# Hardware Design and Security Needs Attention: From Survey to Path Forward

Sujan Ghimire[1], Muhtasim Alam Chowdhury[2], Banafsheh Saber Latibari[2], Muntasir Mamun[1], Jaeden Wolf Carpenter[2], Benjamin Tan[3], Hammond Pearce[4], Krishnendu Chakrabarty[5], Pratik Satam[1], and Soheil Salehi[2]



***Abstract*—Recent advances in attention-based artificial intelligence (AI) models have unlocked vast potential to automate digital hardware design while enhancing and strengthening security measures against various threats. This rapidly emerging field leverages Large Language Models (LLMs) to generate HDL code, identify vulnerabilities, and sometimes mitigate them. The state of the art in this design automation space utilizes optimized LLMs with HDL datasets, creating automated systems for register transfer level(RTL) generation, verification, and debugging and establishing LLM-driven design environments for streamlined logic designs. Additionally, attention-based models like graph attention have shown promise in chip design applications, including floorplanning. This survey investigates the integration of these models into hardware-related domains, emphasizing logic design and hardware security, with or without the use of IP libraries. This study explores the commercial and academic landscape, highlighting technical hurdles and future prospects for automating hardware design and security. Moreover, it provides new insights into the study of LLM-driven design systems, advances in hardware security mechanisms, and the impact of influential works on industry practices. Through the examination of 30 representative approaches and illustrative case studies, this paper underscores the transformative potential of attention-based models in revolutionizing hardware design while addressing the challenges that lie ahead in this interdisciplinary domain.**

***Index Terms*—Attention, Transformer, Large Language Models (LLMs), Electronic Design Automation (EDA), Hardware Design, Hardware Security.**

## I. Introduction

HARDWARE design is a complex process that begins with specific requirements outlined in natural language. It involves hardware engineers translating these requirements into Hardware Description Languages (HDLs) before verifying them extensively [1], [2]. During this process, hardware engineers must ensure that design constraints (such as performance, power, and area) are met, which can be repetitive and prone to errors and inefficiencies. Automating these processes with machine learning (ML) models, such as recent deep neural networks featuring the *attention* mechanism [3], can significantly reduce the likelihood of human error and accelerate the design cycle by streamlining tasks and improving design efficiency. Initially, the attention-based model was introduced as an alternative to recurrent neural networks (RNNs) for natural language processing (NLP) [3]. The transformer model, which uses self-attention to focus on different parts of input data to make predictions, has led to pioneering outcomes in many domains like speech recognition, computer vision, sentiment analysis, named entity recognition, and time series analysis [4]–[6]. LLMs such as the Generative Pre-trained Transformer (GPT) series have demonstrated remarkable capabilities in generating human-like text, understanding context, and performing complex language-related tasks [7].

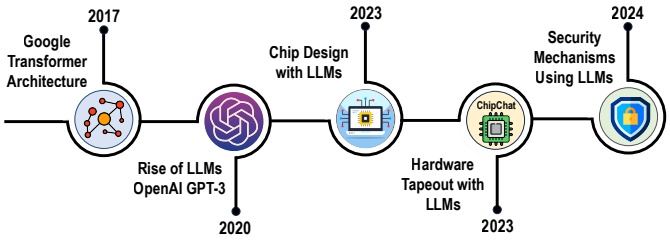


Fig. 1. Timeline of AI advancements and their integration into hardware design and security from 2017 to 2024.

Figure 1 shows the evolution of key advancements in AI and hardware design, from the introduction of google transformer architecture (2017) to the rise of LLMs like GPT-3 (2020), their application in chip design (2023), the first hardware tapeout of hardware designed with LLMs (2023), and advancements in security mechanisms using LLMs (2024).

In the realm of hardware design and security, the application of attention-based models presents a transformative opportunity still in its early stages of exploration. Figure 2 shows the projection for the number of publications through 2025. The numbers presented in this figure are derived from an extensive survey of 117 references included in this paper. These references encompass studies on applying attention-based models in hardware design and security. The trajectory in the figure was determined by categorizing and analyzing the publication trends of the surveyed resources, reflecting the growing interest and progress in this interdisciplinary domain. Specifically, the prediction for 2025, where the number of articles is expected to increase to 345, was extrapolated based on the observed growth pattern, which increased fourfold from 21 articles in 2023 to 86 in 2024, indicating a compounding

[1]Systems and Industrial Engineering Department, University of Arizona, Tucson, AZ, USA (e-mail: sghimire@arizona.edu, muntasir@arizona.edu, pratiksatam@arizona.edu).

[2]Electrical and Computer Engineering Department, University of Arizona, Tucson, AZ, USA (e-mail: mmc7@arizona.edu, banafsheh@arizona.edu, carpenterjaeden@arizona.edu, ssalehi@arizona.edu).

[3]Department of Electrical and Software Engineering, University of Calgary, Calgary, Alberta, CA (e-mail: benjamin.tan1@ucalgary.ca).

[4]School of Computer Science and Engineering, University of New South Wales, Sydney, AU (e-mail: hammond.pearce@unsw.edu.au).

[5]School of Electrical Computer and Energy Engineering, Arizona State University, Tempe, AZ, USA (e-mail: Krishnendu.Chakrabarty@asu.edu).

trend.

Recent work has explored the potential of LLMs for automated HDL generations from instructions in Natural Language (known as 'prompts') [8]. Although this showed potential for speeding up the design process (including successful tape-out) by prompting LLMs, significant human intervention was still required, as pre-trained attention-based models could not handle all the tasks necessary for a complete tape-out design without errors.

Overcoming this performance gap between LLMs and humans is not trivial. One of the significant challenges here is the relative scarcity of public HDL code bases usable for LLM fine-tuning. Notable initiatives include optimizing previously pretrained LLMs using Verilog datasets, creating automated frameworks such as Autochip [9] for bug correction in LLM-generated HDL code, and creating LLM-based design environments. These initiatives represent initial steps in adapting LLMs to the hardware engineering landscape, focusing on logic design optimization through natural language interaction [9]. Other research using attention-based models for the rest of the chip design stages has also faced the challenge of limited data.

To address the aforementioned limitations and make the semiconductor industry more efficient, significant interest and support have been directed toward this research area. Figure 3 shows a snapshot of some of the funding resources supporting this research. To compile this data, we targeted the references that are cited in this paper to identify the key funding agencies supporting research in attention-based hardware design and security. The data reflects a total of 38 funding instances across various research areas, with notable contributions from leading organizations such as NSF, NSFC, SRC, and DARPA. Furthermore, the funding distribution underscores the growing research interest in attention-based hardware design and security, reflecting its increasing significance in shaping future advancements in the field.

In recent work, Bei Yu summarized the application of ML techniques in Electronic Design Automation (EDA) [10]. He highlighted their impact on various stages of chip design and proposed an adaptive integrated ML paradigm to address challenges such as netlist representation, timing modeling, and multimodal fusion. LLM4EDA provided a systematic study on

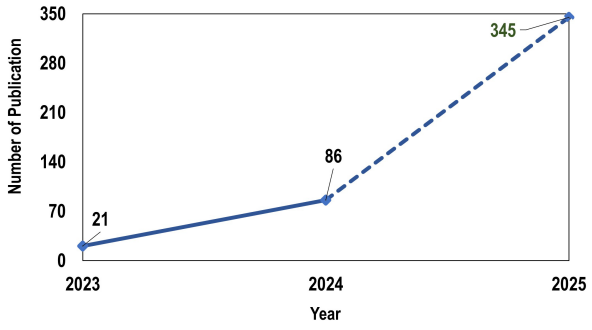


Fig. 2. Projected growth in research publications on LLM-based hardware design and hardware security, highlighting a nearly 4x increase from 2023 to 2024, with an estimated surge in 2025 following the same growth pattern.

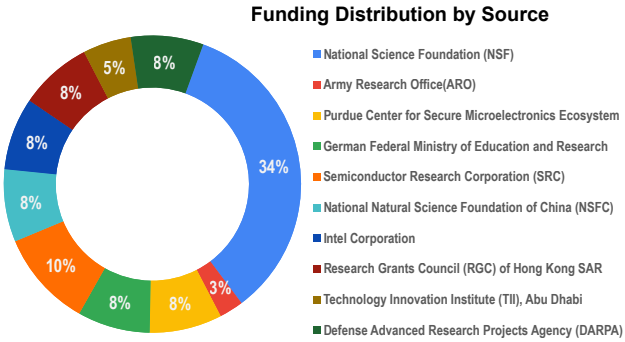


Fig. 3. Funding distribution across various research areas and organizations, illustrating the contributions of multiple funding sources. The chart highlights the diversity and extent of support provided to hardware design, verification, security, and optimization research.

the application of LLMs in EDA, exploring their potential to automate various stages of chip design and improve power, performance, and area (PPA) metrics while highlighting future research directions and maintaining an open-source repository of relevant papers and datasets [11]. Tsung-Yi Ho et al. explored the development and application of AI-native large circuit models (LCMs) in EDA, highlighting their potential to revolutionize the field through comprehensive analysis and creation capabilities while addressing challenges such as data scarcity and scalability [12]. Sharma et al. conducted a thorough analysis of the total cost of ownership (TCO) and performance between ChipNeMo, a domain-adaptive LLM, and leading general-purpose LLMs (Claude 3 Opus and ChatGPT-4 Turbo), demonstrating significant cost savings and performance benefits for ChipNeMo in chip design coding tasks [13]. Nan Wu et al. provided a comprehensive overview of ML methodologies applied to hardware design and verification, identifying challenges and proposing solutions within EDA, highlighting the potential and limitations of ML in enhancing design efficiency and accuracy [14]. Yang et al. presented an in-depth review of LLMs, focusing on their applications, capabilities, challenges, and prospects [15]. It explores evaluation metrics like BLEU scores and human-aligned evaluations, emphasizes the importance of prompt engineering for LLM efficacy, compares fine-tuning with prompt tuning, discusses scalability and generalization capabilities, and addresses ethical concerns such as bias and misinformation.

An emerging challenge in hardware design is the need to incorporate security. Akyash et al. explored LLMs in hardware security, detailing their use for RTL vulnerability detection and mitigation, evaluation frameworks, comparative analyses, case studies, and future needs for specialized architectures, improved datasets, and EDA tool integration [16]. Similarly, Pearce et al. argue for the potential of LLMs in this space, highlighting successes already obtained by LLMs working in limited hardware security tasks and scenarios [17].

This paper offers a complementary extension to prior surveys. In considering all types of models based on attention mechanisms, we do not limit our survey purely to LLMs, and we survey the use of these models in all design stages, from RTL generation and verification to layout design and

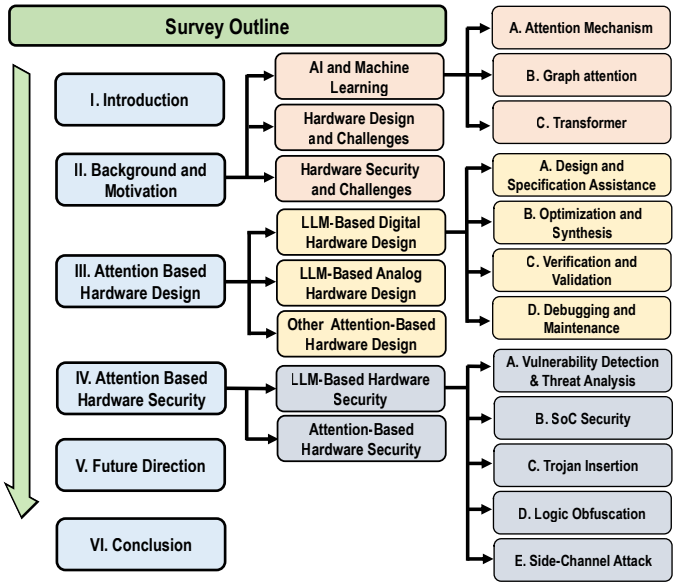


Fig. 4. Overview of the survey in this manuscript.

routing. Further, we provide insights into both hardware *design* and hardware *security*, such as Trojan insertion and logic obfuscation.

