

In contrast to VAE-based approaches, GAN-based models produce more visually sharp and realistic defect images because the adversarial discriminator explicitly penalizes outputs that do not resemble real defects. However, GANs can be harder to train and are more prone to mode collapse, in which the generator produces only a limited variety of outputs. For wafer inspection tasks, GAN-based models were shown to generate realistic defect maps that strengthened the robustness of downstream classifiers, especially against rare defect types [98].

The challenge of small-sample learning is equally pressing in analog and mixed-signal circuits, where circuit faults may only be observable under certain test conditions. Jia *et al.* combined Deep Convolutional GANs (DCGANs) with Transformer architectures to generate time-frequency fault signatures, enabling accurate detection of soft faults despite limited labeled data [99]. These works collectively show that generative AI can function as a "data equalizer," mitigating class imbalance and enabling the use of conventional classifiers in otherwise data-sparse regimes.

Beyond defect detection, post-silicon reliability analysis also benefits from generative augmentation. Carbunescu-Stoenescu *et al.* used VAEs to reconstruct multivariate parameter distributions from small post-silicon datasets, supporting accurate yield estimation despite data scarcity [100]. Yan *et al.* further advanced this approach by integrating semantic autoencoders with a physics-based percolation model of gate oxide breakdown, allowing dielectric thickness distributions to be inferred directly from measured dielectric lifetime distributions [24]. Figure 7 provides a detailed overview of the semantic autoencoder framework for dielectric lifetime modeling. This physics-informed generative approach not only produced more accurate lifetime distribution estimates but also improved interpretability by aligning predictions with the physics of dielectric wearout. Through a physics-embedded decoder, it can generate trustworthy synthetic data to support large-scale reliability analysis. Follow-up work refined the loss functions used in training, tightening prediction intervals and producing physically meaningful parameter distributions [101]. These examples underscore how generative autoencoders provide an alternative to traditional statistical fitting and Monte Carlo simulation for yield and lifetime estimation.

 doi:10.20944/preprints202607.0874.v1

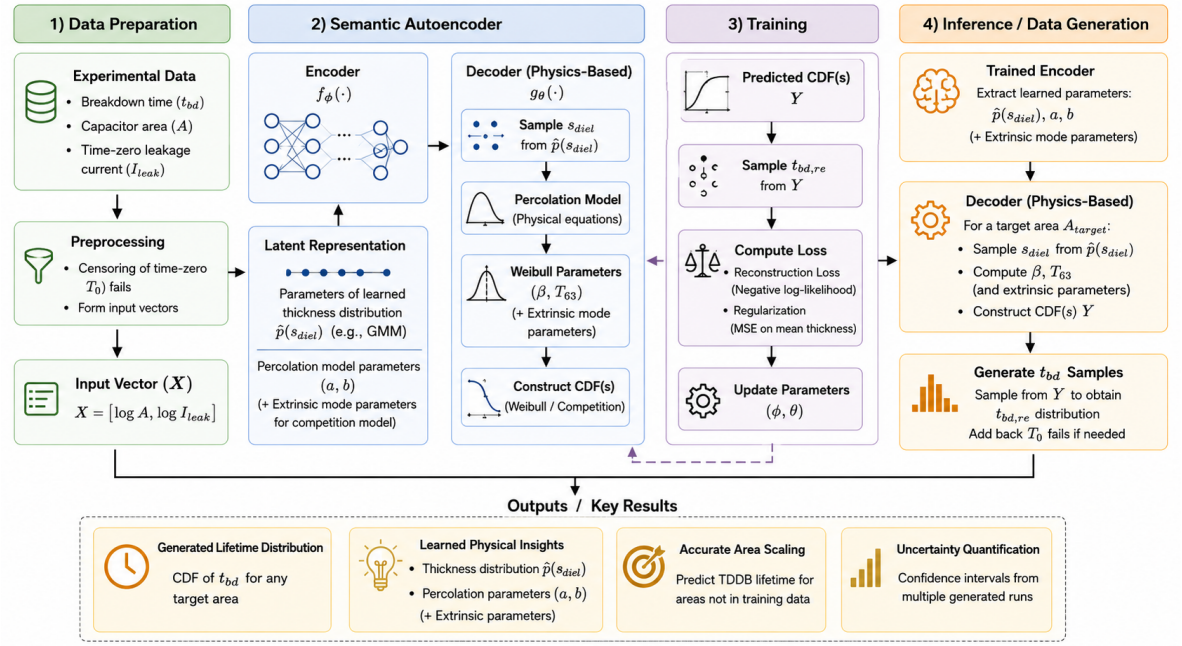

Figure 7. Overview of the semantic autoencoder framework for dielectric lifetime modeling [24]. The encoder $f_\phi(\cdot)$ maps measured breakdown data—breakdown time t_{bd}, capacitor area A, and time-zero leakage current I_{leak}—to a latent representation of the dielectric thickness distribution $\hat{p}(s_{\text{diel}})$ and percolation model parameters a and b. The physics-based decoder $g_\theta(\cdot)$ samples from this distribution, applies a percolation model to compute Weibull parameters β and T_{63}, and constructs the lifetime cumulative distribution function (CDF). Training minimizes a combination of reconstruction loss and a regularization term on mean thickness. Once trained, the framework generates lifetime distributions for target capacitor areas not present in the training data, enabling area scaling, uncertainty quantification, and large-scale reliability analysis without additional silicon measurements.

6.2. Defect Localization and Spatio-Temporal Modeling

While dataset augmentation mitigates data scarcity, accurately modeling the spatial location and temporal evolution of defects and mechanical stresses remains a critical challenge that often exceeds the capabilities of conventional rule-based analysis. Generative methods increasingly move beyond binary "defective vs. non-defective" classification toward pixel-level defect segmentation—the task of identifying the precise location and extent of each defect within an inspection image—and temporal stress prediction. In inspection tasks, SEM-based imaging is complicated by noise and the high cost of annotation. Diffusion models have been employed to reconstruct "defect-free" baseline images and then isolate defects as residuals, enabling pixel-level segmentation under weak supervision [31]. Knowledge-distilled variants of these diffusion models, in which a smaller student model is trained to mimic the outputs of the full diffusion model, further reduce inference times by one to two orders of magnitude [31], a key step toward industrial feasibility.

In the context of defect localization, multimodal integration serves as a key enabler for correlating spatial information from different inspection sources. Adversarial learning has been leveraged to achieve such integration: Kim *et al.* used conditional GANs to combine SEM and layout information, allowing systematic defects—linked to design patterns—to be distinguished from random particle-induced defects [102]. Such integration reflects a trend where generative AI is no longer an add-on but a core part of the inspection pipeline.

