(Special Session)

# ChipMind: LLMs for Agile Chip Design

Farshad Firouzi*, David Z. Pan†, Jiaqi Gu*,
Bahar Farahani§, Jayeeta Chaudhuri*, Ziang Yin*, Pingchuan Ma*, Peter Domanski*, and Krishnendu Chakrabarty*
*Arizona State University
†University of Texas at Austin
§Shahid Beheshti University

***Abstract*—The increasing complexity of semiconductor design, along with stringent performance, power, and time-to-market requirements, has outpaced the capabilities of traditional Electronic Design Automation (EDA) methodologies. Conventional design workflows rely on manual intervention for critical tasks such as hardware description, synthesis optimization, and verification, leading to inefficiencies and scalability limitations. Large Language Models (LLMs) present a transformative approach by automating key stages of the design pipeline, enabling intelligent synthesis tuning, test generation, and security analysis. This paper introduces ChipMind, an LLM-driven framework comprising specialized agents and modules for digital and analog chip design. ChipMind integrates AI-driven methodologies to enhance design efficiency, accelerate prototyping, and optimize key design trade-offs, thereby addressing fundamental challenges in modern semiconductor development.**

***Index Terms*—AI for chip design, Large Language Models (LLMs), Electronic Design Automation (EDA), hardware security**

## I. Introduction

The growing complexity of semiconductor design, alongside demands for higher performance, lower power, and faster development, has outpaced traditional design methodologies. Traditional Electronic Design Automation (EDA) workflows rely heavily on human expertise, making them labor-intensive and inefficient. Artificial Intelligence (AI), particularly Multi-Modal Large Language Models (LLMs), offers a transformative solution by automating key stages of the chip design pipeline, enabling rapid exploration, intelligent optimization, and automated verification. By reducing time-to-market and lowering the barrier to entry, LLMs enhance accessibility, improve security analysis, and enable cost-effective prototyping, making AI-driven design essential for the future of hardware development.

To fully leverage the benefits of LLMs in chip design, it is crucial to understand the specific challenges they address and the improvements they bring. The following key areas highlight the advantages of incorporating LLM-driven approaches into chip design:

- **Bridging the Specification Gap:** The disparity between high-level, natural language specifications provided by domain experts and the precise, low-level hardware descriptions required for implementation introduces inefficiencies and potential misinterpretations. LLMs facilitate the automated transformation of abstract design intent into formal hardware representations, enhancing accuracy, consistency, and design productivity.
- **Accelerating Time-to-Market:** Traditional chip design methodologies are inherently iterative, requiring extensive synthesis, verification, and optimization cycles, leading to prolonged development timelines. AI-driven automation accelerates the process by optimizing design decisions and improving efficiency, significantly reducing turnaround time while maintaining key design trade-offs.
- **Enhancing Design Efficiency and Managing Complexity:** The increasing architectural complexity of modern semiconductor devices challenges traditional design and optimization methodologies. AI-driven approaches efficiently navigate large design spaces, automate repetitive tasks, and optimize Power, Performance, and Area (PPA) trade-offs.
- **Democratizing Chip Design for Non-Experts:** The steep learning curve associated with Hardware Description Languages (HDLs) and EDA tools limits accessibility to chip design. LLM-driven design frameworks lower the barrier, enabling individuals with minimal hardware design experience (e.g., industry practitioners) to contribute effectively.
- **Enabling Rapid Prototyping and Design Exploration:** LLM facilitates the automated generation and evaluation of multiple variants, allowing for systematic assessment of trade-offs before committing to fabrication. This enables a more iterative and agile hardware development process, mitigating design risks.
- **Addressing Limited Market Incentives:** Developing custom silicon for niche applications is often constrained by high Non-Recurring Engineering (NRE) costs and limited market potential. LLM-driven design automation significantly reduces the resource investment required for hardware development, making it economically viable to develop specialized, Application-Specific Integrated Circuits (ASICs) for emerging applications.
- **Reducing Costs and Resource Requirements:** LLM-driven methodologies reduce reliance on large engineering teams by automating critical aspects of the chip design flow, including synthesis, verification, testing, and performance tuning. Additionally, LLM-empowered frameworks contribute to cost-efficient silicon utilization,

979-8-3315-2144-8/25/$31.00 ©2025 IEEE

lowering manufacturing expenses and improving yield.

- **Enhancing Customization and Adaptability:** LLM-powered frameworks enable the rapid adaptation of hardware designs to meet domain-specific requirements, such as specialized AI accelerators, low-power embedded processors, or high-performance computing architectures. These methodologies can facilitate tailored optimizations aligned with workload characteristics.
- **Improving Security, Reliability, and Robustness:** The increasing sophistication of hardware security threats, such as hardware Trojans, necessitates advanced verification and detection mechanisms. LLM-driven security analysis enables automated vulnerability detection, anomaly identification, and risk assessment, strengthening the security posture of hardware implementations while enhancing reliability through predictive failure analysis.

While LLMs are transforming chip design by automating complex hardware development tasks, the efficient training and inference of LLMs themselves depend on specialized hardware accelerators, which, in turn, can be optimized and co-designed using LLM-driven methodologies. The increasing computational demands of large-scale AI models have accelerated the development of domain-specific architectures, such as TPUs, AI-optimized GPUs, and custom silicon for deep learning. These advancements illustrate a symbiotic relationship between AI-driven chip design and AI accelerators, wherein innovations in one domain directly influence progress in the other.

This paper presents ChipMind, an LLM-driven agentic framework designed to automate and optimize the entire chip design process, covering both digital and analog domains. ChipMind integrates specialized LLM modules/agents for tasks such as HDL generation, synthesis, testing, SPICE modeling, and security analysis. LLM-driven approaches have received considerable attention since the advent of GPT models. However, all existing solutions typically focus on specific tasks within the broader domains of chip design, such as digital or analog design, addressing individual functions like code generation. To date, no comprehensive framework has been developed that integrates all facets of the chip design process in a unified and cohesive manner. This paper presents the first effort to create a holistic suite of LLM modules designed to address the full spectrum of chip design tasks, providing an end-to-end solution for both digital and analog domains. By utilizing LLM-powered methodologies, ChipMind enhances design efficiency, scalability, and adaptability, addressing key challenges in modern semiconductor development. This framework represents a unified approach to chip design, enabling faster prototyping, improved hardware security, and more efficient, agile exploration of design trade-offs across both digital and analog circuits.

The rest of this paper is structured as follows. Section II reviews related work in AI-driven chip design. Section III presents the ChipMind architecture, describing its modular and agentic framework. Section IV examines the application of ChipMind in digital design, focusing on LLM-driven methodologies for HDL generation, synthesis, and test generation. Section V discusses the role of ChipMind in analog circuit design, including SPICE modeling and simulation-based optimization. Section VI explores LLM-based techniques for hardware Trojan detection. Section VII introduces a specialized accelerator for LLM processing, leveraging silicon photonics to enhance computational efficiency for AI-driven chip design. Finally, Section VIII provides concluding remarks and outlines directions for future research.