Figure 4 shows the organization of this paper. Section II provides background on ML models used in the literature. Moreover, the limitations and challenges that make using these attention-based models necessary in hardware design and security are discussed. Sections III and IV summarize the attention-based hardware design and security, respectively. We talk about the future directions of the research in Section V, and Section VI concludes the paper.

## II. Background and Motivation

In this section, we provide a comprehensive background on the evolution of AI and ML, with a particular focus on the transformative advancements in neural network architectures. We begin by introducing the foundational concepts of AI and ML, followed by a detailed discussion on graph attention mechanisms and the revolutionary transformer model, highlighting their development and impact over the years. Furthermore, we offer an in-depth overview of these concepts, emphasizing their significance in modern computational paradigms. Subsequently, we transition to the domain of hardware design, exploring the challenges associated with it. Finally, we address the critical aspect of hardware security, examining the unique vulnerabilities and challenges in hardware design.

### A. AI and Machine Learning

*1) Attention Mechanism:* The attention mechanism is a neural network structure engineered to concentrate on the most relevant aspects of the input data during prediction. Bahdanau et al. (2015) initially developed it in neural machine translation to overcome the shortcomings of conventional models such as Recurrent Neural Networks (RNNs), which struggle to capture long-range relationships and manage variable-length sequences [18]. The core principle of attention is computing a weighted sum of input items, where the weights reflect the importance of each input component. This enables the model to focus on critical elements of the input, enhancing its ability to comprehend complex data. Various types of attention processes are available, such as global attention, which considers the whole input sequence, and local attention, which concentrates on certain portions of the sequence. Self-attention enables each element in the input to focus on all other elements, constituting a fundamental aspect of transformer models. Furthermore, multi-head attention utilizes many attention processes concurrently, allowing the model to discern several facets of the input. Figure 5-(a) illustrates the attention mechanism focusing on the scaled dot-product attention. The mechanism begins with the inputs, queries ($Q$), keys ($K$), and values ($V$). Then, it requires computing the similarity in scores between the query and keys via matrix multiplication ($QK^T$), which is then scaled by the square root of the key dimensions to stabilize gradients during training. Masking can be applied to selectively ignore specific parts of the input, such as future tokens, in sequence prediction tasks. To produce attention weights, the similarity scores are normalized by using the softmax function. This also highlights the importance of each key relative to the query. The Values ($V$) are then combined with the weighted sum, which is guided by these attention weights. The Multi-head attention model works like an extension by running multiple attention computations in parallel, capturing diverse contextual relationships within the system.

*2) Graph Attention:* Graph Attention Networks (GATs) are a neural network architecture specifically engineered for graph-structured data, employing the attention mechanism to overcome the shortcomings of previous graph neural networks (GNNs) [19]. GATs utilize self-attention layers rather than fixed convolution procedures, enabling each node to concentrate on its most pertinent neighbors selectively. This is accomplished by calculating attention scores that signify the significance of adjacent nodes according to their attributes. The model subsequently consolidates these attributes, weighted by the attention ratings, to enhance the node's representation. GATs do not require the complete graph structure in advance and may accommodate variable-sized neighborhoods, rendering them quite adaptable. Moreover, multi-head attention captures many dimensions of node interactions, augmenting the network's ability to understand intricate patterns. GATs have demonstrated robust efficacy in node categorization, connection prediction, and community discovery across several domains, including social networks, biological networks, and citation graphs. Furthermore, GATs are useful for assessing circuit interdependence and finding possible flaws. By conceptualizing circuit parts (e.g., gates, flip-flops) as nodes and their interactions as edges, GATs can model whole circuits.

Figure 5-(b) illustrates the overview of the graph attention mechanism, which is the heart of GATs. The figure showcases how each node computes attention scores to focus on their most relevant neighbors by employing this attention mechanism. It does this specifically by computing attention coefficients ($\alpha$), which signifies the importance of each neighboring

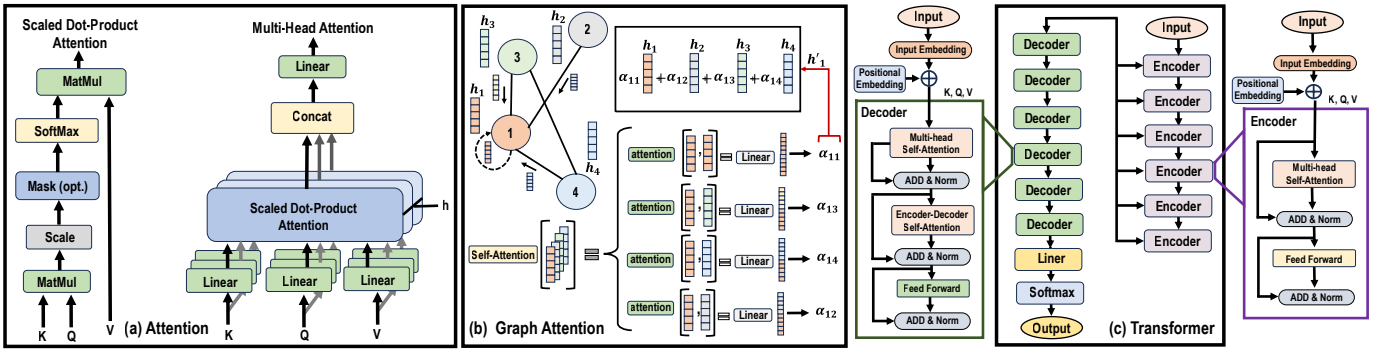


Fig. 5. Illustration of the core mechanisms of modern deep learning frameworks: (a) Attention, including scaled dot-product and multi-head attention (b) Graph Attention for processing graph-based data, and (c) Transformer architecture with encoder-decoder structure.

node based on features $(h_1, h_2, h_3, ...)$. These coefficients are also used in generating updated representation $(h'_1)$ for the target node. As shown in this figure 5-(b), the process also involves the multiple attention heads that compute various perspectives of node interactions. This approach enhances the model's ability by identifying the complex patterns in graph-structured data.

*3) Transformer:* The transformer is a neural network architecture that uses self-attention mechanisms, eliminating traditional recurrent and convolutional layers. It consists of an encoder and a decoder, each built with layers of self-attention and feed-forward networks. The self-attention mechanism allows the model to capture relationships across entire input sequences in a highly parallelizable manner, improving computational efficiency. Multi-head attention enables the model to focus on multiple input segments simultaneously, enhancing tasks like machine translation and text generation. The encoder-decoder transformer is ideal for sequence-to-sequence applications, such as machine translation. The encoder in this architecture converts the input sequence into a context vector, which the decoder uses to generate the output sequence. Encoder-based transformers, such as BERT [20], are designed for tasks requiring an understanding of entire input sequences, like text classification and entity recognition. Decoder-based transformers, such as GPT, specialize in sequence generation, predicting output tokens sequentially in an autoregressive manner by analyzing relationships among previously generated tokens.

The architecture, illustrated in Figure 5-(c), consists of encoder and decoder modules with self-attention and feed-forward layers. The encoder captures contextual relationships within input sequences, while the decoder generates output sequences autoregressive. Transformers architecture also supports hardware design and security by enabling efficient sequence modeling, dependency extraction, circuit representation learning, Verilog code generation, vulnerability identification, and mitigation generation in hardware designs.

## B. Hardware Design and Challenges

The hardware design flow is a systematic process that converts high-level design specifications into a functioning integrated circuit ready for implementation, as shown in Figure 6. The process starts by delineating functional and performance criteria in the design specification [1], [2]. The design team incorporates third-party intellectual property (IP) blocks to accelerate development. The design is captured at the RTL using HDLs such as Verilog or VHDL, emphasizing functionality. The RTL description undergoes synthesis into a gate-level netlist through logic synthesis, which translates the design into fundamental logic gates while optimizing for performance, power, and area. This netlist undergoes physical synthesis, encompassing placement, and routing, to produce a comprehensive physical layout. The design is manufactured onto silicon wafers using sophisticated semiconductor techniques, such as photolithography, ion implantation, and chemical vapor deposition, to precisely pattern and build the intricate layers of the integrated circuit. Upon fabrication, the wafers are encased for protection and seamless integration, followed by stringent testing to verify compliance with functional and performance standards. The design team delivers the evaluated integrated circuits as the final product for incorporation into electronic systems. This rigorous process ensures the attainment of design objectives while minimizing mistakes and inefficiencies.

The advancement in hardware design is progressing exponentially, leading to significant and unpredictable challenges that require continuous attention. Figure 7 illustrates the key challenges in the hardware design process, which include power efficiency, scalability, performance optimization, signal integrity, timing analysis, verification and validation, physical design constraints, and time-to-market pressures. Modern Integrated Circuits (ICs) integrate billions of transistors with intricate interdependencies, making seamless functionality and scalability critical challenges. This increasing complexity has driven the development of System-on-Chip (SoC) architectures [21], which address these challenges by compacting designs and enhancing performance. However, SoCs introduce new difficulties in verification and validation, as traditional methods struggle to handle the vast number of components and interactions within a single chip [10], [22]. Power management is a critical focus, especially with the proliferation of IoT and mobile devices. Designers must innovate to reduce power consumption and manage heat dissipation without compromising performance [23]. Performance optimization remains a

primary goal in applications such as AI accelerators and high-speed communication systems, where achieving high-speed operations, low latency, and superior throughput are essential [24]. Signal integrity and timing analysis are equally vital. As signal paths shrink and frequencies rise in advanced nodes, ensuring reliable data transmission and synchronization has become increasingly challenging. Static Timing Analysis (STA) is commonly employed, but its accuracy is often hindered by the growing complexity and miniaturization of components [25]–[27]. Physical design challenges, such as floor planning, layout optimization, and routing, further complicate the hardware development process. Efficiently arranging chip components with proper routing to balance performance, power consumption, and thermal management is critical [28] and necessitates advanced algorithms and tools. Verification and validation have become increasingly resource-intensive and time-consuming, demanding advanced techniques to ensure all modules adhere to specifications and function as intended. Physical constraints in design also impact manufacturability and yield, requiring careful optimization of layouts and static timing. Time-to-market pressures add to these challenges, as delays in addressing any issue can lead to significant market disadvantages [29].

ML can change traditional hardware design by increasing efficiency and accuracy at each stage of the workflow and by helping to solve challenges such as scalability, optimization, and decision-making that otherwise involve considerable manual effort and computational resources. The selection and integration of third-party IPs can be optimized through ML models in the design specification and integration stage using compatibility and performance metrics prediction. Predictive ML techniques, for instance, analyze design requirements to provide suitable recommendations for IPs by reducing integration errors and improving design efficiency. Tools based on ML, such as ChipGPT [30], already help designers at the RTL design stage with power-performance-area-optimized Verilog code. Similarly, ML models have been trained on hardware datasets to detect and repair logical errors in HDL designs, which helps enhance design reliability and scalability. ML algorithms enhance optimization during logic synthesis and gate-level netlist generation by translating HDL into efficient gate-level netlists. In this respect, reinforcement learning methods optimize adaptive constraints for balanced timing, power, and area requirements [31]. ML significantly improves physical synthesis and layout, especially for placement and routing. Gao et al. demonstrated using graph-based neural networks to predict congestion hotspots and optimize wire length, reducing timing violations and ensuring manufacturability [32]. Additionally, convolutional neural networks (CNNs) accelerate power grid performance prediction, offering faster evaluations than traditional methods. Additionally, ML can be to testing and validation to automate test pattern generation and improve fault detection. Advanced ML models analyze test data to predict future failures, optimizing overall testing and fault coverage. These ML-driven improvements meet demands for complexity, scalability, and performance in modern hardware design with significantly reduced manual effort throughout the flow.

### C. Hardware Security and Challenges

The evolution of hardware in recent years has introduced numerous security challenges that demand innovative solutions. Significant hardware challenges prevalent today include Trojans, IP theft, side-channel attacks, fault injection attacks, reverse engineering, and supply chain attacks [33]–[36]. The globalization of the hardware supply chain has exacerbated these problems, necessitating robust detection and mitigation solutions [37]. Hardware Trojans are malicious and stealthy circuit alterations that compromise device integrity and functionality. The insertion of hardware Trojans is one of the most significant security challenges in hardware security, reported as one of the stealthiest forms of cyber attacks [25]. Another critical attack is the side-channel attack, where attackers exploit circuit leakages, such as power consumption and EM emissions, to extract sensitive information from the system [38]. Fault injection attacks, where faults are deliberately induced in the circuit, represent another form of hardware attack, leading to data corruption and unauthorized access to the device's data and systems [39]. Additionally, ICs are vulnerable to reverse engineering, which involves analyzing the IC to uncover its design and functions, preventing unauthorized replications, and identifying design security vulnerabilities [9].