Generative AI for spatio-temporal modeling is equally important in reliability. Traditional finite-element solvers for electromigration (EM) and time-dependent dielectric breakdown (TDDB) require hours or days to simulate stress evolution across interconnects. GAN-based frameworks such as EM-GAN recast this as an image-to-image translation problem: given interconnect topologies and current density maps, the model generates stress distributions at different stages of aging [25]. This approach provides large speedups while retaining fidelity to stress physics. Because the GAN is

 doi:10.20944/preprints202607.0874.v1

trained to produce outputs that match the spatial structure of finite-element solutions, it functions as a generative surrogate: unlike a regression model that produces a single scalar output, the GAN produces a full two-dimensional stress map that can be used in the same way as a simulation result. Zhou *et al.* extended the concept with GridNet, which not only predicts EM-induced IR drop but also computes sensitivity maps for localized engineering-change-order fixes, embedding generative prediction directly into the design-optimization loop [103].

GridVAE [64], a VAE-based surrogate model, improved prediction stability and reduced error by nearly 40% compared to prior methods, while achieving over 100× acceleration in full-chip EM-aware IR-drop analysis. Similar principles have been applied to dielectric breakdown modeling, where Lamichhane *et al.* developed a GAN-based electrostatic solver that accelerates TDDB field calculations by 200× compared with finite-element methods, maintaining near-physical accuracy [104]. This trend also extends to coupled power–thermal settings [105,106]. These works illustrate how generative models function not only as data augmenters but also as computational surrogates, replacing or accelerating traditional solvers while producing outputs that respect underlying physical constraints.

6.3. Mask Optimization and Lithography Modeling

Lithography represents one of the most computationally demanding stages of the IC design flow, where small improvements in mask optimization or hotspot correction translate into large yield gains. Generative models have been widely adopted as surrogates to replace costly optical simulations. Sim *et al.* pioneered the use of CycleGAN for hotspot correction, formulating the task as unpaired image-to-image translation: hotspot patterns are directly transformed into corrected non-hotspot patterns, eliminating the need for paired training data and reducing correction times from days to hours [63]. Yang *et al.* proposed GAN-OPC, which combines GANs with inverse lithography pretraining to produce masks that are already near-optimal and require minimal refinement [66].

Similarly, Alawieh *et al.* introduced GAN-SRAF, which formulates subresolution assist feature (SRAF) insertion as a translation problem and employs novel heatmap encodings to achieve over 100× runtime reduction compared with model-based methods [65]. Ye *et al.* further developed LithoGAN, an end-to-end conditional GAN that maps masks directly to resist contours, bypassing both optical simulation and contour extraction. LithoGAN achieved accelerations of ∼1800× relative to rigorous solvers [62], highlighting the potential of generative AI for lithography. These works demonstrate how generative methods compress traditionally slow lithography flows into real-time or near-real-time pipelines, enabling their integration into design-stage verification and correction.

6.4. Generative and Physics-Aware Modeling for Packaging Yield and Reliability

Packaging-aware yield and reliability are emerging as central challenges as chiplet-based systems shift critical failure modes beyond the monolithic die into the package, interposer, and die-to-die (D2D) interconnect. The standardization of UCIe has accelerated this transition by enabling package-level interoperability while pushing toward higher data rates (48/64 GT/s), 3D integration, and enhanced reliability features, thereby increasing sensitivity to routing topology, bump architecture, power delivery, and thermal gradients [107,108]. In this context, heterogeneous-integration roadmaps increasingly emphasize packaging as a multiphysics co-design problem, where signal integrity (SI), power integrity (PI), thermal effects, and thermo-mechanical reliability must be addressed jointly rather than in isolation [109,110].

These trends align naturally with the rise of physics-aware surrogate and generative modeling frameworks. Early efforts in packaging have demonstrated the value of fast, parameterized models that bridge traditionally disjoint design stages. For example, optimization of decoupling-capacitor placement across hierarchical PDNs enables tighter coupling between package design and voltage-noise constraints [111], while surrogate-based approaches to interposer design facilitate efficient exploration of routing, shielding, and SI tradeoffs under process variation [112]. Similarly, temperature-dependent RLGC/SPICE representations provide a pathway to unify electromagnetic channel extraction with circuit-level evaluation for chiplet-package co-design, and recent work incorporating temperature,

 doi:10.20944/preprints202607.0874.v1

process variation, and yield into UCIe channel design highlights the strong coupling between operating conditions and reliability-optimal design points [113]. Collectively, these approaches point toward the need for compact, physics-grounded models that can support rapid design-space exploration under realistic operating constraints.

Recent literature further reinforces this direction while moving toward more scalable, data-driven paradigms. Techniques such as SPIRAL+ demonstrate that high-speed inter-chiplet validation increasingly requires joint SI/PI analysis rather than isolated channel evaluation [114], while learning-based approaches such as GNN-SP enable fast S-parameter estimation for UCIe-style interconnects, making early-stage exploration tractable [115]. At the system level, multi-fidelity thermal modeling frameworks and electrothermal co-optimization techniques highlight the importance of capturing hotspot formation, PDN behavior, and reliability-sensitive gradients throughout the design process, rather than only at signoff [116].

Taken together, these developments position advanced packaging as a natural next frontier for the physics-aware generative frameworks discussed in this review. In the near term, surrogate and hybrid models can act as fast digital twins for SI/PI, thermal, and yield closure across the package-design space. Looking forward, physics-integrated generative models offer the potential to directly synthesize interposer routing, shielding structures, bump and ground patterns, and PDN decoupling strategies under explicit reliability constraints, extending prior work on physics-aware learning and generative modeling toward full-stack package co-design [117].

6.5. Trends, Limitations, and Future Directions

Each model family discussed in this section addresses a distinct aspect of the yield and reliability problem. Variational autoencoders (VAEs) are well suited to distribution learning under data scarcity because their probabilistic latent space allows new samples to be drawn from the learned distribution even when training data are limited. GANs are effective for generating realistic defect maps and spatio-temporal stress distributions because the adversarial training objective directly encourages outputs that are indistinguishable from real physical data. Diffusion models have demonstrated strong performance for pixel-level defect segmentation because their iterative denoising process preserves fine spatial detail better than reconstruction-based methods. Taken together, these complementary strengths suggest that the field is moving from isolated data augmentation toward tightly integrated, physics-aware workflows in which different model families are combined to address different parts of the problem.

Nevertheless, limitations remain. Many published works continue to rely on VAEs and GANs, which were introduced more than a decade ago. The continued dominance of these models is not inherently a problem—they remain well matched to the image-structured data common in yield and reliability tasks—but it does mean that more recent architectural advances have not yet been fully exploited. Although recent studies have begun exploring the use of large language models [118], LLMs are primarily designed for text and code and there is no strong reason to expect them to outperform image-based generative models on tasks such as defect inspection or lithography correction that are fundamentally spatial in nature. Diffusion models, on the other hand, are a more natural candidate for improvement in these domains: their iterative denoising process can generate high-resolution spatial outputs with fine-grained detail, and conditional variants can be guided by layout geometry or process parameters to produce physically plausible results. Early evidence from defect segmentation tasks supports this expectation [31].