## II. Related Work

### A. LLMs for Design Automation and Hardware Security

LLMs have been utilized in multiple stages of EDA, including tasks such as code generation, circuit synthesis and design verification, and security applications such as Trojan insertion, detection, and mitigation. For Verilog code generation, recent work fine-tuned existing LLMs with Verilog datasets [1]. The open-source CodeGen LLM outperformed state-of-the-art commercial models in generating functionally correct designs [2]. The Chip-Chat project utilized GPT-4 to develop an 8-bit accumulator-based microprocessor architecture, showcasing the potential of LLMs in hardware design. To support industrial chip design, Domain-Adaptive Pre-Training (DAPT) followed by Supervised Fine-Tuning (SFT) was applied to foundation LLMs, leading to the development of an assistant chatbot for chip design [3]. Additionally, LLM-based script generation methods were explored in [4], [5].

The capability of LLMs have been further leveraged to assist in debugging and verifying design correctness, and exploring design vulnerabilities. RTLFixer explores LLM agents along with Retrieval-Augmented-Generation, to address syntax errors in HDL [6]. In [7], authors fix buggy lines in Verilog by applying LLM-suggested replacement codes. On the other hand, LLMs have been leveraged for generating malicious circuits or Trojans in digital circuits. In [8], LLMs are taught to generate digital Trojans with different types, targeting DoS and timing attacks. In [9] LLMs are used to identify specific segments in Verilog codes that can be modified for Trojan insertion, while in [10], LLMs are harnessed to insert vulnerabilities e.g., deadlocks, in finite-state machines.

Beyond digital circuits, LLMs have been applied to analog circuit design. [11] proposed an LLM-based approach for generating analog circuits with a feedback-enhanced flow to enable self-correcting analog circuit generation without requiring LLM training. In [12], LLMs are used to convert schematics into SPICE netlists by integrating iterative manual feedback. [13] leverages LLMs for detecting and localizing analog Trojans within circuit netlists based on prompt engineering and few-shot learning approach. The authors extend this work [13] in [14], where they explore Trojan mitigation strategies in A/MS designs.

### B. Hardware Accelerators for LLM

Large language models (LLMs) such as GPT-4 [15] and DeepSeek R1 [16] have undergone a rapid evolution, driven

by advances in deep learning architecture, massive parallelism, and vast computational resources. Their ability to understand and generate human language with remarkable fluency has unlocked new possibilities in diverse applications, from customer service and content creation to scientific research and real-time analytics. This growing use of LLMs and generative AI has consequently led to extensive efforts to develop specialized hardware accelerators capable of handling their enormous computational demands while scaling efficiently in both performance and power consumption.

Conventional hardware accelerators for LLMs primarily rely on digital architectures, including GPUs, FPGAs, and ASICs. GPUs, as widely adopted platforms, offer high throughput and flexible programmability, with optimized inference frameworks such as LLama [17], LightSeq2 [18], and UltraFastBERT [19] further improving efficiency. FPGAs have been explored for low-latency, reconfigurable inference using various optimizations. Designs like MnnFast [20] and HA-FPGA [21] focus on memory-efficient data streaming to reduce bandwidth overhead and data movement, while other works employ structured weight compression methods, leveraging block-circulant representations or quantization-aware training to reduce model size while maintaining accuracy [22], [23]. Dynamic hardware reconfiguration techniques have also been proposed to adapt FPGA architectures for different NLP workloads, enhancing utilization and flexibility [24]. Similarly, ASIC-based accelerators (e.g., Google's TPU [25], Habana Gaudi [26]) provide massive parallelism and efficiency benefits, especially for matrix operations central to Transformer-based LLMs.

In addition to digital accelerators, photonic accelerators, leveraging the inherent parallelism and low-latency propagation of optics for large-scale matrix multiplications, have emerged as a promising avenue for ultra-fast, energy-efficient neural network acceleration. Early photonic designs [27]–[34] primarily targeted convolutional or fully connected networks, they often faced challenges such as limited operand encoding speed, narrow operand-value representations, and scalability constraints. However, recent advancements have significantly expanded their capabilities, enabling dynamic, full-range computations for large-scale workloads. For instance, Lightening Transformer (LT) [35] addresses high-speed reconfigurability by introducing a dynamic photonic tensor core (DPTC) that supports full-range tensor multiplication, while TeMPO [36] employs a time-multiplexed strategy, integrating customized slow-light modulators, optical splitters, and parallel photocurrent accumulation to minimize area and power overhead, making it well-suited for large-scale scaling. Optical Transformer (OT) [37] highlights the potential for scaling energy efficiency beyond digital systems by performing large-scale matrix multiplications with interference-based optical hardware, finding an asymptotic advantage in energy per multiply–accumulating as model width grows. Meanwhile, to manage the noise crosstalk and thermal fluctuations caused by tuning photonic devices, TRON [38] employs careful microring resonator design and a hybrid tuning approach that models and reduces these overheads, thereby enabling end-to-end inference of Transformer-based models and marking a significant milestone in large-scale photonic acceleration.

## III. ChipMind: Overview

ChipMind is an agentic, modular LLM-driven framework designed to automate and optimize chip design workflows across digital, analog, and security domains (see Fig. 1). It comprises task-specific LLM modules, each tailored to a distinct aspect of the design process, ensuring efficiency, scalability, and continuous refinement.

- **LLM for Digital Design Flow**, which facilitates HDL code generation, synthesis, and test automation, enabling the seamless transformation of high-level specifications into optimized hardware implementations.
- **LLM for Analog Design Flow**, which enables SPICE code generation, circuit topology optimization, and simulation-driven refinement, improving the precision, efficiency, and performance of analog circuit design.
- **LLM for Hardware Security**, which identifies and mitigates malicious modifications (i.e., hardware Trojans) in analog circuits.

Each module/agent functions autonomously while aligning with shared design principles, iterative refinement strategies, and knowledge-based optimization, ensuring a highly adaptive and AI-enhanced chip design process.

## IV. LLM for Digital Design Flow

The modular and agentic workflow of ChipMind for Digital Design Flow, as outlined in Algorithm 1, is built on customized LLM-driven modules, each dedicated to a specific design task—code generation, synthesis, and test generation. This structured and flexible pipeline ensures efficient, automated, and continuously optimized design refinement, where each LLM module independently processes its task while aligning with shared constraints and iterative feedback mechanisms to enhance overall chip design quality.