Figure 8 outlines key hardware security threats across the IC design lifecycle, including hardware Trojans, side-channel attacks, fault injection, and reverse engineering. Attackers try to exploit the IC for unintended leakages, such as power consumption or electromagnetic emissions, while the IC operates. In fault injection attacks, the attacker deliberately induces faults to corrupt data and disrupt systems. Conversely, reverse engineering attacks can apply to deployed chips. Such attacks allow the attacker to reconstruct designs and extract proprietary information from ICs during the testing and packaging stages.

ML for hardware security has emerged as a research subject and effectively mitigates complex security challenges. One key

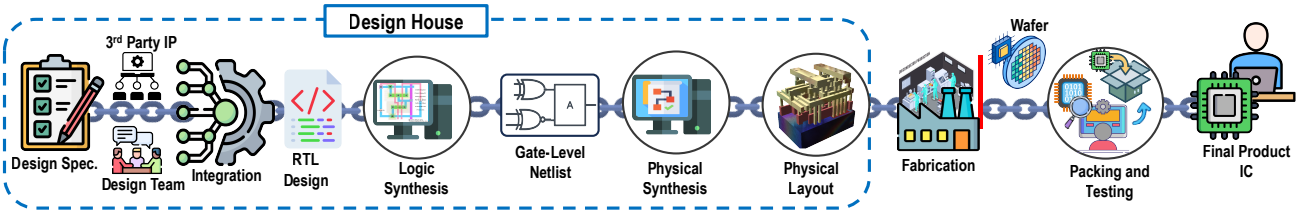


Fig. 6. Overview of the key stages in the globalized IC design flow.

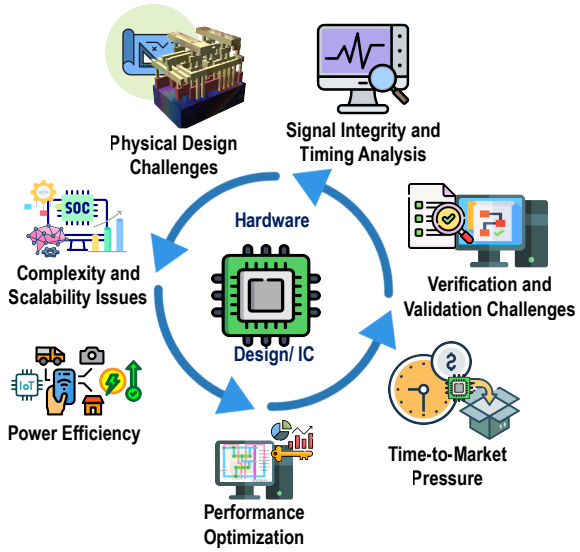


Fig. 7. Key challenges in hardware design.

area where ML techniques excel is hardware Trojan detection, with models trained to identify anomalies in circuit behavior that may indicate malicious modifications [40], [41]. Studies have shown that ML algorithms analyze power and other side-channel data to detect discrepancies caused by hardware Trojans [42]. Additionally, ML-based techniques effectively mitigate side-channel vulnerabilities by understanding and reducing leakage patterns, enhancing the system's overall security posture [43]–[46]. Logic obfuscation is another critical area in hardware security that uses ML to enhance circuit design security by introducing transformations that make reverse engineering significantly harder [25], [47]. Surveys further highlight ML's role in both physical and logic testing techniques for hardware Trojan detection [48], [49]. ML also extends its applications to fault injection attack prevention and supply chain security, providing scalable, automated, and efficient solutions to reinforce hardware security [50]–[52]. By leveraging ML's capabilities, researchers and engineers are developing robust countermeasures to protect against a wide variety of hardware security threats.

## III. Attention-Based Hardware Design

In Section II, we covered the current challenges in hardware design. Now, we will delve into the LLM-based and other attention-based solutions identified for these challenges and examine ongoing research into new technologies to provide better solutions.

### A. LLM-based Digital Hardware Design

This section explores the research that used LLMs for hardware design using various techniques. Figure 9 illustrates the integrated overview of the critical applications of LLMs in digital hardware design. It indicates the processes in the hardware design flow that benefit most from LLMs. This paper categorizes the design process into four major aspects: design and specification assistance, optimization and synthesis, verification and validation, and debugging and maintenance. Each section indicates distinct hardware stages where LLMs ensure seamless integration, automation, and enhancement across these sections. LLMs are capable of supporting tasks such as specification drafting, Verilog code generation, performance optimization, testbench generation, and error detection. This figure provides a comprehensive overview, setting the stage for a deeper examination of each domain in the subsections that follow.

*1) Design And Specification Assistance:* In this category, LLMs aid in the initial stages of hardware design by drafting design specifications and converting natural language descriptions into formal HDLs. Figure 10 illustrates the comprehensive workflow of hardware design using LLMs, demonstrating their role in automating and enhancing various stages of the IC design process. The process begins with a human prompting in natural language, and then, depending on the type of LLMs, the output files as Verilog code are generated. This process enables efficient design and specification assistance in the chip design process. The generated Verilog codes are then synthesized and proceed to fabrication, with the verification steps in between to ensure correctness and compliance with performance requirements. The iterative feedback loop between the verification and synthesis stages highlights how LLMs optimize the overall design flow by reducing errors and improving efficiency at each stage of the design timeline.

To support this, a significant body of research has focused on leveraging LLMs for Verilog code generation. Thakur et al. pioneered this area by evaluating the ability of LLMs to generate syntactically correct and functionally accurate Verilog RTL code by fine-tuning models on a large Verilog corpus and benchmarking their performance with the fine-tuned CodeGen-16B model, achieving the highest accuracy [53]. In another similar research, Thakur et al. compiled the largest corpus of Verilog code from different open-source repositories, which was then used to fine-tune an LLM. The LLM model called VeriGen was then used for HDL generation, and the results were evaluated against the advanced general-purpose LLMs regarding the correctness of the code generated. The performance of the fine-tuned model was found to be significantly improved [56]. In another research endeavor, Thakur et al. introduced Autochip, a fully automated feedback-driven verilog code generation tool that iteratively refines designs based on the compiler output [9].

Further studies have expanded the exploration of LLMs in functional HDL generation. Yang et al. further highlighted the potential of LLMs by using GPT-4 for generating functional Verilog code for complex designs like systolic arrays and AI accelerators for ResNet and MobileNet. The approach yielded results; however, it required significant manual optimization, especially for complex designs [1]. Similarly, Yanık et al. introduced ShortCircuit, a tool that automates front-end digital integrated circuit design using ChatGPT and OpenLane. ShortCircuit significantly reduced design time for ASIC and FPGA implementations by generating HDL, GDS, and bitstream files [55].In the same way, Tomlinson et al. leveraged ChatGPT to generate synthesizable and functional Verilog descriptions

TABLE I
LLM-BASED DIGITAL HARDWARE DESIGN

| Year | Method Name | Design Assistance | Optimization Synthesis | Verification Validation | Debugging Maintenance | AI Model |
|---|---|---|---|---|---|---|
| 2022 | [53] | ✓ | | ✓ | | CodeGen(2B, 6B, 16B), MegatronLM-355M, Code-davinci-002 |
| 2023 | ChipChat [8] | ✓ | | | | ChatGPT-4 |
| 2023 | VerilogEval [24] | ✓ | | ✓ | | GPT(3.5, 4), CodeGen-16B-verilog |
| 2023 | RTLLM [23] | | | ✓ | | GPT-3.5 |
| 2023 | [54] | ✓ | ✓ | ✓ | | GPT(3.5, 4) |
| 2023 | RTLFixer [39] | | | | ✓ | GPT(3.5, 4) |
| 2023 | RTLCoder 2023 [38] | | | ✓ | | Mistral-7B, DeepSeek-Coder-6.7B |
| 2023 | AutoChip [9] | ✓ | | | | GPT(3.53.5-turbo, 4), Claude 2, Code Llama 2 |
| 2023 | ShortCircuit [55] | ✓ | | | | ChatGPT |
| 2023 | VeriGen [56] | ✓ | | | | MegatronLM-355M, CodeGen(2B, 6b, 16b), J1-Large-7B, code-davinci-002, GPT(3.5-turbo, 4), PALM2 |
| 2023 | LLM4EDA [11] | | ✓ | | | GPT(3.5, 4), LLaMA2 |
| 2024 | SpecLLM [57] | ✓ | | | | GPT-4 |
| 2024 | AssertLLM [29] | | | ✓ | | GPT(3.5, 4, 4-Turbo) |
| 2024 | ChIRAAG [58] | | | ✓ | | GPT-4 Turbo |
| 2024 | [59] | | | ✓ | | VeriGen-2B (Fine-tuned version of CodeGen LLM) |
| 2024 | [60] | | ✓ | | | GPT-4 |
| 2024 | [2] | ✓ | | | | ChatGPT-4 |
| 2024 | [61] | | | ✓ | | GPT-4, LLaMA2 |
| 2024 | HDLdebugger [62] | | | | ✓ | GPT-4, RTLFixer, VeriGen, CodeLlama-13b (base model for HDLdebugger) |
| 2024 | [63] | | | | ✓ | Llama 2-13B, Llama 2-7B |
| 2024 | ChatPattern [64] | ✓ | | | | |
| 2024 | [65] | | | | ✓ | GPT(3.5-Turbo, 4) |
| 2024 | [66] | ✓ | | | | GPT(3.5, 4), CodeLlama(7B, 13B), VeriGen(6B, 16B) |
| 2024 | [67] | | | ✓ | | GPT(3.5, 4), Claude, Gemini Advanced |
| 2024 | [68] | ✓ | | ✓ | | GPT-4 |
| 2024 | [69] | | | ✓ | ✓ | ChatGPT-4 |
| 2024 | [70] | ✓ | | ✓ | | ChatGPT(3.5, 4), Bard, HuggingChat |
| 2024 | RTLCoder [71] | ✓ | | | | RTLCoder (based on Mistral-7B and DeepSeek-Coder-6.7B) |
| 2024 | [72] | ✓ | | | | CodeGen(2B, 6B, 16B), GEMMA(2B, 7B) |
| 2024 | [73] | | | | ✓ | GPT-4, Codex, CodeGen |
| 2024 | [74] | | | ✓ | | Hardware Phi-1.5B |
| 2024 | [75] | ✓ | ✓ | ✓ | | Microsoft Bing Chat |
| 2024 | [76] | ✓ | | | | Llama2-70B |
| 2024 | [77] | | | ✓ | | GPT(4, 4V), LLaVA, LLaMA |
| 2024 | [78] | | | ✓ | ✓ | GPT-3.5, CodeLlama, DeepSeekCoder, CodeQwen |
| 2024 | [79] | ✓ | ✓ | | | GPT(3.5, 4) |
| 2024 | [80] | | | ✓ | | GPT(3.5, 4, 4o) |
| 2024 | [81] | ✓ | | | | AutoMage (fine-tuned on Llama2-70B), GPT(3.5, 4) |
| 2024 | [82] | | ✓ | | | GPT-3.5-turbo-16k-0613 |
| 2024 | [83] | ✓ | | ✓ | | Codellama-7B, DeepSeek-Coder-6.7B, CodeQwen1.5-7B |
| 2024 | [84] | ✓ | | | | Llama 2, Code Llama, VeriGen, CL-Verilog, RTL-Coder, Llama 3, GPT(3.5, 4) |

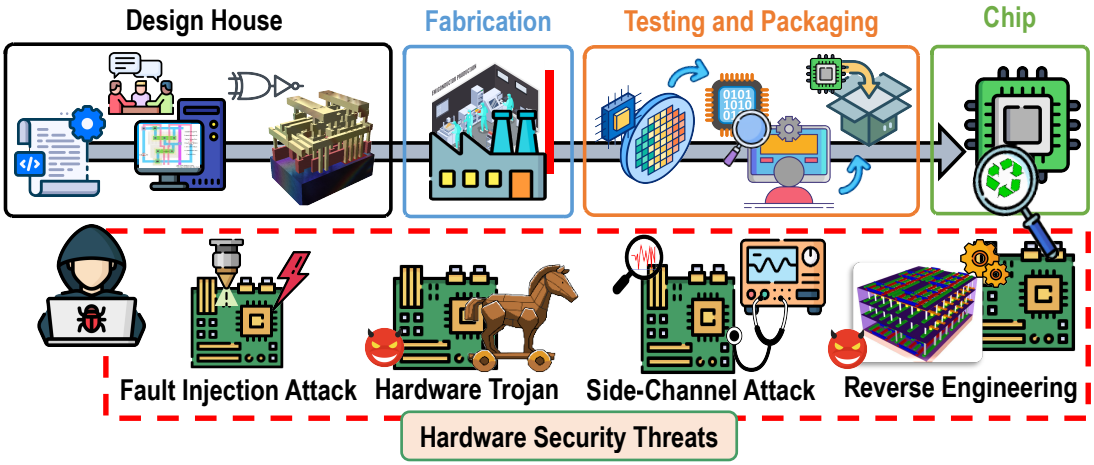


Fig. 8. Overview of key hardware security threats and challenges in modern hardware design.