A second limitation is that most frameworks assume the accuracy of the underlying physics equations they interface with. In practice, process physics models are incomplete or approximate, which constrains the reliability of generated outputs. One promising direction is physics-integrated generative modeling, where neural networks are trained jointly with or in correction of physics-based equations, as demonstrated in physics-integrated VAEs [20]. Emerging models such as Kolmogorov–Arnold Networks (KANs) [119] further expand this space by learning functional relationships directly from data in cases where no closed-form equation is available.

 doi:10.20944/preprints202607.0874.v1

Finally, data efficiency remains underexplored. Active learning methods, which adaptively select the most informative samples for labeling or simulation, could substantially reduce the amount of training data required, especially in domains where silicon measurement campaigns are costly. Yet few studies have incorporated active learning [120,121] into generative AI for yield and reliability. Addressing these gaps—adopting more capable generative architectures where appropriate, embedding physics priors more rigorously, and integrating data-efficient strategies—will be critical for next-generation yield and reliability tools.

In summary, the integration of generative AI into IC yield and reliability analysis reflects a broader shift: from ad hoc data augmentation to physics-informed, computationally efficient frameworks that model the full distribution of physical outcomes. As methods mature, the potential exists to scale these approaches to full-chip inspection, cross-node generalization, and hybrid physics–AI simulations, embedding generative models as standard components of semiconductor design and manufacturing workflows.

7. Conclusions

This survey examined generative AI tools across the full silicon lifecycle. We organized recent work by what is generated—circuits, scripts, layouts, masks, surrogates—and by how generation is grounded in retrieval, simulators, and physics, linking AMS, logic, physical design, and manufacturing to datasets, evaluation practice, and deployment constraints. Three lessons stand out: generation must close the loop with analysis and verification or it drifts; grounding matters—use retrieval and tool context when semantics dominate, use physics priors when geometry and reliability rule; progress depends on shared artifacts—open interfaces, curated datasets with clear provenance, and benchmarks that align design and fab views. For practitioners, the path is pragmatic: start with narrow, high-value tasks, prefer small domain-tuned agents near the tools they drive, add physics where failure costs are high, and measure with published protocols.

Looking ahead, we expect tighter coupling of LLM agents with simulators and sign-off, standard schemas that carry intent from topology to masks, and benchmarks that score end-to-end quality, not only local gains. Smaller models that can be deployed locally within a design team's own infrastructure, with well-documented training data and methodology, will speed adoption so that generation, grounding, and governance act together to shorten design cycles and raise yield.

Acknowledgments: This material is based upon work supported by the National Science Foundation the Center for Advanced Electronics in Machine Learning (CAEML) under Grant No. CNS 2137288, CNS 2137283, CNS 2137255 – and its industry members. The authors acknowledge the limited use of generative AI tools (ChatGPT and Claude) for language polishing and figure preparation. All content was verified and edited by the authors, who assume full responsibility for the published work.

References

1. Foster, H. 2024 Wilson Research Group IC/ASIC Functional Verification Trend Report. White paper, Siemens EDA, 2025. Accessed 28-Oct-2025.
2. IEEE. International Roadmap for Devices and Systems (IRDS) 2022 Edition. Technical report, 2022.
3. Huang, G.; Hu, J.; He, Y.; Liu, J.; Ma, M.; Shen, Z.; Wu, J.; Xu, Y.; Zhang, H.; Zhong, K.; et al. Machine Learning for Electronic Design Automation: A Survey. *ACM Transactions on Design Automation of Electronic Systems (TODAES)* **2021**, *26*, 40:1–40:46. https://doi.org/10.1145/3451179.
4. Yu, B. Machine Learning in EDA: When and How. In Proceedings of the 2023 ACM/IEEE 5th Workshop on Machine Learning for CAD (MLCAD), 2023, pp. 1–6. https://doi.org/10.1109/MLCAD58807.2023.10299822.
5. Li, L.; Cai, Y.; Zhou, Q. A Survey on Machine Learning-Based Routing for VLSI Physical Design. *Integration, the VLSI Journal* **2022**, *86*, 51–56. https://doi.org/10.1016/j.vlsi.2022.05.003.
6. Qiu, Y.; Xing, Y.; Zheng, X.; Gao, P.; Cai, S.; Xiong, X. Progress of Placement Optimization for Accelerating VLSI Physical Design. *Electronics* **2023**, *12*, 337. https://doi.org/10.3390/electronics12020337.
7. Mina, R.; Jabbour, C.; Sakr, G.E. A Review of Machine Learning Techniques in Analog Integrated Circuit Design Automation. *Electronics* **2022**, *11*, 435. https://doi.org/10.3390/electronics11030435.