The process begins with HDL code generation, where a user provides a high-level design specification consisting of module names, input/output signals, and, optionally, architectural details. The LLM for HDL generation translates this specification into synthesizable HDL, after which syntax validation ensures correctness before execution. If errors are detected, the LLM iteratively refines and regenerates the HDL until correctness is achieved. Once the HDL is validated, the LLM for synthesis generates a synthesis script, which is executed to convert the HDL into a gate-level netlist. The achieved PPA metrics (timing, power, area) are analyzed, and if they do not meet the target constraints, the LLM queries the knowledge base (KB) and knowledge graph (KG) to derive improved synthesis constraints. The process then iterates, updating and refining the synthesis script until optimal PPA values are achieved. Following synthesis, the LLM for test generation automatically generates ATPG test scripts and executes them to analyze achieved fault coverage. If the coverage is insufficient, the

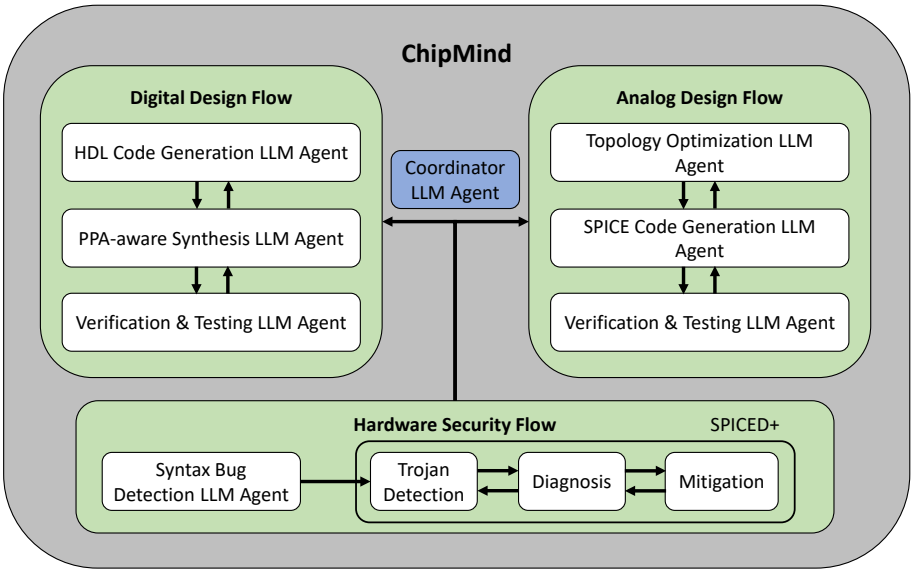


Fig. 1: ChipMind Framework: Detailed Components and Workflow Process.

LLM queries KG/KB for optimization strategies and iteratively refines the test patterns.

As depicted in Algorithm 1, the iterative feedback loop ensures continuous improvement across all design stages. The framework leverages LLM-driven optimization, knowledge-based learning, in-context learning, and iterative validation to guarantee high-quality HDL generation, optimized synthesis, and effective test generation. Its modular architecture supports future enhancements such as place-and-route automation, formal verification, and advanced optimization techniques.

## V. LLM for Analog Circuit Design

Analog circuit design has long been a heavily manual process, from circuit topology/schematic generation to device sizing and layout generation. In the entire design process, extensive circuit simulations are performed to check if various design constraints and objectives can be met and optimized. However, such design processes are very tedious and not scalable. There are some recent efforts on LLM for analog circuit design [39], but the field is evolving rapidly.

Among all analog design steps, analog circuit schematic or topology design is the first step. AnalogCoder [40] proposed the first training-free LLM agent for analog topology design through Python code generation, i.e., PySpice which can be mapped to the corresponding SPICE netlist and circuit topology. Through a feedback-enhanced flow / prompt engineering that includes four-stage checking and tailored domain-specific prompts, AnalogCoder enables the automated and self-correcting analog circuit design with a high success rate. To further allow for designing more complicated analog circuits, the AnalogCoder flow stores the working basic circuits into a library, which can then be used as reusable modules to build more complicated analog circuits. The overall flow of AnalogCoder is shown in Fig. 2. AnalogCoder also provided a set of analog circuit benchmarks and compared many baseline LLM agents (e.g., GPT 4o, Claude 3.5, Llama 3) with the proposed AnalogCoder techniques. AnalogCoder has been made open source, including the 24 circuit benchmarks under test.

Most recently, a domain-specific GPT model AnalogGenie [41] was proposed to generate analog circuits. It built an extensive dataset that consists of more than 3,000 distinct analog circuit topologies with diverse functionalities from various analog IC design textbooks, etc., and then applied data augmentation to expand the circuit topologies by over 70×. Using an expressiveness-enhanced graph representation and customized tokenizer to pre-train the GPT model, AnalogGenie showed improved scalability to handle a variety of analog circuits and generate novel designs.

While LLM-based AnalogCoder or AnalogGenie can enable non-experts to generate circuit topologies efficiently, to meet stringent performance requirements, sophisticated device sizing optimization is usually needed, sometimes together with topology/schematic co-optimization. There are many papers published on analog device sizing, including some recent ones using reinforcement learning [42]. The recent work [43] further proposed a data-driven approach by building a sized topology library and directly searching for properly sized topologies under given circuit specifications. On the LLM side, the ADO-LLM work [44] proposed the first LLM-aided Bayesian optimization (BO) framework to leverage LLM's ability to infuse domain knowledge to quickly generate viable design points for BO in finding high-value design areas during exploration. How to effectively combine LLM-based circuit topology generation with sizing while meeting various stringent design specifications and targets is still an open

**Algorithm 1** ChipMind Digital Design Flow

**Input:** Task Specification $T$ (e.g., Code Generation, Synthesis, or Test Generation), Target Constraints (e.g., PPA, Coverage), Max Iterations
**Output:** Optimized and Validated Design Artifact
**procedure** CHIPMIND-WORKFLOW
    **Initialize Task-Specific LLM** (Code, Synthesis, or Test)
    **Generate Initial Artifact** (e.g., HDL Code, Synthesis Script, or Test Script) using LLM
    **Perform Syntax Validation** ▷ Ensures correctness before execution
    **if** Syntax Errors Detected **then**
        Apply Fixes and **Re-generate Artifact**
    **end if**
    **Execute Task** (e.g., Compile HDL, Run Synthesis, or Generate Test Results)
    **Extract Achieved Metrics** (e.g., PPA for Synthesis, Coverage for Testing)
    **Initialize Optimization Loop**
    **while** Target Constraints Not Met **and** Max Iterations Not Reached **do**
        **Analyze Achieved Metrics vs. Target Constraints**
        Query **Knowledge Graph (KG)** and **Knowledge Base (KB)** for Optimization Strategies
        **Regenerate and Update Artifact** Using LLM with KG/KB Insights
        **Perform Syntax Validation on the Updated Artifact**
        **if** Syntax Errors Detected **then**
            Apply Changes and **Re-generate Artifact**
        **end if**
        **Execute Updated Artifact**
        **Extract New Achieved Metrics**
    **end while**
    **Store Final Optimized Artifact**
    **Return: Optimized and Validated Design Output**
**end procedure**

problem for industry-strength applications.