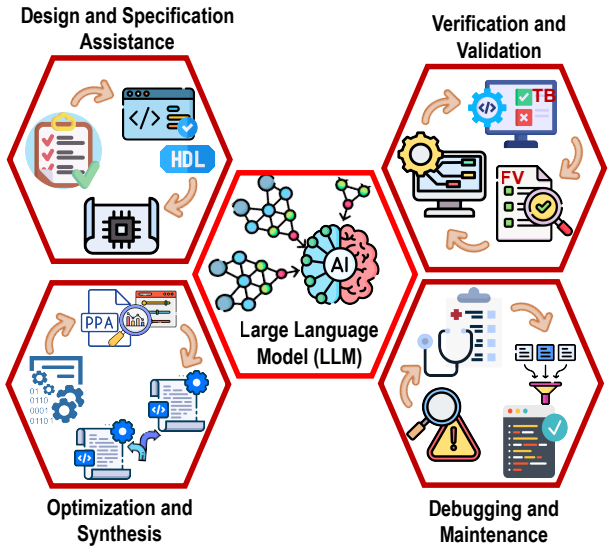


Fig. 9. Overview of the critical applications of LLMs in digital hardware design, encompassing specification drafting, optimization, verification, and debugging to streamline the design lifecycle. Each application category includes key functionalities such as specification drafting, PPA optimization, testbench generation, and error correction

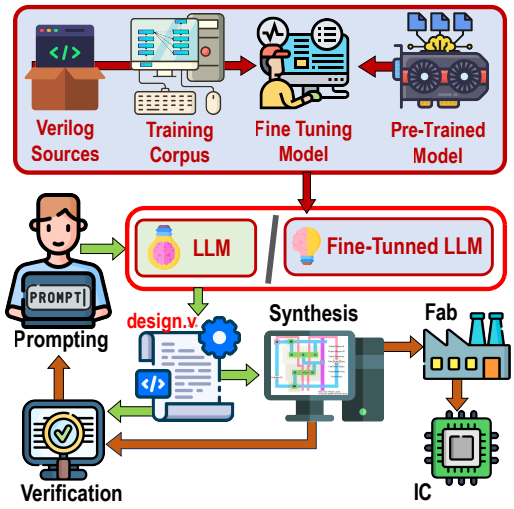


Fig. 10. Workflow of hardware design using LLMs, highlighting how both prompting with pre-trained models and fine-tuning with Verilog sources and training data enable the complete hardware design process.

for a programmable Spiking Neuron Array ASIC through natural language prompts, validating the design through simulation and preparing it for fabrication using Skywater 130nm technology [2]. Efforts have also focused on enhancing dataset quality and benchmarking. Chang et al., in the "Data is all you need," paper, proposed an automated data augmentation framework to generate high-quality verilog and EDA script generation. By converting Verilog files into abstract syntax trees, applying predefined repair rules, and leveraging GPT-3.5 for EDA script descriptions, the framework generates high-quality datasets. When the Llama 2-7B and Llama 2-13B models were fine-tuned with these data, a significant improvement in accuracy was achieved. The author used the Thakur et al. benchmark and achieved an 11. 8% improvement in the pass rate compared to Thakur et al. [63].

Expanding beyond single-model fine-tuning, Several studies have focused on fine-tuning and customizing LLMs for hardware design tasks. Liu et al. presented a lightweight, customized, and fine-tuned LLM called RTLCoder that outperforms GPT-3.5. They quantized the RTLCoder to 4 bits, making the model lightweight and compatible with running on a normal laptop as a local assistant to hardware engineers [71].In parallel, Nadimi and Zheng introduced the Multi-Expert Verilog LLM (MEV-LLM) architecture, which leveraged multiple LLMs fine-tuned with datasets categorized by design complexity to enhance Verilog code generation [72]. By developing a categorized dataset with fine-grained annotations, the MEV-LLM shows up to a 23.9% improvement in generating syntactically and functionally correct Verilog code compared to state-of-the-art models. Evaluated on tasks from the HDLBits website, the architecture demonstrates significant performance gains, emphasizing the effectiveness of multi-expert models in addressing various design complexities. A

similar goal was pursued by Gao et al. through AutoVCoder, which implemented a retrieval-augmented generation (RAG) mechanism for enhancing the syntactic and functional correctness of Verilog code generation by LLMs [83]. AutoVCoder outperformed existing LLMs, showing significant improvements in syntax and functional correctness across multiple benchmarks. The first fine-tuning round focused on basic Verilog syntax, while the second addressed realistic problem-solving tasks. The RAG module integrated examples and knowledge retrievers, improving accuracy and addressing hallucination issues. The study highlighted AutoVCoder as a robust and efficient methodology for Verilog code generation.

To further refine HDL generation capabilities, Nakkab et al. explored the use of hierarchical prompting techniques to improve the effectiveness of LLMs in handling complex hardware design tasks [84]. The study introduced a suite of hierarchical prompting techniques, facilitating stepwise design methods for efficient HDL code generation. They developed a generalizable automation pipeline and presented a new benchmark set of hardware designs with hierarchical solutions. Empirical evaluations demonstrated that hierarchical prompting significantly improved performance, particularly for complex designs, by breaking down tasks into manageable submodules. Notably, they successfully generated a 16-bit MIPS processor and a 32-bit RISC-V processor using purely generative hierarchical prompting. The study concludes that hierarchical prompting is a crucial tool for automated HDL generation, allowing smaller LLMs to compete with larger models.

Another line of research investigates the interface between LLMs and design environments. Abdelatty and Reda introduced HDLCopilot, an LLM-powered query system designed to interact with Process Design Kits (PDKs) using natural language, significantly enhancing the efficiency and accuracy of hardware design workflows [80]. HDLCopilot converts natural language queries into SQL queries to retrieve relevant data from a PDK database, providing natural language responses. The system comprises four agents: Dispatcher, Selector, SQL Generator, and Interpreter, each handling different aspects of query processing. Evaluated on the Skywater 130nm PDK database, HDLCopilot achieved high accuracy (94.23%) and efficiency (98.07%) in information retrieval. This multi-agent framework demonstrated the ability to handle complex queries and streamline the design and verification processes, showing promise for further integration with hardware design tools and enhanced training schemes to support diverse and complex PDK queries.Futhermore, Chang et al. introduced a novel approach leveraging multimodal generative AI that combines natural language and visual input to improve Verilog code generation [77]. The study highlights the limitations of purely natural language-based models in capturing complex spatial relationships and proposes a benchmark for evaluating multimodal models. The results show substantial improvements in syntax correctness and functionality, with test bench passing rates for GPT-4 models improving from 46.88% to 71.81% and LLaMA models from 13.41% to 25.88%. The research underscores the advantages of multimodal input in generating accurate Verilog code, particularly for multimodule hardware and state machines. The authors are working to refine multimodal models, develop more comprehensive benchmarks, and improve their integration with EDA tools, ultimately improving hardware design processes.

Recent works have also explored broader applications. Mengming et al. showed LLMs' potential to enhance IC development by streamlining the creation of architecture specifications, from high-level to detailed descriptions, using specific prompts that integrate RTL code [57]. Their text analysis capabilities also provide effective overviews and unique insights, reducing human error and saving time. Wang et al. introduced ChatPattern, a framework powered by LLMs for generating and customizing layout patterns based on natural language inputs. ChatPattern leveraged an LLM agent capable of interpreting user requirements, decomposing them into subtasks, and autonomously operating design tools to generate layout patterns [64]. The framework performs tasks such as conditional layout generation, pattern modification, and both fixed-size and free-size pattern generation. Experimental results highlight ChatPattern's performance in producing diverse and legally compliant patterns compared to existing methods. The study underscores the potential of ChatPattern to provide a user-friendly interface for the customization of layout patterns, although future research is needed to improve the handling of complex and oversized patterns and to enhance global guidance during pattern generation.

Liu et al. introduced a tool leveraging the Llama2-70B language model to automate the layout of silicon photonic devices, enhancing photonic design automation (PDA) [76]. This tool significantly reduces human intervention by generating layout code from natural language prompts. Preliminary tests showed a success rate of over 99% in generating accurate layout code. Direct comparisons with GPT-4 highlighted superior performance in error-free code generation. The tool was tested with various user prompts, consistently producing accurate layouts. It used system prompts, SQL database checks, and code verification to ensure accuracy. This tool shows potential for automating silicon photonic device layout with high efficiency and minimal human intervention, though further development is needed to handle more complex tasks such as routing and component placement.

He et al. introduced ChatEDA, an autonomous agent for EDA powered by the AutoMage large language model [81]. ChatEDA streamlines the EDA design flow from RTL to Graphic Data System Version II (GDSII) by managing task planning, script generation, and task execution. AutoMage, fine-tuned using a self-instruction paradigm and QLoRA technique, outperformed other models like GPT-4 in task planning and script generation. Evaluations showed that AutoMage achieved a Grade A in 88% of test cases compared to 58% for GPT-4, demonstrating its effectiveness in automating the RTL-to-GDSII design flow. The study highlighted the potential of ChatEDA to enhance design efficiency and reduce errors, though further improvements are needed for more complex design scenarios.

To evaluate creativity in hardware design, DeLorenzo et al. introduced CreativEval, a framework for assessing LLM generated design based on fluency, flexibility, originality, and

elaboration. They evaluated various LLMs, including GPT models, CodeLlama, and VeriGen, demonstrating GPT-3.5 emerging as the most creative. [66]. Complementarily, Zhao et al. introduced CodeV, a framework enhancing LLMs for Verilog code generation through multi-level summarization [78]. By curating high-quality GitHub data and leveraging GPT-3.5, it improves instruction-tuning datasets and outperforms GPT-4 and BetterV on benchmarks. Fine-tuning models like CodeLlama boosts performance, demonstrating its potential for circuit design automation.

HLS tools facilitate rapid hardware design from C code but face limitations due to incompatible code constructs. In this paper, Collini et al. investigate the use of LLMs to refactor C code into HLS-compatible formats. Several case studies are presented where an LLM rewrites C code for NIST 800-22 randomness tests, QuickSort, and AES-128 into HLS-synthesizable C. The LLM iteratively transforms the code with user guidance, adding functions like streaming data and hardware-specific signals. This study highlights the LLM's potential in aiding the refactoring of standard C code into HLS-synthesizable code [85]. similarly, Meech introduced a methodology combining HLS tools and LLMs to generate, simulate, and deploy hardware designs, focusing on a uniform random number generator with a wishbone interface. The study demonstrated using LLMs, specifically Microsoft Bing Chat, to generate Python-based Amaranth HDL scripts. The design was validated through simulations and the Dieharder randomness test suite, showing performance comparable to established random number generators. The methodology facilitated iterative refinement of designs, ensuring correctness and functionality. The final design was successfully deployed on an Icebreaker FPGA, integrating with the Caravel SoC platform. This approach lowers the entry barrier for hardware design, making it more accessible for prototyping domain-specific computing accelerators, with future work aiming to incorporate true random number generators and explore additional HLS tools and languages [75].