 doi:10.20944/preprints202607.0874.v1

8. Martins, R.M.F. A Survey of Machine and Deep Learning Techniques in Analog Integrated Circuit Layout Synthesis. *Microelectronics* **2025**, *1*, 2. https://doi.org/10.3390/microelectronics1010002.
9. Chen, L.; Chen, Y.; Chu, Z.; Fang, W.; Ho, T.Y.; Huang, R.; Huang, Y.; Khan, S.; Li, M.; Li, X.; et al. Large circuit models: opportunities and challenges. *Science China Information Sciences* **2024**, *67*. https://doi.org/10.1007/s11432-024-4155-7.
10. Shi, Y.; Shang, M.; Qi, Z. Intelligent Layout Generation Based on Deep Generative Models: A Comprehensive Survey. *Information Fusion* **2023**, *100*, 101940. https://doi.org/10.1016/j.inffus.2023.101940.
11. Chen, Y.L.; Sacchi, S.; Dey, B.; Blanco, V.; Halder, S.; Leray, P.; De Gendt, S. Exploring Machine Learning for Semiconductor Process Optimization: A Systematic Review. *IEEE Transactions on Artificial Intelligence* **2024**. https://doi.org/10.1109/TAI.2024.3429479.
12. Li, Z.; Jiang, Y.; Liu, J.; Xing, J. A Survey of Deep Learning for Industrial Visual Anomaly Detection. *Artificial Intelligence Review* **2025**, *58*, 279. https://doi.org/10.1007/s10462-025-11287-7.
13. Zheng, S.; Yang, H.; Zhu, B.; Yu, B.; Wong, M.D.F. LithoBench: Benchmarking AI Computational Lithography for Semiconductor Manufacturing. In Proceedings of the NeurIPS 2023 Datasets and Benchmarks Track, 2023.
14. Pan, J.; Zhou, G.; Chang, C.C.; Jacobson, I.; Hu, J.; Chen, Y. A Survey of Research in Large Language Models for Electronic Design Automation. *ACM Computing Surveys* **2025**. Also available as arXiv:2501.09655, https://doi.org/10.1145/3715324.
15. Fang, W.; Wang, J.; Lu, Y.; Liu, S.; Wu, Y.; Ma, Y.; Xie, Z. A Survey of Circuit Foundation Model: Foundation AI Models for VLSI Circuit Design and EDA. arXiv preprint arXiv:2504.03711, 2025.
16. Kingma, D.P.; Welling, M. Auto-Encoding Variational Bayes. In Proceedings of the 2014 International Conference on Learning Representations (ICLR), 2014, Vol. abs/1312.6114.
17. Goodfellow, I.J.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; Bengio, Y. Generative Adversarial Nets. In Proceedings of the Advances in Neural Information Processing Systems. Curran Associates, Inc., 2014, Vol. 27.
18. Ho, J.; Jain, A.; Abbeel, P. Denoising diffusion probabilistic models. In Proceedings of the Proceedings of the 34th International Conference on Neural Information Processing Systems, Red Hook, NY, USA, 2020; NIPS '20.
19. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, L.; Polosukhin, I. Attention is all you need. In Proceedings of the Proceedings of the 31st International Conference on Neural Information Processing Systems, Red Hook, NY, USA, 2017; NIPS'17, p. 6000–6010.
20. Takeishi, N.; Kalousis, A. Physics-Integrated Variational Autoencoders for Robust and Interpretable Generative Modeling. In Proceedings of the Advances in Neural Information Processing Systems; Ranzato, M.; Beygelzimer, A.; Dauphin, Y.; Liang, P.; Vaughan, J.W., Eds. Curran Associates, Inc., 2021, Vol. 34, pp. 14809–14821.
21. Jin, W.; Sadiqbatcha, S.I.; Zhang, J.; Tan, S.X.D. Full-Chip Thermal Map Estimation for Commercial Multi-Core CPUs with Generative Adversarial Learning. In Proceedings of the Proc. IEEE/ACM Int. Conf. on Computer-Aided Design (ICCAD), 2020, pp. 1–9.
22. Chen, L.; Jin, W.; Tan, S.X.D. Fast Thermal Analysis for Chiplet Design Based on Graph Convolution Networks. In Proceedings of the Proc. Asia and South Pacific Design Automation Conf. (ASP-DAC), 2022, pp. 485–492.
23. Chhabria, V.A.; Ahuja, V.; Prabhu, A.; Patil, N.; Jain, P.; Sapatnekar, S.S. Thermal and IR Drop Analysis Using Convolutional Encoder-Decoder Networks. In Proceedings of the ASP-DAC, 2021, pp. 690–696. https://doi.org/10.1145/3394885.3431583.
24. Yan, W.; Wu, E.; Schwing, A.G.; Rosenbaum, E. Semantic Autoencoder for Modeling BEOL and MOL Dielectric Lifetime Distributions. In Proceedings of the 2023 IEEE International Reliability Physics Symposium (IRPS), 2023, pp. 1–9. https://doi.org/10.1109/IRPS48203.2023.10117878.
25. Jin, W.; Sadiqbatcha, S.; Sun, Z.; Zhou, H.; Tan, S.X.D. EM-GAN: Data-Driven Fast Stress Analysis for Multi-Segment Interconnects. In Proceedings of the 2020 IEEE 38th International Conference on Computer Design (ICCD), 2020, pp. 296–303. https://doi.org/10.1109/ICCD50377.2020.00057.
26. Liao, Y.; Adegbija, T.; Lysecky, R.; Tandon, R. Skip the Benchmark: Generating System-Level High-Level Synthesis Data using Generative Machine Learning. In Proceedings of the Proceedings of the Great Lakes Symposium on VLSI 2024. ACM, 2024, GLSVLSI '24, p. 170–176. https://doi.org/10.1145/3649476.3658738.

 doi:10.20944/preprints202607.0874.v1

27. Chhabria, V.A.; Kunal, K.; Zabihi, M.; Sapatnekar, S.S. BeGAN: Power Grid Benchmark Generation Using a Process-Portable GAN-Based Methodology. In Proceedings of the Proc. IEEE/ACM Int. Conf. on Computer-Aided Design (ICCAD), 2021, pp. 1–8.
28. Lu, Y.C.; Ren, H.; Hsiao, H.H.; Lim, S.K. GAN-Place: Advancing Open Source Placers to Commercial-quality Using Generative Adversarial Networks and Transfer Learning. *ACM Trans. Des. Autom. Electron. Syst.* **2024**, *29*. https://doi.org/10.1145/3636461.
29. Guo, T.; Herber, D.R.; Allison, J.T. Circuit Synthesis Using Generative Adversarial Networks (GANs). *AIAA Scitech 2019 Forum* **2019**.
30. Lu, Y.C.; Lee, J.; Agnesina, A.; Samadi, K.; Lim, S.K. GAN-CTS: A generative adversarial framework for clock tree prediction and optimization. In Proceedings of the Proc. IEEE/ACM Int. Conf. on Computer-Aided Design (ICCAD), 2019, pp. 1–8.
31. Yang, Y.; Sun, M. Knowledge Distillation Cross Domain Diffusion Model: A Generative AI Approach for Defect Pattern Segmentation. *IEEE Transactions on Semiconductor Manufacturing* **2024**, *37*, 634–642. https://doi.org/10.1109/TSM.2024.3472611.
32. Zhou, G.; Korrapati, B.; Reddy, G.R.; Chang, C.C.; Pan, J.; Hu, J.; Chen, Y.; Thakurta, D.G. PatternPaint: Practical Layout Pattern Generation Using Diffusion-Based Inpainting, 2024, [arXiv:cs.CV/2409.01348].
33. Yin, Y.; Wang, Y.; Xu, B.; Li, P. ADO-LLM: Analog Design Bayesian Optimization with In-Context Learning of Large Language Models. In Proceedings of the Proceedings of the 43rd IEEE/ACM International Conference on Computer-Aided Design. ACM, 2024, ICCAD '24, p. 1–9. https://doi.org/10.1145/3676536.3676816.
34. Chen, Z.; Huang, J.; Liu, Y.; Yang, F.; Shang, L.; Zhou, D.; Zeng, X. Artisan: Automated Operational Amplifier Design via Domain-specific Large Language Model. In Proceedings of the Proceedings of the 61st ACM/IEEE Design Automation Conference, New York, NY, USA, 2024; DAC '24. https://doi.org/10.1145/3649329.3655903.
35. Chang, C.C.; Shen, Y.; Fan, S.; Li, J.; Zhang, S.; Cao, N.; Chen, Y.; Zhang, X. LaMAGIC: language-model-based topology generation for analog integrated circuits. In Proceedings of the Proceedings of the 41st International Conference on Machine Learning. JMLR.org, 2024, ICML'24.
36. Ding, H.; Wang, Q.; et al.. LLM4Floorplan: Large language model agents for chip floorplanning. *arXiv preprint arXiv:2406.06345* **2024**.
37. Lu, S.C.; Yeh, C.C.; Cho, H.L.; Lin, Y.C.; Lin, R.B. Subitizing-Inspired Large Language Models for Floorplanning. *arXiv preprint arXiv:2504.12076* **2025**.
38. Guo, X.; Li, Y.; Kong, X.; JIANG, Y.; Zhao, X.; Gong, Z.; Zhang, Y.; Li, D.; Sang, T.; Zhu, B.; et al. Toward Engineering AGI: Benchmarking the Engineering Design Capabilities of LLMs. In Proceedings of the Advances in Neural Information Processing Systems, 2025, Vol. 38.
39. Mehradfar, A.; Zhao, X.; Niu, Y.; Babakniya, S.; Alesheikh, M.; Aghasi, H.; Avestimehr, S. AICircuit: A Multi-Level Dataset and Benchmark for AI-Driven Analog Integrated Circuit Design. *Machine Learning and the Physical Sciences Workshop @ NeurIPS* **2024**.
40. Jiang, X.; chai, z.; Zhao, Y.; Lin, Y.; Wang, R.; Huang, R. CircuitNet 2.0: An Advanced Dataset for Promoting Machine Learning Innovations in Realistic Chip Design Environment. In Proceedings of the International Conference on Learning Representations; Kim, B.; Yue, Y.; Chaudhuri, S.; Fragkiadaki, K.; Khan, M.; Sun, Y., Eds., 2024, Vol. 2024, pp. 16140–16159.
41. Blocklove, J.; Garg, S.; Karri, R.; Pearce, H. Chip-Chat: Challenges and Opportunities in Conversational Hardware Design. In Proceedings of the 2023 ACM/IEEE 5th Workshop on Machine Learning for CAD (MLCAD), 2023, pp. 1–6. https://doi.org/10.1109/MLCAD58807.2023.10299874.
42. Pearce, H.; Tan, B.; Karri, R. DAVE: Deriving Automatically Verilog from English. In Proceedings of the 2020 ACM/IEEE 2nd Workshop on Machine Learning for CAD (MLCAD), 2020, pp. 27–32. https://doi.org/10.1145/3380446.3430634.
43. Liu, S.; Fang, W.; Lu, Y.; Wang, J.; Zhang, Q.; Zhang, H.; Xie, Z. RTLCoder: Fully Open-Source and Efficient LLM-Assisted RTL Code Generation Technique. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* **2025**, *44*, 1448–1461. https://doi.org/10.1109/TCAD.2024.3483089.
44. Tsaras, D.; Grosnit, A.; Chen, L.; Xie, Z.; Bou-Ammar, H.; Yuan, M. ShortCircuit: AlphaZero-Driven Circuit Design, 2024, [arXiv:cs.LG/2408.09858].
45. Wang, Z.; Wang, J.; Yang, Q.; Bai, Y.; Li, X.; Chen, L.; Hao, J.; Yuan, M.; Li, B.; Zhang, Y.; et al. Towards Next-Generation Logic Synthesis: A Scalable Neural Circuit Generation Framework. In Proceedings of the Advances in Neural Information Processing Systems 37 (NeurIPS 2024), 2024.