In terms of layout design and optimization, analog circuits often have a lot more constraints than their digital counterparts, e.g., symmetric placement and routing [45]. LLM can offer recommendations on layout constraints and provide human interactive feedback/insights to improve analog layout performance, as shown by a recent work on the LLM-powered multi-agent for interactive LayoutCopilot [46]. LayoutCopilot employs multiple agents, including classifier agent, analyzer agent, solution refinement agent, and solution adapter agent to convert natural language instructions into executable script commands, thus improve the layout design process.

LLM for analog circuit design is a rapidly evolving field. While some promising results have been shown, there are still a lot of challenges due to lack of large volume, high-quality designs/dataset at various levels of abstractions. Our overarching goal is to have an analog design automation framework from specification to layout. We expect LLM to work with other optimization and ML techniques to achieve holistic and user-friendly design automation and optimization.

## VI. LLM FOR HARDWARE TROJANS

The objective of this module is to enhance the detection, localization, and mitigation of both syntactical and functional bugs in analog designs. Note that functional bugs, also referred to as Trojans, are malicious modifications in the circuit netlist. These Trojans usually occupy small footprint and are activated only under specific analog stimuli (e.g., voltage), and remain dormant otherwise. Integrating LLM-guided analysis for Trojan detection in analog designs offers several key advantages.

- **Textual analysis capability**: Pre-trained LLMs can understand SPICE syntax, netlist connectivity, and simulation logs, reducing preprocessing effort and improving engineering productivity.
- **Simulation insights**: LLMs implicitly learn normal vs. anomalous nodal behavior through internal attention mechanisms, aiding anomaly detection.
- **Trojan Detection and Localization**: By parsing logs and identifying outlier current/voltage patterns, LLMs can detect and localize Trojan-impacted nodes in the analog design.
- **No Area and Power Overhead**: LLM-based analysis eliminates the need for additional Trojan detection hardware, ensuring zero area and power overheads.

Fig. 3 illustrates the overall workflow of bug detection and mitigation.

### A. Syntactical Bug Detection

We leverage pre-trained LLMs to detect syntactical bugs in SPICE netlists. Syntactical bugs may include floating nodes in the netlist, incorrect instance specifications, or missing component nodes. Although advanced LLMs such as GPT-3.5 and GPT-4 excel at understanding SPICE syntax and component configuration, they often flag syntactically correct SPICE code as buggy, thereby resulting in increased false positives [13]. To address this, we exploit a feedback-based refinement approach. Specifically, the SPICE syntax rules are iteratively refined/updated based on the LLM response. The refined rules are then incorporated into the prompt along with the test netlist for evaluation. The LLM generates a well-structured bug detection report detailing: (i) identified syntactical bugs, (ii) location of bugs, and (iii) suggested corrections of the buggy lines. By applying the iterative refinement technique, false positives were reduced from 23.9% to 3.8% across benchmark circuits, including designs from the open-source AMSNet repository [12], OPAMP, and bandgap filter.

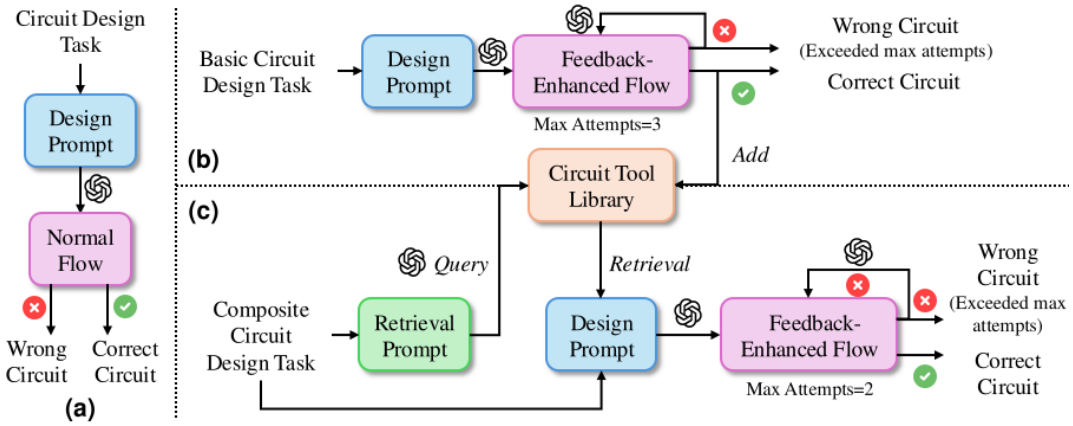


Fig. 2: **AnalogCoder Overview [40]. (a) Previous method with simple prompts. (b) AnalogCoder method for basic circuit design tasks.** The feedback-enhanced design flow enables automated error fixing with LLMs. Successfully designed circuits are added to the circuit tool library, while failed designs are returned to the LLM for automatic fixing. **(c) AnalogCoder method for composite circuit design tasks.** The process adds a step of querying the library to retrieve invocation methods for subcircuits, which are then integrated into the design prompt to facilitate the design of composite circuits.

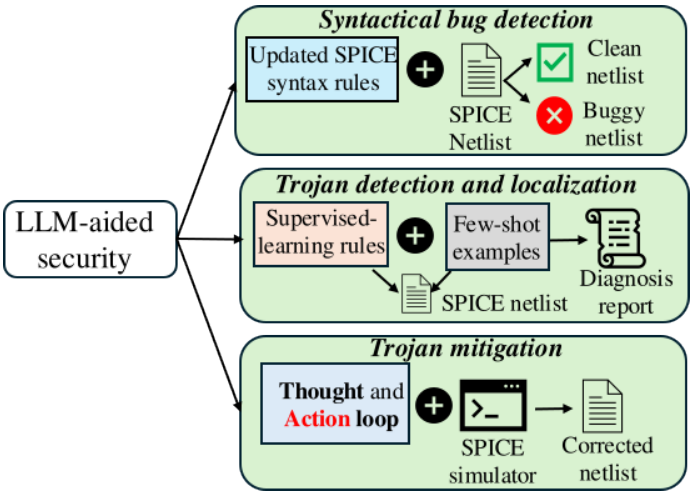


Fig. 3: Exploring different applications of LLMs in analog design security, including bug detection, Trojan detection, and SPICE netlist correction.

### *B. Trojan Detection and Localization*

To detect analog Trojans, the proposed approach employs a supervised-learning framework that teaches the LLM to analyze netlists and detect Trojan-induced anomalies from current and voltage simulation logs [13]. The supervised-learning rules are crafted based on the following Trojan-induced anomalies: (i) deviation in primary output voltage, (ii) abnormal voltage deviation in Trojan-impacted nodes of the circuit, and (iii) anomalous surge of transistor current.