*2) Optimization And Synthesis:* LLMs are increasingly being employed to enhance the efficiency and quality of hardware designs in optimization and synthesis. These models are leveraged to optimize power, performance, and area (PPA) designs as well as assist in synthesizing HDL code from high-level specifications. By automating and refining these processes, LLMs enable more efficient design implementations and a significant reduction in optimization time.

A common thread among several studies is the use of LLMs to optimize PPA metrics through innovative frameworks and methodologies. For instance, Thorat et al. introduced the VeriPPA framework to optimize the PPA constraints in Verilog Designs, significantly improving the quality of LLM-generated Verilog code to meet industry standards [54]. Similarly, Kaiyan Chang et al. introduced ChipGPT, a framework that optimizes the quality of hardware designs generated by LLMs through a post-LLM search approach. Integrating design space exploration into the workflow improves PPA metrics [65].

Chowdhury et al. further advanced this domain by introducing the ABC-RL methodology, which leverages past design data and a retrieval-guided reinforcement learning approach to optimize logic synthesis, achieving up to 24.8% improvement in Quality-of-Result (QoR) and up to 9x runtime reduction compared to state-of-the-art techniques [86]. Attention was used in this work to help encode synthesis recipes in a format that allowed the other ML elements of the system to better learn/predict the circuit state resulting from each synthesis optimization step.

Another prominent theme is the use of LLMs for automated HDL/RTL generation and synthesis, often combined with advanced search algorithms to ensure functional correctness and optimality. DeLorenzo et al. proposed a novel approach integrating LLMs with Monte Carlo Tree Search (MCTS) to generate high-quality RTL code [59]. Their automated transformer decoding algorithm, guided by MCTS, produces compilable and functionally correct RTL code, significantly improving the area-delay product (ADP) for various hardware modules . Nazzal et al. introduced the Systolic Array-based Accelerator DataSet (SA-DS), designed to facilitate the application of LLMs in generating and optimizing DNN hardware accelerators [67]. The dataset, created using the Gemmini generator, includes a variety of spatial array configurations with accompanying verbal descriptions and Chisel code. The study demonstrates that fine-tuning LLMs with SA-DS significantly improves the quality and efficiency of hardware design generation, highlighting the potential for minimal human intervention.

Beyond PPA optimization and RTL generation, researchers are also exploring the integration of LLMs with electronic design automation (EDA) tools to enhance scalability and usability. Jiang et al. introduced IICPilot, an intelligent IC backend design system utilizing LLMs and a multi-agent framework to automate various backend design processes using open-source EDA tools [79]. IICPilot automates script generation, EDA tool invocation, design space exploration of parameters, and dynamic resource allocation through containerization. The framework separates the backend workflow from specific EDA tools via a unified interface and leverages LangChain's multi-agent system for tasks such as floorplanning and routing. Experimental validation on platforms like iEDA and OpenROAD demonstrated significant improvements in task completion time and resource utilization, achieving up to 32.75% optimization in PPA metrics. The system's scalability and fault tolerance were highlighted, with experiments conducted on a multi-node setup on Alibaba Cloud involving 400 benchmarks from OpenCores. IICPilot enhances IC backend design efficiency, reducing the entry barrier for designers and showing promise for future integration with more EDA tools and platforms.

Ho and Ren further extended the application of LLMs to physical layout optimization, specifically targeting advanced semiconductor nodes at the 2nm technology node [82]. Their methodology employs the ReAct prompting technique to integrate human design expertise with LLM reasoning, resulting in up to 19.4% smaller cell area and 23.5% more LVS/DRC clean layouts compared to traditional methods. This work underscores the versatility of LLMs in understanding and optimizing netlist topologies, device clustering constraints, and physical layouts.

*3) Verification And Validation:* Verification and validation are crucial in hardware design to ensure functionality and compliance with specifications. LLMs may automate the generation of test cases, verify the correctness of HDL code, generate hardware assertions, and perform formal verification, thereby identifying design flaws early in the development process. This reduces the time and cost associated with verification.

A key area of focus has been the development of benchmarking frameworks to systematically evaluate the capabilities of LLMs in hardware design. Lu et al. introduced RTLLM as an open-source benchmark for systematic and quantitative evaluation of the design RTL from the natural language [23], focusing mainly on syntax correctness, functional accuracy, and design quality. They also explored a self-planning prompting technique that they use to improve the performance of the LLMs. Similarly, Mingjie et al. developed VerilogEval, a benchmarking framework for automatic functional correctness testing of Verilog code generations [24]. They derived the evaluation dataset from HDLBits, an online coding platform for Hardware Engineers. Their study demonstrates that supervised fine-tuning (SFT) with synthetic problem-code pairs can enhance pre-trained LLMs' Verilog code generation capabilities.

Another prominent theme is the use of LLMs for assertion generation and formal verification, which are essential for ensuring design reliability. Fang et al. proposed AssertLLM, which illustrated how LLMs can generate and evaluate hardware verification assertions, improving the reliability of hardware designs [29]. Complementing this, Mali et al. introduced ChiRAAG, a framework that automates the generation of SystemVerilog Assertions(SVAs) from natural language specifications using LLM prompting [58]. This research helps to streamline the formal property verification process in hardware design. Further advancing this area, Hassan et al. proposed a formal verification methodology that utilizes OpenAI's GPT-4 for automating invariant generation and integrates mutation testing to validate these invariants for hardware designs, demonstrated with the ISCAS-85 C432 27-channel interrupt controller, achieving effective invariant validation and a mutation score of 1 out of 20 mutants [60].

The automation of testbench generation and functional verification has also been a significant focus. Qui et al.. introduced AutoBench, a systematic and generic framework for generating Verilog testbenches using LLMs for RTL verification. AutoBench combined Python and Verilog codes in a hybrid testbench architecture, enhancing efficiency and effectiveness [87]. The framework included comprehensive code generation, scenario checking, standardization, and automatic debugging. AutoEval, an automatic evaluation framework, assessed the generated test benches using multiple metrics, demonstrating higher pass ratios and task completion rates than traditional methods. The self-enhancement system improved test benches through scenario checking and auto-debugging, significantly enhancing coverage and correctness. The study underscored AutoBench's impact on automating HDL verification and suggests future research to refine LLM capabilities and integrate with EDA tools for more complex verification tasks. Similarly, Blocklove et al. evaluated the capability of LLMs like ChatGPT-4 and ChatGPT-3.5 in generating functional Verilog code and corresponding testbenches for hardware design and verification [70]. The study developed benchmarks to assess LLMs' performance across various hardware functions and validated the designs through simulation and silicon testing. Results showed that while LLMs can generate functional HDL code, their ability to produce comprehensive testbenches is limited, with ChatGPT-4 outperforming ChatGPT-3.5. The authors emphasize the need for improvements in error understanding, testbench generation, and handling complex designs. The author plans to futher enhance dataset quality, training schemes, and integration with EDA tools to advance the automation of the digital design pipeline.

Beyond general-purpose frameworks, researchers have also investigated the use of LLMs for application-specific hardware design and verification. Vitolo et al. explored using LLMs, specifically ChatGPT-4, to automate the generation and validation of hardware description code for a Recurrent Spiking Neural Network (RSNN) in Verilog [68]. The study showcased the successful design of a 3x3x3 RSNN and corresponding test benches through an iterative process facilitated by ChatGPT-4, achieving high accuracy in tasks such as exclusive OR, IRIS flower classification, and MNIST handwritten digit classification. Prototyping on an FPGA and implementing using SkyWater 130 nm technology, the RSNN demonstrated efficient performance metrics. This work underscores the feasibility of using LLMs for complex hardware design, suggesting future research to enhance dataset quality, training schemes, and integration with electronic design automation tools for more sophisticated hardware projects. Similarly, Thorat et al. discussed the veriRectify process, which uses error diagnostics from simulators to improve the integrity of generations of Verilog code [54]. Finally, the optimization of LLMs for hardware design tasks and their integration with Electronic Design Automation (EDA) tools have emerged as important research directions. Lily Jiaxin Wan et al. explored techniques such as quantization, pruning, and software/hardware co-design to optimize LLMs for efficient inference [61]. They applied these optimized models to functional verification in circuit design and introduced the Chrysalis dataset to enhance LLM-based debugging tools. This work aligns with broader efforts to refine LLM capabilities and integrate them into existing EDA workflows, as highlighted by several studies mentioned earlier.

*4) Debugging And Maintenance:* LLMs may play a crucial role in debugging and maintaining hardware designs. They can assist in identifying and fixing syntax errors in HDL code, streamlining the debugging process, and ensuring the correct operation of hardware designs. This facilitates efficient troubleshooting and maintenance.

Researchers have explored the role of LLMs in debugging and maintenance. For instance, YunDa et al. developed an RTLFixer framework that automates syntax corrections in Verilog code using LLMs [39]. They utilize Retrieval-Augmented Generation (RAG) and ReAct prompting, enabling LLMs to act as autonomous agents in interactively debugging the code with feedback . Similarly, the HDLdebugger by Yoa et al. provided insights into how LLMs can simplify the debugging process for HDL codes, making it easier for hardware

engineers to maintain high-quality designs [62]. They used RAG, which integrates a search engine that retrieves relevant HDL documentation and code instances to enhance the LLM's debugging capability.

Another shared focus is on improving the interpretability of error messages and making hardware design tools more accessible to novices. Qiu et al. explored this by using LLMs to generate user-friendly explanations for compile-time synthesis error messages from EDA tools like Quartus Prime and Vivado, demonstrating that approximately 71% of LLM-generated explanations were correct and complete, aiding in educational contexts for novice digital hardware designers [88]. This aligns with the broader trend of using LLMs to bridge the gap between complex hardware design processes and user comprehension.

In addition to debugging, LLMs have been applied to enhance the reliability of HDL code generation for specific hardware applications. Xiang et al. investigated this by developing a three-phase Pulse Width Modulation (PWM) generator using LLMs. Their study emphasized strategies such as role specification, hierarchical design, and error feedback mechanisms to address syntax errors and high-level semantic challenges in LLM-generated code [69]. The successful fabrication and validation of the PWM generator using tools like iVerilog and GTKWave underscore the potential of LLMs in integrated circuit (IC) design. This work also highlights the importance of integrating LLMs with existing EDA tools to improve their practical utility.

Furthermore, LLMs have shown potential in addressing security-relevant hardware bugs. Ahmad et al. investigated the potential of LLMs for automatically repairing security-relevant hardware bugs in Verilog code [73]. The study introduced a framework for quantitatively evaluating LLMs' performance in fixing these bugs and compares their effectiveness against state-of-the-art automated repair tools. Experimental results showed that LLMs, particularly GPT-4, can effectively repair simple hardware security bugs and sometimes outperform existing tools. Detailed prompt engineering significantly enhanced the success rate of LLM-generated repairs. However, LLMs struggled with more complex issues like race conditions and multi-line fixes. The paper highlighted the potential of LLMs in hardware bug repair while acknowledging their limitations and suggests future research should focus on hybrid methods, fine-tuning LLMs for hardware-specific tasks, and improving prompt strategies for more complex bug fixes.

Table I provides a comprehensive view of different methods and AI models used in digital hardware design from 2022 to 2024. It further classifies these methods into the year they were introduced and their capabilities regarding design assistance, optimization, synthesis, verification, validation, debugging, and maintenance. The last column lists the associated AI models used in each paper, such as GPT, CodeGen, etc. This table highlights the evolution and integration of AI-driven tools in digital hardware design, focusing on methods that involve various levels of assistance in the design and verification process. The presence of diverse models, such as GPT-3.5, GPT-4, and versions of Llama, and some fine-tuned models, such as VeriGen and CodeGen, indicates a rise in the use of powerful AI models over time to support hardware-related tasks.