 doi:10.20944/preprints202607.0874.v1

46. Lv, C.; Wei, Z.; Qian, W.; Ye, J.; Feng, C.; He, Z. GPT-LS: Generative Pre-Trained Transformer with Offline Reinforcement Learning for Logic Synthesis. In Proceedings of the 2023 IEEE 41st International Conference on Computer Design (ICCD), 2023, pp. 320–326. https://doi.org/10.1109/ICCD58817.2023.00056.
47. Pasandi, G.; Pratty, S.; Forsyth, J. AISYN: AI-driven Reinforcement Learning-Based Logic Synthesis Framework, 2023, [arXiv:cs.AI/2302.06415].
48. Karimi, R.; Faez, F.; Zhang, Y.; Li, X.; Chen, L.; Yuan, M.; Biparva, M. Logic Synthesis Optimization with Predictive Self-Supervision via Causal Transformers, 2025, [arXiv:cs.AI/2409.10653].
49. Shi, Y.; Tao, Z.; Gao, Y.; Zhou, T.; Chang, C.; Wang, Y.; Chen, B.; Zhang, G.; Liu, A.; Yu, Z.; et al. AMSnet-KG: A Netlist Dataset for LLM-based AMS Circuit Auto-Design Using Knowledge Graph RAG. In Proceedings of the Proceedings of the ACM/IEEE Transactions on Design Automation of Electronic Systems, New York, NY, USA, 2024; TODAES '24, pp. 1–27. https://doi.org/10.1145/3679065.
50. Lai, Y.; Lee, S.; Chen, G.; Poddar, S.; Hu, M.; Pan, D.Z.; Luo, P. AnalogCoder: Analog Circuit Design via Training-Free Code Generation. *Proceedings of the AAAI Conference on Artificial Intelligence* **2025**, *39*, 379–387. https://doi.org/10.1609/aaai.v39i1.32016.
51. Lai, Y.; Poddar, S.; Lee, S.; Chen, G.; Hu, M.; Yu, B.; Luo, P.; Pan, D.Z. AnalogCoder-Pro: Unifying Analog Circuit Generation and Optimization via Multi-modal LLMs, 2025, [arXiv:cs.LG/2508.02518].
52. Shen, J.; Chen, Z.; Zhuang, J.; Huang, J.; Yang, F.; Shang, L.; Bi, Z.; Yan, C.; Zhou, D.; Zeng, X. Atelier: An Automated Analog Circuit Design Framework via Multiple Large Language Model-Based Agents. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* **2025**, pp. 1–1. https://doi.org/10.1109/TCAD.2025.3573228.
53. Gao, J.; Xu, Y.; Li, W.; Chen, H.; Zhang, R.; Wang, Z.; Lin, Y.; Pan, D.Z. AnalogGenie: A Generative Engine for Automatic Discovery of Analog Circuit Topologies. In Proceedings of the Proceedings of the ACM/IEEE Design Automation Conference (DAC), New York, NY, USA, 2025; DAC '25, pp. 1–9. arXiv preprint arXiv:2503.00205, https://doi.org/10.48550/arXiv.2503.00205.
54. Liu, B.; Zhang, H.; Gao, X.; Kong, Z.; Tang, X.; Lin, Y.; Wang, R.; Huang, R. LayoutCopilot: An LLM-Powered Multiagent Collaborative Framework for Interactive Analog Layout Design. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* **2025**, *44*, 3126–3139. https://doi.org/10.1109/TCAD.2025.3529805.
55. Bhandari, J.; Bhat, V.; He, Y.; Rahmani, H.; Garg, S.; Karri, R. Masala-CHAI: A Large-Scale SPICE Netlist Dataset for Analog Circuits by Harnessing AI, 2025, [arXiv:cs.AR/2411.14299].
56. Wu, H.; He, Z.; Zhang, X.; Yao, X.; Zheng, S.; Zheng, H.; Yu, B. ChatEDA: A Large Language Model Powered Autonomous Agent for EDA. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* **2024**, *43*, 3184–3197. https://doi.org/10.1109/TCAD.2024.3383347.
57. Wu, H.; Zheng, H.; He, Z.; Yu, B. Divergent thoughts toward one goal: Llm-based multi-agent collaboration system for electronic design automation. In Proceedings of the Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), 2025, pp. 1710–1721.
58. Wang, Y.; Ye, W.; He, Y.; Chen, Y.; Qu, G.; Li, A. MCP4EDA: LLM-Powered Model Context Protocol RTL-to-GDSII Automation with Backend Aware Synthesis Optimization, 2025, [arXiv:cs.AR/2507.19570].
59. Pasandi, G.; Kunal, K.; Tej, V.; Shah, K.; Sun, H.; Jain, S.; Li, C.; Deng, C.; Ene, T.D.; Ren, H.; et al. JARVIS: A Multi-Agent Code Assistant for High-Quality EDA Script Generation, 2025, [arXiv:cs.SE/2505.14978].
60. Yu, C.; Zhang, Z. Painting on Placement: Forecasting Routing Congestion Using Conditional Generative Adversarial Nets. In Proceedings of the Proc. 56th Design Automation Conference (DAC), 2019, pp. 1–6.
61. Cheng, R.; Lyu, X.; Li, Y.; Ye, J.; Hao, J.; Yan, J. The policy-gradient placement and generative routing neural networks for chip design. In Proceedings of the Proceedings of the 36th International Conference on Neural Information Processing Systems, Red Hook, NY, USA, 2022; NIPS '22.
62. Ye, W.; Alawieh, M.B.; Lin, Y.; Pan, D.Z. LithoGAN: End-to-End Lithography Modeling with Generative Adversarial Networks. In Proceedings of the 2019 56th ACM/IEEE Design Automation Conference (DAC), 2019, pp. 1–6.
63. Sim, W.; Lee, K.; Yang, D.; Jeong, J.; Hong, J.S.; Lee, S.; Lee, H. Automatic correction of lithography hotspots with a deep generative model. In Proceedings of the Optical Microlithography XXXII. SPIE, 2019, Vol. 10961, p. 1096105.
64. Liu, Y.; Tan, S.X.D. GridVAE: Fast Power Grid EM-Aware IR Drop Prediction and Fixing Accelerated by Variational AutoEncoder. In Proceedings of the 2024 25th International Symposium on Quality Electronic Design (ISQED), 2024, pp. 1–6. https://doi.org/10.1109/ISQED60706.2024.10528766.