To enhance learning, the LLM is provided with supervised-learning rules along with few-shot examples of both Trojan-free and Trojan-inserted netlists. This enables it to establish correlations between anomalies in the node voltage/current behavior and Trojan presence. The netlist, current simulation log, and voltage simulation log are fed as inputs to the LLM for evaluation. The LLM generates a structured Trojan diagnosis report, identifying the Trojan lines in the netlist, the Trojan-impacted nodes, and a detailed reasoning explaining the detection based on anomalous deviations in the log files. Additionally, the LLM assigns a confidence score corresponding to each suspected Trojan-impacted node. The confidence score is critical in determining the likelihood of Trojan presence, further enhancing the reliability of LLM-based analysis. The proposed approach achieves an average Trojan coverage of 84%, a true-positive rate of 93.4%, and precision of 90.3% across analog benchmark circuits.

### *C. Trojan Mitigation*

In the scenario where list of Trojan-impacted nodes are identified, an absence of verification and mitigation techniques can significantly impact the detection accuracy. We address this by providing an automated tool, SPICED+, for Trojan detection as well as mitigation [14]. SPICED+ integrates LLM-guided analysis with iterative netlist modification and simulation-based validation of circuit performance. It analyzes the HSPICE-generated simulation logs of the netlist-under-evaluation, comparing them against pre-defined performance specifications using the SPICED tool [13]. If anomalies are identified, SPICED+ leverages a systematic evaluation by iteratively removing the Trojan lines identified by SPICED, re-simulating the modified netlist using HSPICE, and comparing results to determine if the Trojan is mitigated or if the circuit performance degrades. The framework maintains a data management system to restore the previous netlist state, preventing modifications in the non-Trojan lines. Key decisions in the workflow, such as removing, retaining, or restoring suspect lines, are guided by pre-defined rules encoded as Thought and Action, enabling automated verification without manual intervention. SPICED+ achieves an average $TMR$ of 94.6% and an average $FPR_{tr}$ of 1.4% for the evaluated benchmark

circuits, where $TMR$ is the percentage of Trojan lines that SPICED+ successfully removes from the netlist and $FPR_{tr}$ is the percentage of non-Trojan lines that are incorrectly classified as malicious by the LLM.

## VII. Electronic-Photonic Heterogeneous LLM Accelerators

While LLMs are transforming semiconductor design, security, and verification by automating complex workflows, their own computational demands are rapidly escalating. The increasing scale of generative AI models necessitates specialized hardware accelerators optimized for throughput, energy efficiency, and scalability. While conventional digital accelerators such as GPUs, TPUs, and ASICs have made significant strides, the increasing model size and complexity of AI models introduce severe memory bottlenecks, excessive power consumption, and scalability limitations. Emerging AI computing platforms, particularly photonic computing and electronic processing-in-memory (PIM) architectures, offer promising solutions by reducing data movement, leveraging massive parallelism, and unlocking new levels of performance for LLM acceleration. In the following sections, we review state-of-the-art (SoTA) photonic and PIM-based AI accelerators, discuss *cross-layer co-optimization* techniques, and highlight their current limitations while proposing a *hybrid computing paradigm* that integrates the strengths of both approaches to achieve breakthrough performance in LLM acceleration.

### A. Photonic Accelerators for Language Models

Photonic accelerators leverage the parallelism and low-latency propagation of optics to perform speed-of-light matrix operations with superior energy efficiency. Most photonic accelerators are based on static photonic tensor cores (PTCs), which encode weight matrices into circuit transmissions to efficiently perform stationary matrix-vector multiplications (MVMs) on the input optically encoded vectors, i.e., $y = \mathbf{W}x$. For example, PTCs based on microring resonator (MRR) weight banks or Mach-Zehnder interferometer (MZI) arrays have been widely explored to build optical multi-layer perceptron (MLP) and convolutional neural networks (CNNs), showing orders-of-magnitude higher TOPS and TOPS/W than their digital electronics counterparts [27], [31], [47]. However, due to their weight-stationary nature, these architectures are inefficient for dynamic tensor operations in Transformer-based models, particularly in self-attention layers that require matrix-matrix multiplications such as $QK^\top$. This limitation has driven research toward dynamic photonic tensor cores, where both matrix operands can be reprogrammed at high speed to support arbitrary positive/negative matrix encoding.

Example Design 1: Lightening-Transformer (LT) [35] exemplifies this paradigm by leveraging spatial/spectral parallelism in a coupler crossbar array, enabling optically encoded dynamic dot-product computations, i.e., $z = x^\top y$. It further enables one-shot matrix-matrix multiplications $Z = X \times Y$ using multiple wavelengths for parallel processing. Beyond its innovative dynamic PTC design, LT integrates architecture-level optimizations, including on-chip optical interconnects for efficient data broadcast and advanced buffering strategies to hide memory access latency. LT achieves $>2.6\times$ energy reduction and $>12\times$ lower latency compared to prior photonic accelerators and delivers 2-3 orders of magnitude lower energy-delay product than electronic Transformer accelerators while maintaining digital-comparable accuracy [35].

Example Design 2: Building on the coupler array PTC architecture, TeMPO [36] introduces a time-multiplexed computing methodology to address the longstanding ADC bottleneck in photonic accelerators. Instead of immediately converting partial products into the digital domain, TeMPO aggregates them spatially via photocurrent summation and temporally via current integrators in the analog domain, significantly lowering both power consumption and system complexity. To further improve hardware utilization, TeMPO shares a common set of input modulators and reuses a single pool of analog integrators and ADCs for partial-product accumulation at the tile level. This amortizes the overhead of additional photonic components, improving efficiency and scalability.

By tailoring customized *computing-specific* photonic components to the demands of analog AI inference, TeMPO effectively mitigates key challenges such as large device footprints and high tuning power, further advancing the feasibility of photonic AI accelerators. TeMPO delivers digital-comparable task accuracy with superior quantization/noise tolerance. It achieves a 368.6 TOPS peak performance, 22.3 TOPS/W energy efficiency, and 1.2 TOPS/mm$^2$ compute density, pushing the Pareto frontier in AI hardware.

While photonic accelerators have demonstrated promising performance for LLM inference, they remain an emerging technology with several critical challenges that hinder their practical deployment in real-world AI applications.

❶ **Hardware Robustness**: As inherently analog AI hardware, photonic accelerators suffer from computing fidelity degradation due to various hardware non-idealities, such as process variations, thermal fluctuations, and fabrication variation-induced errors. For example, LT and TeMPO both rely on coherent optical interference for computing, where phase stability is essential to preserving the integrity of encoded data. However, even small fabrication errors or temperature-induced phase shifts can disrupt the accuracy of computations, which can be further amplified for deep models. Robust error compensation and adaptive calibration techniques are necessary to mitigate these effects and ensure reliable LLM acceleration in practical deployment [48]–[50].