### *B. LLM-based Analog Hardware Design*

LLMs have demonstrated transformative potential not only in digital systems but also in analog systems and is now gaining significant traction. These frameworks leverage the contextual understanding, domain knowledge, and generative capabilities of LLMs to address unique challenges in analog circuit design, topology generation, and layout optimization. A key area of focus is using LLMs to automate and assist in analog circuit design. For instance, Lai et al. introduced AnalogCoder, a training-free LLM agent designed specifically for analog circuit design through Python code generation. AnalogCoder leveraged a feedback-enhanced design flow and a comprehensive circuit tool library to automate the design process [89]. It successfully designed 20 out of 24 benchmark circuits, outperforming other LLM-based methods. The feedback mechanisms, which included requirement check, simulation and operating point check, DC sweep check, and function check, significantly improved design success rates. By archiving successful designs as reusable modular sub-circuits, AnalogCoder simplified the creation of complex circuits. Similarly, Chang et al. introduced LaMAGIC, a pioneering framework using language models to generate optimized analog circuit topologies from user-defined specifications, particularly for power converter applications [90]. Leveraging supervised fine-tuning, LaMAGIC can efficiently produce high-quality circuit designs in a single pass, significantly reducing design time compared to traditional methods. The framework demonstrated a success rate of up to 96% under strict tolerances and showed superior performance with adjacency-matrix-based formulations for more complex circuits. The model's scalability was validated with 6-component circuits, and data augmentation improved generation capabilities. These works underscore the role of LLMs in automating repetitive design tasks and improving design accuracy through iterative feedback and modularization. Another significant application of LLMs is in optimizing analog circuit design processes. Chen et al. presented LLANA, a framework that leveraged LLMs to enhance bayesian optimization for generating analog design-dependent parameter constraints [91].LLANA reduces simulation requirements while maintaining adaptability across various circuit designs, achieving results comparable to state-of-the-art Bayesian optimization methods with fewer simulations. The framework's success is demonstrated in the design of two-stage operational amplifiers, where it achieves better optimization with fewer sampless.

Similarly, Yin et al. introduced ADO-LLM, a framework that combines bayesian optimization with LLMs to enhance analog circuit design efficiency and effectiveness [92]. ADO-LLM leverages LLMs' domain knowledge to generate initial design points and uses Gaussian process-based bayesian optimization to optimize these points iteratively. The framework demonstrated notable improvements in design efficiency and performance on two analog circuits: a two-stage differential amplifier and a hysteresis comparator.

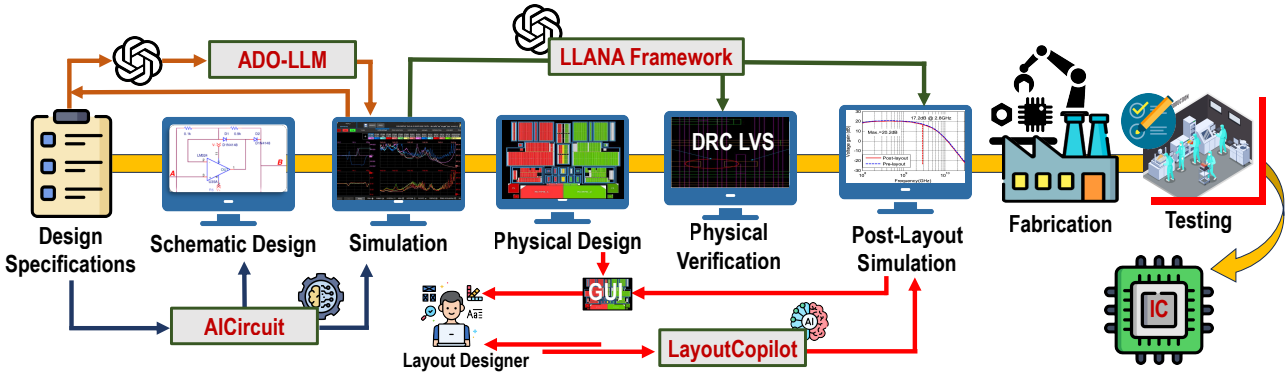


Fig. 11. Key stages in the IC design flow showcasing the integration of AI frameworks like ADO-LLM, LLANA, AICircuit, and LayoutCopilot for optimizing schematic, simulation, and physical design processes.

TABLE II
LLM-BASED ANALOG HARDWARE DESIGN

| Year | Method Name | Application | AI Model |
|---|---|---|---|
| 2024 | LLANA [91] | Design Assistance | GPT3.5 |
| 2024 | LaMAGIC [90] | Design Assistance | Language Model |
| 2024 | Analog Coder [89] | Design Assistance | GPT3.5 |
| 2024 | ADO-LLM [92] | Optimization | GPT-3.5 Turbo |
| 2024 | Layout Copilot [93] | Layout Design | GPT-3.5,4,Claude 3 |

Both frameworks highlight the synergy between LLMs and optimization algorithms, enabling more efficient exploration of design spaces and reducing computational overhead. LLMs have also been applied to analog layout design, where they assist in optimizing physical layouts and improving verification processes. Chang et al. introduced LayoutCopilot, an LLM-powered multi-agent collaborative framework designed to enhance the interactive analog layout design process [93]. This framework integrates LLMs within a multi-agent system to streamline and optimize design tasks, achieving 6.49% improvement in logical verification, 4.85% in syntactic verification, and 5.89% in overall accuracy using structured instructions. Multi-agent configurations proved more effective, achieving a correctness rate of 96.80% with Claude-3. Post-interactive adjustments, the layout area was reduced to 66% of its original size, with notable gains in performance metrics such as Gain, Phase Margin (PM), and Common-Mode Rejection Ratio (CMRR) tools.

Figure 11 illustrates the integration of AI frameworks into the analog IC design flow. The figure primarily highlights how these frameworks optimize various IC design stages. All these AI frameworks are utilized to enhance the efficiency and precision of the hardware design process. This figure showcases the underlying roles of these AI tools in streamlining the analog IC design workflow.

Table II summarizes recent works that apply LLMs to analog hardware design. This table lists five significant papers from 2024, highlighting their applications and the AI models they use. The papers: LLANA, LaMAGIC, and Analog Coder focus specifically on design assistance, whereas ADO-LLM and Layout Copilot address optimization and layout design, respectively. In these studies, researchers employ advanced LLMs such as GPT-3.5, GPT-3.5 Turbo, GPT-4, and Claude 3, showcasing their ability to handle complex design processes. These papers emphasize how LLMs increasingly enhance efficiency and accuracy throughout the analog hardware design workflow.

### C. Other Attention-based Hardware Design

Attention-based models have emerged as a revolutionary approach in the design of hardware, efficiently solving open challenges in areas like floor planning and macro placement [94], logic synthesis, and standard cell design [95]. Figure 12 provides a structured approach to how attention-based models can be leveraged to these challenges. It starts with identifying the challenges of optimal macro placement, efficient floor planning, and enhancing logic synthesis and standard cell design. These challenges are then tackled by advanced attention-based models, such as GATs, transformers, and diffusion models. GATs show the unparalleled performance of capturing the dependencies of graph-structured data and could serve well in tasks like logic synthesis or floor planning, wherein the circuit can be visualized as a graph. Transformers capture long-range dependencies and contextual information and are thus applied to tasks such as macro placement and logic synthesis. Often, this is combined with MCTS to refine the generated solution further. It uses diffusion models, known to generate high-quality distributions, for macro placement and standard cell design challenges. Methods like reinforcement learning further enhance these techniques to optimize the solutions. Furthermore, different techniques are employed, such as EAGAT combined with reinforcement learning for floor planning, Transformers with MCTS for logic synthesis, and Diffusion Models for macro placement. This model, when combined, will provide a solid framework for

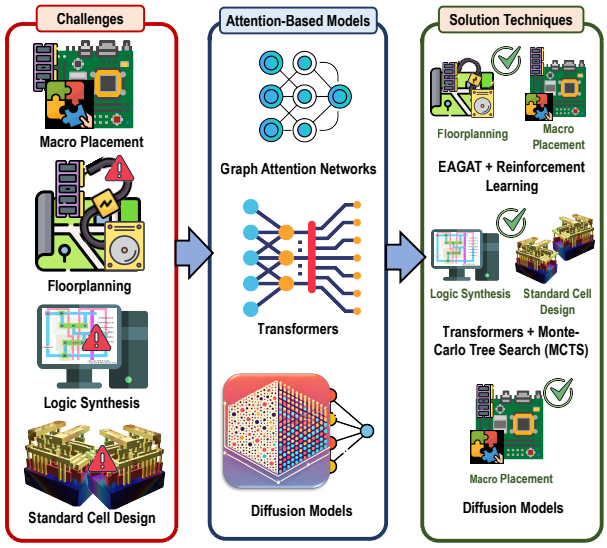


Fig. 12. Overview of attention-based hardware design techniques leveraging models like graph attention networks (GATs), transformers, and diffusion models to address challenges in hardware design.

improving efficiency and automation in the hardware design workflow.

One prominent area of application is floorplanning and macro placement, where optimizing chip area, wire length, and placement legality are critical. Bo Yang et al. introduced an end-to-end reinforcement learning methodology for chip floorplanning that optimizes macro placement and orientation to minimize chip area and wire length, leveraging an edge-aware graph attention network (EAGAT) [28]. Similarly, Lee et al. introduced an approach using diffusion models for macro placement in digital circuit design to overcome RL methods' limitations [96]. The study proposed a diffusion model that placed all macros simultaneously, avoiding the sequential limitations associated with RL. A novel neural network architecture combining graph convolutions and multi-headed attention layers was developed for efficient and expressive placement. The authors introduced a method for generating large synthetic datasets for pre-training the diffusion model, facilitating training without relying on proprietary data. Empirical evaluations demonstrated competitive performance on benchmark datasets, showing high legality and low half-perimeter wire length (HPWL) ratios.

In logic synthesis and standard cell design, attention-based models excel because they handle graph-structured data and long-range dependencies. Li et al. introduced the Circuit Transformer. This transformer-based model predicts the next logic gate for end-to-end logic synthesis. It leveraged a novel encoding scheme and integrated Monte-Carlo Tree Search to optimize and improve the generated circuit designs [97].

In standard cell design, Chia-Tung Ho et al. from Nvidia introduced an innovative transformer-based clustering methodology to improve standard cell design automation [98]. By training the model with LVS/DRC clean cell layouts and utilizing personalized page rank vectors, the authors addressed challenges in routability for advanced nodes. Their approach significantly enhanced performance, generating 15% more LVS/DRC clean layouts, reducing average cell width by 3.9 %, and total wire length by 3.3%, while achieving a 12.7× speedup over previous methods. The study demonstrated the model's scalability, achieving 100% LVS/DRC clean layouts for over 1000 single-row cells and reducing cell area by 14.5% compared to industrial standards.

Scalability and generalizability are key challenges in applying attention-based models to large-scale circuit design. Deng et al. introduced Hop-Wise Graph Attention (HOGA), a novel approach to enhance the scalability and generalizability of GNNs for large-scale circuit representation learning [99]. HOGA precomputes hop-wise features to enable efficient distributed training and adaptively captures high-order structures in circuits. The model demonstrated superior performance, reducing the estimation error for Quality of Results (QoR) prediction by 46.76% and improving reasoning accuracy by 10% over conventional GNNs. Evaluations conducted on the OpenABC-D benchmark and functional reasoning tasks highlighted HOGA's ability to handle complex circuit tasks with improved training efficiency and scalability. Despite its higher computational cost, HOGA's architecture facilitates massive parallelization, making it suitable for large-scale circuit datasets and presenting a significant advancement in EDA.