 doi:10.20944/preprints202607.0874.v1

65. Alawieh, M.B.; Lin, Y.; Zhang, Z.; Li, M.; Huang, Q.; Pan, D.Z. GAN-SRAF: Subresolution Assist Feature Generation Using Generative Adversarial Networks. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* **2021**, *40*, 373–385. https://doi.org/10.1109/TCAD.2020.2995338.
66. Yang, H.; Li, S.; Ma, Y.; Yu, B.; Y. Young, E.F. GAN-OPC: Mask Optimization with Lithography-guided Generative Adversarial Nets. In Proceedings of the 2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC), 2018, pp. 1–6. https://doi.org/10.1109/DAC.2018.8465816.
67. Thakur, S.; Ahmad, B.; Fan, Z.; Pearce, H.; Tan, B.; Karri, R.; Dolan-Gavitt, B.; Garg, S. Benchmarking Large Language Models for Automated Verilog RTL Code Generation. In Proceedings of the 2023 Design, Automation & Test in Europe Conference & Exhibition (DATE), 2023, pp. 1–6. https://doi.org/10.23919/DATE56975.2023.10137086.
68. Xu, K.; Zhang, G.L.; Yin, X.; Zhuo, C.; Schlichtmann, U.; Li, B. Automated C/C++ Program Repair for High-Level Synthesis via Large Language Models. In Proceedings of the 2024 ACM/IEEE 6th Symposium on Machine Learning for CAD (MLCAD), 2024, pp. 1–9. https://doi.org/10.1109/MLCAD62225.2024.10740262.
69. Nakkab, A.; Zhang, S.Q.; Karri, R.; Garg, S. Rome was Not Built in a Single Step: Hierarchical Prompting for LLM-based Chip Design. In Proceedings of the 2024 ACM/IEEE 6th Symposium on Machine Learning for CAD (MLCAD), 2024, pp. 1–11. https://doi.org/10.1109/MLCAD62225.2024.10740247.
70. Vijayaraghavan, P.; Nitsure, A.; Mackin, C.; Shi, L.; Ambrogio, S.; Haran, A.; Paruthi, V.; Elzein, A.; Coops, D.; Beymer, D.; et al. Chain-of-Descriptions: Improving Code LLMs for VHDL Code Generation and Summarization. In Proceedings of the 2024 ACM/IEEE 6th Symposium on Machine Learning for CAD (MLCAD), 2024, pp. 1–10. https://doi.org/10.1109/MLCAD62225.2024.10740226.
71. Xu, H.; Hu, H.; Huang, S. Optimizing High-Level Synthesis Designs with Retrieval-Augmented Large Language Models. In Proceedings of the 2024 IEEE LLM Aided Design Workshop (LAD), 2024, pp. 1–5. https://doi.org/10.1109/LAD62341.2024.10691855.
72. Nadimi, B.; Zheng, H. A Multi-Expert Large Language Model Architecture for Verilog Code Generation. In Proceedings of the 2024 IEEE LLM Aided Design Workshop (LAD), 2024, pp. 1–5. https://doi.org/10.1109/LAD62341.2024.10691683.
73. Wong, S.Z.; Wan, G.W.; Liu, D.; Wang, X. VGV: Verilog Generation using Visual Capabilities of Multi-Modal Large Language Models. In Proceedings of the 2024 IEEE LLM Aided Design Workshop (LAD), 2024, pp. 1–5. https://doi.org/10.1109/LAD62341.2024.10691753.
74. Shi, F.; Lee, C.; Bashar, M.K.; Shukla, N.; Zhu, S.C.; Narayanan, V. Transformer-based Machine Learning for Fast SAT Solvers and Logic Synthesis, 2021, [arXiv:cs.NE/2107.07116].
75. Salcedo, W.; Achour, S.; McBeth, C. Leveraging Generative AI for Rapid Design and Verification of a Vector Processor SoC. *IEEE Design & Test* **2024**, *41*, 8–18. https://doi.org/10.1109/MDAT.2024.3404117.
76. Rashid, M.I.; Schafer, B.C. Fast and Inexpensive High-Level Synthesis Design Space Exploration: Machine Learning to the Rescue. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* **2023**, *42*, 3939–3950. https://doi.org/10.1109/TCAD.2023.3258341.
77. Liu, M.; Ene, T.D.; Kirby, R.; Cheng, C.; Pinckney, N.; Liang, R.; Alben, J.; Anand, H.; Banerjee, S.; Bayraktaroglu, I.; et al. ChipNeMo: Domain-Adapted LLMs for Chip Design, 2024, [arXiv:cs.CL/2311.00176].
78. Achiam, J.; Adler, S.; Agarwal, S.; Ahmad, L.; Akkaya, I.; Aleman, F.L.; Almeida, D.; Altenschmidt, J.; Altman, S.; Anadkat, S.; et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774* **2023**.
79. Kashyap, P.; Ravichandiran, P.P.; Baron, D.; Wong, C.W.; Wu, T.; Franzon, P.D. Generative Adversarial Network Based Adaptive Transmitter Modeling. In Proceedings of the 2023 IEEE 73rd Electronic Components and Technology Conference (ECTC), 2023, pp. 2183–2187. https://doi.org/10.1109/ECTC51909.2023.00376.
80. Kashyap, P.; Pitts, W.S.; Baron, D.; Wong, C.W.; Wu, T.; Franzon, P.D. High Speed Receiver Modeling Using Generative Adversarial Networks. In Proceedings of the 2021 IEEE 30th Conference on Electrical Performance of Electronic Packaging and Systems (EPEPS), 2021, pp. 1–3. https://doi.org/10.1109/EPEPS51341.2021.9609124.
81. Kashyap, P.; Choi, Y.; Dey, S.; Baron, D.; Wong, C.W.; Wu, T.; Cheng, C.; Franzon, P.D. Modeling of Adaptive Receiver Performance Using Generative Adversarial Networks. In Proceedings of the 2022 IEEE 72nd Electronic Components and Technology Conference (ECTC), 2022, pp. 1958–1963. https://doi.org/10.1109/ECTC51906.2022.00307.
82. Chen, W.; Liu, C.; Huang, W.; Lyu, J.; Yang, M.; Du, Y.; Du, L.; Yang, J. AnalogTester: A Large Language Model-Based Framework for Automatic Testbench Generation in Analog Circuit Design. In Proceedings of the 2025 International Symposium of Electronics Design Automation (ISEDA), 2025, pp. 201–207. https://doi.org/10.1109/ISEDA65950.2025.11100397.