❷ **Numerical Precision Constraints**: Photonic LLM accelerators encode data using optical signal magnitudes or intensities, requiring high-speed digital-to-analog converters (DACs) and ADCs for encoding and readout. However, the power consumption of DACs/ADCs scales exponentially with bit precision, making high-resolution conversion infeasible for energy-efficient operation. Additionally, analog circuit noise further limits the effective number of bits (ENOB) that can be reliably supported. As a result, photonic accelerators typically

| Property | SRAM 22nm | ReRAM 32nm | Photonics |
|---|---|---|---|
| Resolution | 1-bit × 8 cells=8-bit | 2-bit × 4 cells = 8-bit | 4∼6-bit |
| Tile Size | 256 crossbars, 128×128 | 64 crossbars, 128×128 | 2 cores, 14×14 |
| ADC/tile | 256 SARADC 7-Bit | 64 SARADC 8-Bit | 392 SARADC 8-Bit |
| Cell Area | $< 1\mu m^2$ | $< 1\mu m^2$ | $> 1000\mu m^2$ |
| Arch Size | 100 Tiles | 100 Tiles | 2 Tiles |
| Program latency | $\sim$ 1 ns | $\sim$ 100 ns | $\sim$ 100 ps |
| Static power | Medium | Low | High |
| Clock | 100 MHz | 100 MHz | 3 GHz |

TABLE I: Comprehensive comparison of multi-tile accelerators with SRAM, ReRAM, and photonics.

operate at 4-6 bit precision, leading to quantization errors that may degrade LLM inference accuracy. Addressing this limitation requires hybrid digital compensation and precision-aware AI model co-design to maintain numerical accuracy while preserving the energy efficiency of photonic computing.

### *B. PIM-based Accelerators*

PIM accelerators co-locate storage and computation to minimize data movement, a major source of inefficiency in large-scale AI workloads. Designs such as ISAAC [51] leverage non-volatile crossbar arrays (e.g., ReRAM [52], MRAM [53]) to directly perform matrix multiplications by applying activation vectors to wordlines and accumulating partial sums along bitlines. This weight-stationary approach is particularly effective for large matrix-vector operations, significantly reducing both memory traffic and overall power consumption. To improve numerical precision, PIM architectures employ bit-slicing techniques, concatenating multiple partial products with different bit significance to achieve precision beyond 8 bits. As electronic memory cells are ultra-compact in size, it can afford to use spatial parallelism for bit-slicing. Additionally, SRAM-based PIM architectures, due to their fully digital nature, generally exhibit high hardware robustness against process variations and environmental fluctuations. However, purely PIM-based solutions face non-trivial challenges that limit their ability to efficiently support highly dynamic workloads such as Transformers.

❶ **Memory Cell Endurance**: Non-volatile PIM technologies like ReRAM and MRAM support weight-stationary dataflows but suffer from limited memory cell re-write endurance, making them unsuitable for frequent weight reconfiguration. This limitation poses challenges for Transformer-based workloads, which require dynamic model updates and matrix operand remapping during execution.

❷ **Power Overhead**: Even with SRAM-based PIM [54], where write endurance is less of a concern, the programming overhead for large networks remains non-trivial and may lead to inflated setup times. Moreover, high-resolution PIM computations require extensive ADCs, leading to increased power draw from ADC operations, leakage currents, and switching costs within large crossbar arrays. These factors make it challenging to scale PIM accelerators for energy-efficient high-performance LLM inference.

### *C. Photonics-PIM Hybrid LLM Accelerator*

Table I summarizes the key characteristics of SRAM-, ReRAM-, and photonics-based accelerators, comparing their resolution, latency, power consumption, and integration cost. Each technology has distinct strengths that can complement one another in AI acceleration. Photonic accelerators excel at executing high-speed matrix multiplications at GHz-scale frequencies, making them well-suited for compute-bound, dynamic workloads. In contrast, PIM accelerators, though relatively slower, are highly efficient for memory-bound operations and weight-stationary computations that require frequent data access with minimal movement.

Since neither approach alone provides an optimal balance of speed, precision, power efficiency, and flexibility, these insights motivate a hybrid computing architecture that integrates both PIM and photonic tiers to leverage their complementary strengths: ❶ Photonic cores accelerate high-throughput layers, such as multi-head attention in Transformers, by performing ultra-fast matrix multiplications with minimal latency. ❷ PIM cores provide non-volatile storage (e.g., ReRAM) or high-speed SRAM arrays for tasks requiring weight-stationary operations or higher numerical precision.

Example Design: An initial exploration of heterogeneous accelerators with multiple emerging technologies introduces a multi-tier accelerator that integrates ReRAM/SRAM PIM and photonic accelerators. This hybrid architecture leverages the strengths of each computing paradigm, where photonic cores accelerate high-speed, low-precision matrix multiplications while PIM tiles efficiently handle memory-bound, weight-stationary operations with high precision and robustness. With smart workload partitioning across 3 computing tiers, we can balance the high performance of photonics for most computing tasks and use a small portion of high-precision PIM to compensate for the noise-induced accuracy drop. Evaluations on both language models (Pythia-70M [55]/TinyStories [56]) demonstrate that this hybrid design yields 2.74× better energy efficiency and 3.47× lower latency compared to homogeneous architectures or naive workload partitioning methods. By intelligently orchestrating workloads across photonic and PIM tiers, this hybrid computing platform enables an efficient and scalable hybrid AI accelerator for next-generation language and generative AI models.

## VIII. Conclusion

This paper introduced ChipMind, an LLM-driven framework that integrates specialized agents and modules for digital and analog chip design, including hardware security. By leveraging AI-driven methodologies and agentic workflows, ChipMind enhances design efficiency, accelerates prototyping, and optimizes key trade-offs in designing digital or analog circuits, facilitating intelligent synthesis optimization, test generation, and security assessment. Therefore, ChipMind efficiently addresses the growing complexity of semiconductor design, along with the growing demands for performance, power efficiency, and rapid development cycles. Furthermore, ChipMind extends conventional workflows, which rely on traditional EDA tools and manual intervention for critical tasks such as hardware description, synthesis optimization, and

verification. By mitigating inefficiencies and scalability limitations inherent in traditional approaches, ChipMind represents a novel research direction toward AI-driven, automated, and scalable semiconductor design methodologies.

## REFERENCES


[1] S. Thakur, B. Ahmad, H. Pearce, B. Tan, B. Dolan-Gavitt, R. Karri, and S. Garg, "Verigen: A large language model for Verilog code generation," *TODAES*, vol. 29, no. 3, pp. 1–31, 2024.

[2] S. Thakur, B. Ahmad, Z. Fan, H. Pearce, B. Tan, R. Karri, B. Dolan-Gavitt, and S. Garg, "Benchmarking large language models for automated verilog RTL code generation," in *DATE*, 2023, pp. 1–6.

[3] M. Liu, T.-D. Ene, R. Kirby, C. Cheng, N. Pinckney, R. Liang, J. Alben, H. Anand, S. Banerjee, I. Bayraktaroglu *et al.*, "Chipnemo: Domain-adapted llms for chip design," *arXiv preprint arXiv:2311.00176*, 2023.

[4] H. Wu, Z. He, X. Zhang, X. Yao, S. Zheng, H. Zheng, and B. Yu, "Chateda: A large language model powered autonomous agent for eda," *TCAD*, 2024.