Attention-based models have also been applied to hotspot detection in semiconductor layout design, a critical aspect of design for manufacturability. Zhu et al. introduced a novel single-stage end-to-end hotspot detection framework leveraging multi-task learning and a transformer Encoder to enhance efficiency and accuracy in semiconductor layout design hotspot detections [100]. The framework utilizes a single-stage detector with center and corner heads for improved representation learning and a transformer Encoder for global feature aggregation. Experimental results showed the framework achieving a 97.31% detection accuracy and a significant speedup of up to 67.84 times compared to previous methods. It effectively reduced false alarms by 87.6% and 59.5% compared to the TCAD'19 and DAC'19 methods, respectively. Evaluations conducted on the ICCAD CAD Contest 2016 Benchmarks demonstrated the framework's superior performance. This efficient and robust solution holds promise for advanced design for manufacturability research, with future work suggested to enhance complexity handling and applicability to diverse semiconductor manufacturing processe. Transferability and automation are increasingly important in chip placement, where reducing runtime and improving placement quality is critical. Yao Lai et al. explored these aspects with ChiPFormer, a transformer-based approach that leveraged offline RL to develop a transferable placement policy, significantly enhancing placement quality and reducing runtime [101]. The model demonstrated a 10× reduction in runtime and superior half perimeter wire length (HPWL) compared to recent RL-based methods. ChiPFormer effectively transferred learned policies to new chip designs, reducing placement time from hours to minutes, and outperformed existing methods such as GraphPlace, MaskPlace, and DeepPR in both HPWL and runtime across extensive experiments on 32 chip circuits.

Researcher have also explored the domain of analog and radio-frequency circuit design. Mehradfar et al. introduced

TABLE III
OTHER ATTENTION-BASED HARDWARE DESIGN

| Year | Method Name | Application | AI Model |
|---|---|---|---|
| 2021 | [100] | Hotspot Detection | Multi headed transformer |
| 2023 | Deep EdgeGAT [25] | STA Analysis | GAT-based model |
| 2023 | ChiPFormer [101] | Placement | Decision Transformer |
| 2024 | [28] | Floorplanning | Edge-aware graph attention network |
| 2024 | Circuit Transformer [97] | Logic Synthesis | Transformer-based model |
| 2024 | [98] | Standard Cell Design Automation | Transformer model |
| 2024 | HOGA [99] | Representation | Attention-based model |

AICircuit, a novel multi-level dataset and benchmark designed to advance the application of ML in analog and radio-frequency circuit design [102]. The paper highlighted the challenges inherent in analog circuit design, particularly the complexity and time intensity as circuit parameters increased. AICircuit included seven fundamental circuits and two complex wireless transceiver systems, serving as a comprehensive resource for evaluating various ML algorithms. The authors demonstrated the potential of ML models, such as multi-layer perceptrons and transformers, in mapping design specifications to circuit parameters, significantly simplifying the design process. However, they noted that more complex circuits with higher non-linearity remained challenging for ML models, underscoring the need for further optimization.

Figure 12 Overview of attention-based hardware design techniques leveraging models like GATs, transformers, and diffusion models to address challenges in floorplanning, logic synthesis, macro placement, and standard cell design.

Table III summarizes some recent works using attention-based models for hardware design. The table covers six important works from 2021 to 2024, listing the application and AI model used. These papers use the attention-based model for hotspot detection, STA, placement, floorplanning, logic synthesis, standard cell design automation, and representation. The researchers apply different advanced attention-based models, including multiheaded transformers, GAT-based models, decision transformers, and edge-aware graph attention networks. These works show an increasing use of attention mechanisms in hardware design workflows.

## IV. ATTENTION-BASED HARDWARE SECURITY

### A. LLM-based Hardware Security

This section presents the role of LLMs in enhancing hardware security on various tiers of applications such as vulnerability detection, threat analysis, security verification, and logic obfuscation. We also explore frameworks on automated debugging, security assertion generation, and thwarting side-channel attacks. A common theme running across these works is to address the challenge of scalability, accuracy, and efficiency in hardware security using LLMs.

Figure 13 highlights key security vulnerabilities and challenges in hardware design, including fault injection attacks, SoC security, logic obfuscation, and side-channel attacks. It illustrates the potential of LLM/attention models such as GPT-3.5, Llama3-70B, GNNs, and HS-BERT in enhancing security by addressing detection accuracy, scalability, debugging efforts, and dataset limitations, enabling effective threat

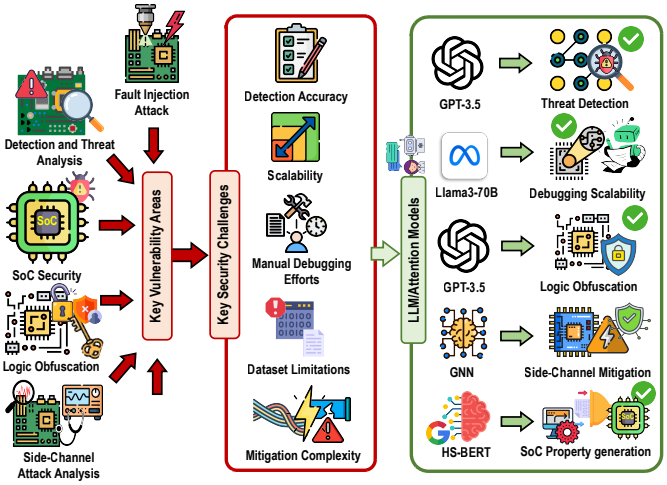


Fig. 13. Overview of security vulnerabilities, challenges, and the role of LLMs and Attention-based models in detection, mitigation, and debugging for hardware security.

detection, logic obfuscation, side-channel mitigation, and SoC property generation.

*1) Vulnerability Detection And Threat Analysis:* A significant focus of recent research has been on leveraging LLMs for vulnerability detection and threat analysis in hardware systems. Lin et al. presented the HW-V2W-Map framework, an NLP-based system focused on IoT hardware vulnerabilities [103]. This framework uses ontology-driven storytelling to analyze and mitigate vulnerabilities, predict future exposures, and provide mitigation suggestions using GPT models. The framework offers an interactive GUI for identifying mitigation strategies and determining exploit and impact scores with the Common Vulnerability Scoring System. Similarly, Saha et al. presented Vul-FSM, a database containing 10,000 vulnerable finite state machine (FSM) designs featuring 16 specific security vulnerabilities [104]. The paper introduces SecRT-LLM, leveraging LLMs to create extensive hardware security datasets efficiently. Results demonstrate the framework's effectiveness: initial attempts achieve an average pass rate of 81.98% for vulnerability insertion and 97.37% for detection, improving to 80.30% and 99.07%, respectively, within five attempts. This highlights LLMs' proficiency in enhancing ML-based methods for hardware security benchmarking and mitigation strategies.

Another critical area of research is the automation of security verification and debugging processes. Akyash et al. introduced Self-HWDebug, a framework designed to enhance

the scalability and efficiency of LLMs in automating the mitigation of security vulnerabilities in hardware designs [105]. The framework leveraged LLMs to autonomously generated detailed debugging instructions, addressing the challenge of manually crafting instructions that demands significant time and expertise. Self-HWDebug prompts LLMs to create targeted mitigation instructions by utilizing pre-identified hardware bugs and resolutions, extending solutions across similar vulnerabilities in different designs. Initial testing shows the framework's efficacy in reducing human intervention and improving debugging quality. Complementing this work, Saha et al. explored the integration of LLMs, particularly GPTs, into the security verification process of SoC designs [106]. Current security solutions struggle with scalability and adaptability, making them inadequate for modern SoCs. The research aims to create a more efficient and comprehensive security verification methodology by leveraging LLMs. The paper analyzes existing works, provides case studies and experiments, and offers guidelines for using LLMs in SoC security. It highlights the potential benefits and challenges of this innovative approach. Formal verification has emerged as a key technique for identifying and mitigating Common Weakness Enumerations (CWEs) in LLM-generated hardware designs. Gadde et al. introduced ReFormAI, a dataset of 60,000 SystemVerilog designs generated by various LLMs, focusing on identifying and addressing CWEs through formal verification, demonstrating that approximately 60% of LLM-generated designs are prone to CWEs, and highlighting the superior reliability of formal verification over traditional testing methods. Gadde et al. investigated the prevalence of CWEs in SystemVerilog hardware designs generated by LLMs and evaluated the effectiveness of formal verification in identifying these vulnerabilities [107]. They introduced the ReFormAI dataset, comprising 60,000 SystemVerilog designs generated by four different LLMs targeting various CWEs. The study revealed a high vulnerability rate, with approximately 60% of the generated designs containing CWEs. Formal verification techniques successfully identified and categorized these vulnerabilities. Among the evaluated LLMs, GPT-3.5-Turbo performed best but still produced a significant number of vulnerable designs. The study underscores the need for better training datasets and fine-tuning techniques to reduce CWEs in LLM-generated hardware designs.

Building on this, Fu et al. presented Hardware Phi-1.5B, an LLM tailored explicitly for the hardware domain, addressing challenges in hardware design and security verification [74]. The model leveraged a specialized, tiered dataset focused on hardware-specific content, demonstrating advancements in predicting the next tokens and handling complex hardware design tasks. Experimental results validated the model's proficiency, particularly on the Common Weakness Enumeration (CWE) dataset.

The automation of hardware security assertion generation is another promising application of LLMs. Kande et al. investigated the potential of LLMs in automating the generation of hardware security assertions [108]. By introducing a new evaluation framework and benchmark suite, the study explored the effectiveness of LLMs in generating correct security assertions for hardware designs. OpenAI's Codex showed a modest success rate, generating correct assertions 4.53% of the time, with detailed prompts improving performance. The study used Siemens Modelsim for simulation and validation, demonstrating the framework's scalability to other LLMs. Despite challenges, such as the need for detailed prompts and context, the results provide a proof-of-concept for LLM-assisted assertion generation, suggesting future work on fine-tuning models, improving datasets, and exploring quick evaluation methods to enhance accuracy and effectiveness in hardware security applications. Recent efforts have also explored the use of LLMs for automated Trojan detection and mitigation in analog designs. Chaudhuri et al. developed SPICED+, a software-based framework that leverages LLM-enhanced analysis of SPICE netlists and simulation logs to iteratively detect, localize, and remove analog hardware Trojans [109]. The framework combines rule-based prompting, few-shot learning, and voltage/current deviation analysis to classify Trojan-impacted nodes and applies an iterative correction loop guided by a confidence scoring mechanism. By integrating with HSPICE for netlist resimulation and validation, SPICED+ achieves an average Trojan coverage of 93.3%, a mitigation rate of 94.6%, and maintains a low false positive rate of 1.4% , outperforming prior ML-based tools like DAWN. Its model-agnostic LLM workflow enables generalization to new circuits without retraining, though large-scale designs such as SAR-ADCs may require modular decomposition for efficient context processing.

*2) SoC Security:* LLM-based tools have demonstrated significant potential in automating and enhancing the security verification process for SoC designs. These tools address the limitations of traditional methods by leveraging NLP and LLMs to extract security properties, identify vulnerabilities, and generate comprehensive security policies.For instance, Meng et al introduced NSPG, an innovative NLP-based tool designed to automate the generation of security properties from SoC documentation [110]. The tool leverages the HS-BERT model, a hardware security-specific language model trained on sentences from RISC-V, OpenRISC, MIPS, OpenSPARC, and OpenTitan SoC documentation. In evaluations on five untrained OpenTitan hardware IP documents, NSPG successfully extracted 326 security properties from 1723 sentences, aiding in the identification of eight security bugs presented in the Hack@DAC 2022 competition. This approach demonstrates superior performance compared to traditional methods and popular tools like ChatGPT, highlighting its potential to streamline the security validation process and improve the robustness of SoC designs. The success of NSPG suggests a promising future for NLP applications in hardware security and SoC verification. Similarly, Tarek et al. introduce SoCureLLM, an LLM-based framework designed to improve hardware security verification for complex SoC designs [111]. SoCureLLM addresses the limitations of traditional methods by effectively identifying security vulnerabilities and generating a comprehensive security policy database. In evaluations, it detected 76.47% of security bugs in three vulnerable RISC-V SoCs, surpassing existing methods. Additionally, SoCureLLM formulated 84 new security policies for large-scale RISC-V

SoC designs, enhancing the security policy database. This framework demonstrates significant improvements in adaptability, scalability, Trojan insertion analysis, and efficiency for SoC security verification.

*3) Trojan Insertion:* Research in hardware security has increasingly leveraged LLMs for generating, inserting, and detecting hardware trojans. These studies demonstrate the dual-use nature of LLMs, addressing challenges such as dataset scarcity, context length limitations, and the need for diverse HT examples in hardware design. Kokolakis et al. investigate the potential of LLMs in offensive hardware security, particularly for inserting hardware trojans into complex hardware designs like CPUs [112]. They test how well LLMs can correlate high-level security concepts with specific hardware modules to overcome context length limitations. By analyzing reduced code bases and identifying modules with security-related features, they simplify the overall analysis. They demonstrate the LLM's capability to modify code and insert HT functionalities. The approach is validated by creating and testing an HT in a RISC-V micro-architecture on an FPGA board, showcasing the LLM's ability to aid in realistic HT attacks.