 doi:10.20944/preprints202607.0874.v1

83. Lu, J.; Lei, L.; Huang, J.; Yang, F.; Shang, L.; Zeng, X. Automatic Op-Amp Generation From Specification to Layout. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* **2023**, *42*, 4378–4390. https://doi.org/10.1109/TCAD.2023.3296374.
84. Shen, J.; Yang, F.; Shang, L.; Yan, C.; Bi, Z.; Zhou, D.; Zeng, X. ATOM: An Automatic Topology Synthesis Framework for Operational Amplifiers. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* **2025**, *44*, 1193–1198. https://doi.org/10.1109/TCAD.2024.3463534.
85. Kochar, D.V.; Wang, H.; Chandrakasan, A.P.; Zhang, X. LEDRO: LLM-Enhanced Design Space Reduction and Optimization for Analog Circuits. In Proceedings of the 2025 IEEE International Conference on LLM-Aided Design (ICLAD), 2025, pp. 141–148. https://doi.org/10.1109/ICLAD65226.2025.00011.
86. Wu, X.; Hu, F.; Babu, S.J.; Zhao, Y.; Guo, X. EasySize: Elastic Analog Circuit Sizing via LLM-Guided Heuristic Search, 2025, [arXiv:cs.AI/2508.05113].
87. Chen, G.; Zhu, K.; Kim, S.; Zhu, H.; Lai, Y.; Yu, B.; Pan, D.Z. LLM-Enhanced Bayesian Optimization for Efficient Analog Layout Constraint Generation, 2024, [arXiv:cs.AI/2406.05250].
88. Wang, P.C.; Lin, M.P.H.; Liu, C.N.J.; Chen, H.M. Layout Synthesis of Analog Primitive Cells with Variational Autoencoder. In Proceedings of the 2023 19th International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD), 2023, pp. 1–4. https://doi.org/10.1109/SMACD58065.2023.10192172.
89. Zhu, K.; Xie, Z.; Lin, Y.; Pan, D.Z. Generative-Adversarial-Network-Guided Well-Aware Placement for Analog Circuits. In Proceedings of the Proceedings of the 27th Asia and South Pacific Design Automation Conference, New York, NY, USA, 2022; ASP-DAC '22, pp. 523–529. https://doi.org/10.1109/ASP-DAC52403.2022.9712592.
90. Chen, H.C.; Wu, L.; Gao, M.; Shen, L.; Zhong, J.; Xu, Y. DocEDA: Automated Extraction and Design of Analog Circuits from Documents with Large Language Model, 2024, [arXiv:cs.AR/2412.05301].
91. Li, J.; Zhi, H.; Lyu, R.; Li, W.; Bi, Z.; Zhu, K.; Zeng, Y.; Shan, W.; Yan, C.; Yang, F.; et al. AnalogGym: An Open and Practical Testing Suite for Analog Circuit Synthesis, 2024, [arXiv:cs.AR/2409.08534].
92. Ajayi, T.; Chhabria, V.A.; Fogaça, M.; Hashemi, S.; Hosny, A.; Kahng, A.B.; Kim, M.; Lee, J.; Mallappa, U.; Neseem, M.; et al. Toward an Open-Source Digital Flow: First Learnings from the OpenROAD Project. In Proceedings of the Proceedings of the 56th Annual Design Automation Conference 2019, New York, NY, USA, 2019; DAC '19. https://doi.org/10.1145/3316781.3326334.
93. Wolf, C.; YosysHQ. Yosys Open SYnthesis Suite. Available at: https://yosyshq.net/yosys/documentation.html, 2025. Accessed: October 2025.
94. Shalan, M.; Edwards, T. Building OpenLANE: A 130nm OpenROAD-based Tapeout- Proven Flow : Invited Paper. In Proceedings of the 2020 IEEE/ACM International Conference On Computer Aided Design (ICCAD), 2020, pp. 1–6.
95. Lu, Y.C.; Lee, J.; Agnesina, A.; Samadi, K.; Lim, S.K. GAN-CTS: A Generative Adversarial Framework for Clock Tree Prediction and Optimization. In Proceedings of the 2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), 2019, pp. 1–8. https://doi.org/10.1109/ICCAD45719.2019.8942063.
96. Fan, S.K.S.; Tsai, D.M.; Yeh, P.C. Effective Variational-Autoencoder-Based Generative Models for Highly Imbalanced Fault Detection Data in Semiconductor Manufacturing. *IEEE Transactions on Semiconductor Manufacturing* **2023**, *36*, 205–214. https://doi.org/10.1109/TSM.2023.3238555.
97. Wang, S.; Zhong, Z.; Zhao, Y.; Zuo, L. A Variational Autoencoder Enhanced Deep Learning Model for Wafer Defect Imbalanced Classification. *IEEE Transactions on Components, Packaging and Manufacturing Technology* **2021**, *11*, 2055–2060. https://doi.org/10.1109/TCPMT.2021.3126083.
98. Huynh, T.; Truong, D.; Bandaragoda, T. Generative AI for Improved Defect Detection in Semiconductor Wafers. In Proceedings of the IECON 2024 - 50th Annual Conference of the IEEE Industrial Electronics Society, 2024, pp. 1–6. https://doi.org/10.1109/IECON55916.2024.10905182.
99. Jia, Z.; Yang, Q.; Li, Y.; Wang, S.; Xu, P.; Liu, Z. A Fault Diagnosis Strategy for Analog Circuits with Limited Samples Based on the Combination of the Transformer and Generative Models. *Sensors* **2023**, *23*. https://doi.org/10.3390/s23229125.
100. Carbunescu-Stoenescu, B.; David, E.; Topa, M.; Buzo, A.; Pelz, G. Distribution Modelling for Yield Analysis using Variational Autoencoders. In Proceedings of the 2023 International Symposium on Signals, Circuits and Systems (ISSCS), 2023, pp. 1–5. https://doi.org/10.1109/ISSCS58449.2023.10190853.
101. Yan, W.; Wu, E.; Rosenbaum, E. New Loss Function for Learning Dielectric Thickness Distributions and Generative Modeling of Breakdown Lifetime. In Proceedings of the 2025 IEEE International Reliability Physics Symposium (IRPS), 2025, pp. 1–9. https://doi.org/10.1109/IRPS48204.2025.10983167.