[5] R. Zhong, X. Du, S. Kai, Z. Tang, S. Xu, H.-L. Zhen, J. Hao, Q. Xu, M. Yuan, and J. Yan, "Llm4eda: Emerging progress in large language models for electronic design automation," *arXiv preprint arXiv:2401.12224*, 2023.

[6] Y. Tsai, M. Liu, and H. Ren, "RTLFixer: Automatically fixing RTL syntax errors with large language models," *arXiv preprint arXiv:2311.16543*, 2023.

[7] B. Ahmad, S. Thakur, B. Tan, R. Karri, and H. Pearce, "On hardware security bug code fixes by prompting large language models," *TIFS*, vol. 19, pp. 4043–4057, 2024.

[8] J. Bhandari, R. Sadhukhan, P. Krishnamurthy, F. Khorrami, and R. Karri, "Sentaur: Security enhanced trojan assessment using llms against undesirable revisions," *arXiv preprint arXiv:2407.12352*, 2024.

[9] G. Kokolakis, A. Moschos, and A. D. Keromytis, "Harnessing the power of general-purpose llms in hardware trojan design," in *International Conference on Applied Cryptography and Network Security*. Springer, 2024, pp. 176–194.

[10] D. Saha, K. Yahyaei, S. Kumar Saha, M. Tehranipoor, and F. Farahmandi, "Empowering hardware security with llm: The development of a vulnerable hardware database," in *HOST*, 2024.

[11] Y. Lai, S. Lee, G. Chen, S. Poddar, M. Hu, D. Z. Pan, and P. Luo, "Analogcoder: Analog circuit design via training-free code generation," *arXiv preprint arXiv:2405.14918*, 2024.

[12] Z. Tao, Y. Shi, Y. Huo, R. Ye, Z. Li, L. Huang, C. Wu, N. Bai, Z. Yu, T.-J. Lin *et al.*, "Amsnet: Netlist dataset for ams circuits," *arXiv preprint arXiv:2405.09045*, 2024.

[13] J. Chaudhuri *et al.*, "Spiced: Syntactical bug and trojan pattern identification in a/ms circuits using llm-enhanced detection," in *IEEE Physical Assurance and Inspection of Electronics (PAINE)*, 2024.

[14] J. Chaudhuri, D. Thapar, A. Chaudhuri, F. Firouzi, and K. Chakrabarty, "Spiced+: Syntactical bug pattern identification and correction of trojans in a/ms circuits using llm-enhanced detection," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 2025.

[15] OpenAI, J. Achiam, S. Adler, and S. Agarwal, "Gpt-4 technical report," 2024. [Online]. Available: https://arxiv.org/abs/2303.08774

[16] DeepSeek-AI, D. Guo, D. Yang, H. Zhang, Z. Zhang, and Z. Zhang, "Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning," 2025. [Online]. Available: https://arxiv.org/abs/2501.12948

[17] N. Yang, T. Ge, L. Wang, B. Jiao, D. Jiang, L. Yang, R. Majumder, and F. Wei, "Inference with reference: Lossless acceleration of large language models," 2023. [Online]. Available: https://arxiv.org/abs/2304.04487

[18] X. Wang, Y. Wei, Y. Xiong, G. Huang, X. Qian, Y. Ding, M. Wang, and L. Li, "Lightseq2: accelerated training for transformer-based models on gpus," in *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, ser. SC '22. IEEE Press, 2022.

[19] P. Belcak and R. Wattenhofer, "Exponentially faster language modelling," 2023. [Online]. Available: https://arxiv.org/abs/2311.10770

[20] H. Jang, J. Kim, J.-E. Jo, J. Lee, and J. Kim, "Mnnfast: A fast and scalable system architecture for memory-augmented neural networks," in *2019 ACM/IEEE 46th Annual International Symposium on Computer Architecture (ISCA)*, 2019, pp. 250–263.

[21] S. Lu, M. Wang, S. Liang, J. Lin, and Z. Wang, "Hardware accelerator for multi-head attention and position-wise feed-forward in the transformer," in *2020 IEEE 33rd International System-on-Chip Conference (SOCC)*, 2020, pp. 84–89.

[22] B. Li, S. Pandey, H. Fang, Y. Lyv, J. Li, J. Chen, M. Xie, L. Wan, H. Liu, and C. Ding, "Ftrans: energy-efficient acceleration of transformers using fpga," in *Proceedings of the ACM/IEEE International Symposium on Low Power Electronics and Design*, ser. ISLPED '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 175–180. [Online]. Available: https://doi.org/10.1145/3370748.3406567

[23] I. Okubo, K. Sugiura, and H. Matsutani, "A cost-efficient fpga-based cnn-transformer using neural ode," *IEEE Access*, vol. 12, pp. 155 773–155 788, 2024.

[24] S. Hur, S. Na, D. Kwon, J. Kim, A. Boutros, E. Nurvitadhi, and J. Kim, "A fast and flexible fpga-based accelerator for natural language processing neural networks," *ACM Trans. Archit. Code Optim.*, vol. 20, no. 1, Feb. 2023. [Online]. Available: https://doi.org/10.1145/3564606

[25] N. P. Jouppi, C. Young, N. Patil, E. Wilcox, and D. H. Yoon, "In-datacenter performance analysis of a tensor processing unit," in *Proceedings of the 44th Annual International Symposium on Computer Architecture*, ser. ISCA '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 1–12. [Online]. Available: https://doi.org/10.1145/3079856.3080246

[26] C. Zhang, B. Sun, X. Yu, Z. Xie, W. Zheng, K. A. Iskra, P. Beckman, and D. Tao, "Benchmarking and in-depth performance study of large language models on habana gaudi processors," in *Proceedings of the SC '23 Workshops of the International Conference on High Performance Computing, Network, Storage, and Analysis*, ser. SC-W '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 1759–1766. [Online]. Available: https://doi.org/10.1145/3624062.3624257

[27] Y. Shen, N. C. Harris, S. Skirlo *et al.*, "Deep learning with coherent nanophotonic circuits," *Nature Photonics*, 2017.

[28] J. Gu, Z. Zhao, C. Feng *et al.*, "Towards area-efficient optical neural networks: an FFT-based architecture," in *IEEE/ACM Asia and South Pacific Design Automation Conference (ASPDAC)*, 2020.

[29] C. Feng, J. Gu, H. Zhu, Z. Ying, Z. Zhao, D. Z. Pan, and R. T. Chen, "A compact butterfly-style silicon photonic-electronic neural chip for hardware-efficient deep learning," *ACS Photonics*, vol. 9, no. 12, pp. 3906–3916, 2022.

[30] H. Zhu, J. Gu, H. Wang, R. Tang, Z. Zhang, C. Feng, S. Han, R. T. Chen, and D. Z. Pan, "Lightening-Transformer: A Dynamically-operated Optically-interconnected Photonic Transformer Accelerator," *arXiv preprint arXiv:2305.19533*, 2023. [Online]. Available: https://arxiv.org/abs/2305.19533

[31] A. N. Tait, T. F. de Lima, E. Zhou *et al.*, "Neuromorphic photonic networks using silicon photonic weight banks," *Sci. Rep.*, 2017.

[32] W. Liu, W. Liu, Y. Ye, Q. Lou, Y. Xie, and L. Jiang, "Holylight: A nanophotonic accelerator for deep learning in data centers," in *IEEE/ACM Proceedings Design, Automation and Test in Europe (DATE)*, 2019.

[33] F. Zokaee, Q. Lou, N. Youngblood *et al.*, "LightBulb: A Photonic-Nonvolatile-Memory-based Accelerator for Binarized Convolutional Neural Networks," in *IEEE/ACM Proceedings Design, Automation and Test in Europe (DATE)*, 2020.

[34] J. Gu, Z. Zhao, C. Feng *et al.*, "Towards Hardware-Efficient Optical Neural Networks: Beyond FFT Architecture via Joint Learnability," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, 2020.

[35] H. Zhu, J. Gu, H. Wang, Z. Jiang, Z. Zhang, R. Tang, C. Feng, S. Han, R. T. Chen, and D. Z. Pan, "Lightening-transformer: A dynamically-operated optically-interconnected photonic transformer accelerator," in *2024 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, 2024, pp. 686–703.

[36] M. Zhang, D. Yin, N. Gangi, A. Begović, A. Chen, Z. R. Huang, and J. Gu, "Tempo: Efficient time-multiplexed dynamic photonic tensor core for edge ai with compact slow-light electro-optic modulator," *Journal of Applied Physics*, vol. 135, no. 22, p. 223105, 06 2024. [Online]. Available: https://doi.org/10.1063/5.0203036

[37] M. Anderson, S.-Y. Ma, T. Wang, L. Wright, and P. McMahon, "Optical transformers," *Transactions on Machine Learning Research*, 2024. [Online]. Available: https://openreview.net/forum?id=Xxw0edFFQC

[38] S. Afifi, F. Sunny, M. Nikdast, and S. Pasricha, "Tron: Transformer neural network acceleration with non-coherent silicon photonics," in *Proceedings of the Great Lakes Symposium on VLSI 2023*, ser. GLSVLSI '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 15–21. [Online]. Available: https://doi.org/10.1145/3583781.3590259

[39] J. Pan, G. Zhou, C.-C. Chang, I. Jacobson, J. Hu, and Y. Chen, "A survey of research in large language models for electronic design automation," *ACM Transactions on Design Automation of Electronic Systems*, 2025.

[40] Y. Lai, S. Lee, G. Chen, S. Poddar, M. Hu, D. Z. Pan, and P. Luo, "AnalogCoder: Analog circuit design via training-free code generation," in *The 39th Annual AAAI Conference on Artificial Intelligence (AAAI)*. AAAI, 2025.

[41] J. Gao, W. Cao, J. Yang, and X. Zhang, "AnalogGenie: A generative engine for automatic discovery of analog circuit topologies," in *The Thirteenth International Conference on Learning Representations*, 2025. [Online]. Available: https://openreview.net/forum?id=jCPak79Kev

[42] A. F. Budak, P. Bhansali, B. Liu, N. Sun, D. Z. Pan, and C. V. Kashyap, "Dnn-opt: An rl inspired optimization for analog circuit sizing using deep neural networks," in *2021 58th ACM/IEEE Design Automation Conference (DAC)*, 2021, pp. 1219–1224.

[43] S. Poddar, A. Budak, L. Zhao, C.-H. Hsu, S. Maji, K. Zhu, Y. Jia, and D. Z. Pan, "A data-driven analog circuit synthesizer with automatic topology selection and sizing," in *2024 Design, Automation Test in Europe Conference Exhibition (DATE)*, 2024, pp. 1–6.

[44] Y. Yin, Y. Wang, B. Xu, and P. Li, "ADO-LLM: Analog design bayesian optimization with in-context learning of large language models," *arXiv preprint arXiv:2406.18770*, 2024.

[45] H. Chen, M. Liu, B. Xu, K. Zhu, X. Tang, S. Li, Y. Lin, N. Sun, and D. Z. Pan, "MAGICAL: An open-source fully automated analog ic layout system from netlist to gdsii," *IEEE Design & Test*, vol. 38, no. 2, pp. 19–26, 2020.

[46] B. Liu, H. Zhang, X. Gao, Z. Kong, X. Tang, Y. Lin, R. Wang, and R. Huang, "LayoutCopilot: An LLM-powered multi-agent collaborative framework for interactive analog layout design," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2025.

[47] X. Xu, M. Tan, B. Corcoran, J. Wu, A. Boes, T. G. Nguyen, S. T. Chu, B. E. Little, D. G. Hicks, R. Morandotti, A. Mitchell, and D. J. Moss, "11 TOPS photonic convolutional accelerator for optical neural networks," *Nature*, 2021.

[48] J. Gu, Z. Zhao, C. Feng, H. Zhu, R. T. Chen, and D. Z. Pan, "ROQ: A noise-aware quantization scheme towards robust optical neural networks with low-bit controls," in *IEEE/ACM Proceedings Design, Automation and Test in Europe (DATE)*, 2020.

[49] Z. Zhao, J. Gu, Z. Ying *et al.*, "Design technology for scalable and robust photonic integrated circuits," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2019.

[50] H. Lu, S. Banerjee, and J. Gu, "Doctor: Dynamic on-chip temporal variation remediation toward self-corrected photonic tensor accelerators," *IEEE Journal of Lightwave Technology*, pp. 1–9, 2024.

[51] A. Shafiee, A. Nag, N. Muralimanohar, R. Balasubramonian, J. P. Strachan, M. Hu, R. S. Williams, and V. Srikumar, "Isaac: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars," *ACM SIGARCH Computer Architecture News*, vol. 44, no. 3, pp. 14–26, 2016.

[52] B. Feinberg, S. Wang, and E. Ipek, "Making memristive neural network accelerators reliable," in *2018 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 2018, pp. 52–65.

[53] A. Yusuf, T. Adegbija, and D. Gajaria, "Domain-specific stt-mram-based in-memory computing: A survey," *IEEE Access*, 2024.

[54] J. Zhang, Z. Wang, and N. Verma, "In-memory computation of a machine-learning classifier in a standard 6t sram array," *IEEE Journal of Solid-State Circuits*, vol. 52, no. 4, pp. 915–924, 2017.

[55] S. Biderman, H. Schoelkopf, Q. G. Anthony, H. Bradley, K. O'Brien, E. Hallahan, M. A. Khan, S. Purohit, U. S. Prashanth, E. Raff *et al.*, "Pythia: A suite for analyzing large language models across training and scaling," in *International Conference on Machine Learning*. PMLR, 2023, pp. 2397–2430.

[56] R. Eldan and Y. Li, "Tinystories: How small can language models be and still speak coherent english?" *arXiv preprint arXiv:2305.07759*, 2023.