 doi:10.20944/preprints202607.0874.v1

102. Kim, J.; Nam, Y.; Kang, M.C.; Kim, K.; Hong, J.; Lee, S.; Kim, D.N. Adversarial Defect Detection in Semiconductor Manufacturing Process. *IEEE Transactions on Semiconductor Manufacturing* **2021**, *34*, 365–371. https://doi.org/10.1109/TSM.2021.3089869.
103. Zhou, H.; Jin, W.; Tan, S.X.D. GridNet: Fast Data-Driven EM-Induced IR Drop Prediction and Localized Fixing for On-Chip Power Grid Networks. In Proceedings of the 2020 IEEE/ACM International Conference On Computer Aided Design (ICCAD), 2020, pp. 1–9.
104. Lamichhane, S.; Peng, S.; Jin, W.; Tan, S.X.D. Fast Electrostatic Analysis For VLSI Aging based on Generative Learning. In Proceedings of the 2021 ACM/IEEE 3rd Workshop on Machine Learning for CAD (MLCAD), 2021, pp. 1–6. https://doi.org/10.1109/MLCAD52597.2021.9531320.
105. Kashyap, P.; Ravichandiran, P.P.; Wang, L.; Baron, D.; Wong, C.W.; Wu, T.; Franzon, P.D. Thermal Estimation for 3D-ICs Through Generative Networks. In Proceedings of the 2023 IEEE International 3D Systems Integration Conference (3DIC), 2023, pp. 1–4. https://doi.org/10.1109/3DIC57175.2023.10154977.
106. Kashyap, P.; Cheng, C.; Choi, Y.; Franzon, P. Generative Multi-Physics Models for System Power and Thermal Analysis Using Conditional Generative Adversarial Networks. In Proceedings of the 2023 IEEE 32nd Conference on Electrical Performance of Electronic Packaging and Systems (EPEPS), 2023, pp. 1–3. https://doi.org/10.1109/EPEPS58208.2023.10314864.
107. Sharma, D.D.; Pasdast, G.; Qian, Z.; Aygun, K. Universal Chiplet Interconnect Express (UCIe): An Open Industry Standard for Innovations With Chiplets at Package Level. *IEEE Transactions on Components, Packaging and Manufacturing Technology* **2022**, *12*, 1423–1431. https://doi.org/10.1109/TCPMT.2022.3207195.
108. UCIe Consortium. UCIe Specifications and UCIe 3.0 Overview. https://www.uciexpress.org/specifications, 2025. Accessed 2026.
109. IEEE Electronics Packaging Society. Heterogeneous Integration Roadmap (HIR). https://resourcecenter.eps.ieee.org/roadmap/heterogenous-integration/heterogenous-integration-roadmap, 2021.
110. Bailey, C.; Hwang, L.; Praful, F.P. Package Modeling and Analysis for Heterogeneous Integration. In Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD), 2024. https://doi.org/10.1145/3676536.3697139.
111. Krishna, R.; Nguyen, T.; Watanabe, A.O.; Becker, D.; Kumar, A.; Rosenbaum, E. A Methodology to Optimize the Number and Placement of Decoupling Capacitors in a Multilevel Power Delivery Network. In Proceedings of the IEEE Electrical Design of Advanced Packaging and Systems (EDAPS), 2022. https://doi.org/10.1109/EDAPS56906.2022.9995085.
112. Krishna, R.; Victor, A.; Penta, S.; Chen, X.; Bakir, M.S.; Kim, N.S.; Rosenbaum, E. Yield-Aware Interposer Design for UCIe Interconnects. In Proceedings of the IEEE Conference on Electrical Performance of Electronic Packaging and Systems (EPEPS), 2024, pp. 1–3. https://doi.org/10.1109/EPEPS61853.2024.10753661.
113. Krishna, R.; Victor, A.; Penta, S.; Watanabe, A.; Chen, X.; Bakir, M.S.; Kim, N.S.; Rosenbaum, E. Temperature-Dependent SPICE Models for UCIe Interconnects. In Proceedings of the IEEE Conference on Electrical Performance of Electronic Packaging and Systems (EPEPS), Milpitas, CA, USA, 2025; pp. 1–3. https://doi.org/10.1109/EPEPS63858.2025.11346572.
114. Dong, X.; Sun, S.; Jiang, Y.; Hu, J.; Gao, D.; Shi, Z.; Zhuo, C. SPIRAL+: Efficient Signal-Power Integrity Co-Analysis for Interchiplet Links Validation. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* **2025**, *44*, 3140–3153. https://doi.org/10.1109/TCAD.2025.3532822.
115. Liu, L.; Li, Y.; Lu, B.; Yang, F. GNN-SP: Fast S-Parameter Estimation of Chiplet Interconnect via Graph Neural Network. *IEEE Transactions on Components, Packaging and Manufacturing Technology* **2024**. https://doi.org/10.1109/TCPMT.2024.3449330.
116. Pfromm, L.; Kanani, A.; Sharma, H.; Solanki, P.; Tervo, E.; Park, J.; Doppa, J.R.; Pande, P.P.; Ogras, U.Y. MFIT: Multi-Fidelity Thermal Modeling for 2.5D and 3D Multi-Chiplet Architectures. *arXiv preprint arXiv:2410.09188* **2025**.
117. National Institute of Standards and Technology. CHIPS R&D Chiplets Interfaces Technical Standards Workshop Summary Report. https://nvlpubs.nist.gov/nistpubs/CHIPS/NIST.CHIPS.1400-2.pdf, 2024.
118. Sawai, Y.; Okuyama, S.; Ono, Y.; Omori, M.; Ueda, Y. Rapid and Flexible Yield Analysis Powered by Large-Scale Language Model. In Proceedings of the 2024 International Symposium on Semiconductor Manufacturing (ISSM), 2024, pp. 1–4. https://doi.org/10.1109/ISSM64832.2024.10874915.
119. Liu, Z.; Wang, Y.; Vaidya, S.; Ruehle, F.; Halverson, J.; Soljacic, M.; Hou, T.Y.; Tegmark, M. KAN: Kolmogorov–Arnold Networks. In Proceedings of the The Thirteenth International Conference on Learning Representations, 2025.

 doi:10.20944/preprints202607.0874.v1

120. Li, D.; Wang, Z.; Chen, Y.; Jiang, R.; Ding, W.; Okumura, M. A survey on deep active learning: Recent advances and new frontiers. *IEEE Transactions on Neural Networks and Learning Systems* **2024**, *36*, 5879–5899.
121. Shen, M.; Zhang, J.Y.; Chen, L.; Yan, W.; Jani, N.; Sutton, B.; Koyejo, O. Labeling Cost Sensitive Batch Active Learning For Brain Tumor Segmentation. In Proceedings of the 2021 IEEE 18th International Symposium on Biomedical Imaging (ISBI), 2021, pp. 1269–1273. https://doi.org/10.1109/ISBI48211.2021.9434098.