Complementing this work, Hayashi et al. developed a comprehensive dataset consisting of 290 Verilog examples generated using ChatGPT-4. This dataset [113], derived from 29 golden models and the TrustHub taxonomy, significantly enhances the limited existing datasets for hardware trojans, particularly for RISC-V and Web3 applications. It addresses the gap in research on Web3 hardware trojans, including modules like hardware wallets. This new dataset is intended to facilitate future research on defense mechanisms against hardware trojans in RISC-V systems, hardware wallets, and hardware Proof of Work (PoW) miners. Building on these advancements, Bhandari et al. introduced SENTAUR, a framework leveraging to generate, detect, and assess hardware Trojans in RTL designs [114], addressing security challenges in the IC supply chain. SENTAUR utilizes GPT-4 to create synthesizable RTL with embedded HTs, validated on the Xilinx FPGA platform. It explores various HT triggers and effects, demonstrating effectiveness in real-world applications like AES-T800 and dual-port RAM. A comparative analysis showed SENTAUR's superiority over existing tools in generating diverse and effective hypotheses. Experimental results highlighted SENTAUR's efficiency and robustness, demonstrating it to be a versatile toolchain for both attackers and defenders. Future work may extend its capabilities to post-synthesized netlist functionalities and various hardware platforms, further enhancing its adaptability and effectiveness.Expanding LLM applications into the analog domain, Chaudhuri et al. proposed LATENT [115], a feedback-guided agentic framework that autonomously inserts stealthy analog hardware Trojans into A/MS netlists using a Thought-Action-Observation loop and SPICE-based simulation feedback. LATENT achieves low activation ranges (15.7%) and area overhead (7.4%) while inducing significant output degradation (11.3%), outperforming fixed-pattern Trojans like A2 and DELTA. However, extending LATENT to support AC and small-signal behavior could improve its applicability to a broader range of analog threat scenarios.

*4) Logic Obfuscation:* Researchers have explored leveraging LLMs for logic obfuscation, with one notable work by Latibari et al.introducing Obfus-chat, an innovative framework utilizing GPT models to automate the obfuscation of hardware IPs [116]. This system takes hardware design netlists and key sizes as inputs, producing obfuscated code to enhance security. The framework's effectiveness is assessed using the Trust-Hub Obfuscation Benchmark, incorporating SAT attacks and functional verification to ensure security and design integrity.

*5) Side-Channel Attack:* Side-channel attacks (SCAs) remain a critical threat to hardware security, with power and cache-based attacks being among the most prominent. Researchers have leveraged LLMs and graph-based techniques to address side-channel vulnerabilities, particularly power and cache-based attacks. These approaches automate vulnerability detection, generate protective measures, and improve the interpretability of security analyzes, enabling early-stage vulnerability identification and reducing the need for costly post-silicon fixes. A significant focus has been on power side channel (PSC) attacks, which exploit power consumption patterns to leak sensitive information. Srivastava et al. addressed this challenge with SCAR, a pre-silicon PSC analysis framework that utilizes GNNs to convert designs into control-data flow graphs for detecting modules susceptible to side-channel leakage [117]. The framework includes a deep-learning-based explainer for providing human-accessible explanations of detection decisions and a fortification component that uses LLMs to generate and insert protective design code at vulnerable points. SCAR achieves significant figures of 94.49% localization accuracy, 100% precision, and 90.48% recall on various encryption algorithms, including AES, RSA, PRESENT, Saber, and CRYSTALS-Kyber. This significantly improves design robustness and efficiency by enabling early vulnerability detection and reducing the need for costly post-silicon fixes. Additionally, SCAR reduces features for GNN model training by 57% while maintaining comparable accuracy.

Cache-based side-channel attacks have also been a focal point, with researchers leveraging LLMs to model and evaluate secure cache architectures. He et al. propossed a novel probabilistic information flow graph to model interactions between victim and attacker programs and the cache architecture [118]. It introduces the Probability of Attack Success (PAS) metric to quantitatively assess a cache's resilience against various cache side-channel attacks. This model and metric are applied to evaluate nine different cache architectures across four classes of such attacks. LLMs are leveraged to automate the analysis and generation of potential attack scenarios. This approach enables verification and comparison of different secure cache architectures' resilience to side-channel attacks without requiring simulation or hardware implementation.

Table IV summarizes the latest contributions that leverage LLMs for secure hardware design. This table lists 13 influential works from 2023 and 2024, showing how researchers apply LLMs to various security applications, including vulnerability detection, SoC security, Trojan insertion, logic obfuscation, and side-channel analysis. The works utilize advanced AI models such as GPT-3.5 Turbo, GPT-4, HS-

TABLE IV
LLM-BASED SECURE HARDWARE DESIGN

| Year | Method Name | Vulnerability Detection | SoC Security | Trojan Insertion | Logic Obfuscation | Side Channel | AI Model |
|---|---|---|---|---|---|---|---|
| 2023 | Lin et al. [103] | ✓ | | | | | GPT-3.5-Turbo |
| 2023 | Thakur et al. [56] | | ✓ | | | | GPT-3.5-Turbo |
| 2023 | Saha et al. [106] | ✓ | ✓ | | | | GPT-3 and GPT-4 |
| 2023 | Meng et al. [110] | | ✓ | | | | HS-BERT |
| 2023 | kande et al. [108] | ✓ | | | | | code-davinci-002 |
| 2024 | Saha et al.(Vul-FSM) [104] | ✓ | ✓ | | | | GPT-3.5-Turbo |
| 2024 | Gadde et al. [107] | ✓ | | | | | GPT-3.5-Turbo |
| 2024 | Akyash et al. [105] | ✓ | ✓ | | | | Llama3-70B |
| 2024 | Tarek et al. [111] | | ✓ | | | | GPT-4 |
| 2024 | kardakis et al. [112] | | ✓ | ✓ | | | GPT-3.5/4 |
| 2024 | Hayashi et al. [113] | | | ✓ | | | GPT-4 |
| 2024 | Latibari et al. [116] | | | | ✓ | | GPT 3.5 |
| 2024 | Bhandari et al. [114] | | | ✓ | ✓ | | GPT-4 |
| 2024 | Srivastava et al. [117] | | ✓ | | | ✓ | GNN |
| 2025 | Chaudhuri et al.(LATENT) [115] | | | ✓ | | | GPT-4o-mini |

BERT, code-davinci-002, Llama-70B, and GNNs. Each paper showcases the ability of these models to tackle specific security challenges, enhancing hardware resilience and protecting against emerging threats. These contributions highlight how researchers increasingly rely on LLMs and attention-based methods to innovate and strengthen hardware security practices.

### *B. Other Attention-based Hardware Security*

Attention-based models have played a pivotal role in transforming hardware security, enabling more advanced threat detection and mitigation strategies.

A common theme across recent works is the use of graph-based representations. For instance, Chen et al. proposed TrojanFormer, a resource-efficient method for hardware Trojan detection using a graph transformer network. It converts HDL designs into graph structures to enhance accuracy and efficiency and employs a NodeFormer-based network with linear-complexity message passing and edge-regularization loss. TrojanFormer was evaluated on a Trust-Hub dataset with over 100,000 nodes, achieving a 97.66% F1 score on small and medium ICs. For large ICs, it showed a 4% accuracy improvement and an 18% reduction in computational overhead. Key limitations include difficulty capturing long-range dependencies in large ICs and reduced performance on circuits with sparse or incomplete graph topologies [119]. Another key direction is the integration of transformer models with graph-based pre-processing layers. For instance, Li e t al. proposed a transformer-based method that usages both transformer and Graph Convolutional Network as a pre-processing layer for detecting and localizing HTs at the RTL level and tested their architecture on the TrustHub dataset. They use RTL features to detect and localize HT attacks [120]. In another endeavor, Zhang et al. proposed a novel bit-level Hardware Trojan localization approach, namely B-HTRecognizer, which leveraged Graph Attention Networks. It converted the RTL Verilog code into bit-level edge-featured Data Flow Graphs to capture multi-dimensional feature representation, thus enhancing the precision of HT detection and localization. They presented TrustHub IMEex, a scaled open-source dataset for ML training. B-HTRecognizer achieved 84% precision, 93% recall for non-cross-designs, and 77% recall for cross-design tests on RISC-V, proving effective for various designs. It characterized the progress in automation and fine-grained localization for pre-silicon hardware security [121].

Finally, we note that few works have considered evaluating the quality of LLM-generated hardware designs from the security perspective, with recent work by Afsharmazayejani et al. providing an initial exploration of how such benchmarking could be undertaken [122].

## V. FUTURE DIRECTION

The application of attention-based models, especially LLMs, in hardware design and security represents a promising frontier with immense innovation potential. One significant research direction is the development of advanced code generation capabilities. While fine-tuning LLMs on domain-specific datasets for HDLs, such as Verilog, is already underway, further improvements can be achieved by leveraging larger and more diverse datasets. These models can improve code accuracy, functional correctness, and performance optimization and streamline the design process. Additionally, integrating multi-modal LLMs capable of processing textual descriptions, visual schematics, and design gate-level netlists could revolutionize hardware design workflows by providing a holistic and comprehensive approach to hardware representation. Automating the generation of detailed, error-free architectural specifications using attention-based models could reduce early design errors and significantly mitigate downstream challenges. In HLS, LLMs can be crucial in simplifying compatibility fixes and adaptive debugging, easing the transformation from C/C++ to HDL. This efficiency can make the overall process smoother and more reliable. In the domain of hardware security, LLMs show great promise in improving the safety and robustness of hardware systems. Secure HDL code generation, powered by domain-trained models, can address risks associated with hardware Trojans and IP piracy.

Furthermore, these models can identify and rectify vulnerabilities in existing designs, ensuring compliance with high-

assurance security standards. Automating formal verification tasks, such as generating SystemVerilog assertions and defining security properties, could accelerate the verification and validation phases. Attention-based models can also aid in detecting malicious modifications, such as hardware Trojans, and attack the design by inserting Trojans. LLMs can also be instrumental in automating EDA workflows. Potentially, autonomous agents powered by LLMs could oversee the entire design, testing, and debugging lifecycle with minimal human intervention. Tools like ChatEDA could evolve to encompass the full chip design process, from RTL development to manufacturing, ensuring seamless integration and scalability. Collaboration among multi-agent LLM systems could solve complex problems in hardware design and security, fostering greater intelligence and collaboration across the design ecosystem. Finally, advancing LLM capabilities in hardware design and security requires improvements in both data availability and model efficiency. Building comprehensive benchmark datasets that cover diversified security scenarios, high-performance designs, and real HDL issues will be critical for effective model training and evaluation. Additionally, lightweight and domain-specific attention-based models could be developed to address resource-constrained environments, such as edge devices or localized hardware labs. Collectively, these advancements have the potential to transform hardware design and security, creating systems that are more area-sparing, energy-efficient, and trustworthy. They can be used to educate the students, familiarize them with hardware design and security, and address workforce development.

## VI. Conclusion

Integrating attention-based models, especially LLMs, into hardware design and security is at a transformative stage, offering groundbreaking automation, efficiency, and robustness capabilities. This paper has presented a detailed investigation into how these models are applied to automate HDL generation, optimize chip design workflows, and enhance security mechanisms against threats such as hardware Trojans and side-channel attacks. The state-of-the-art survey shows the tremendous improvement made so far and the challenges ahead, such as data scarcity and the limitation of general-purpose models applied to hardware-specific tasks. Although the latest achievements in attention mechanisms and fine-tuned models have provided significant possibilities, challenges persist, including the need for domain-specific datasets and lightweight models.

## References


[1] K. Yang, H. Liu, Y. Zhao, and T. Deng, "A new design approach of hardware implementation through natural language entry," *IET Collaborative Intelligent Manufacturing*, vol. 5, no. 4, p. e12087, 2023.

[2] M. Tomlinson, J. Li, and A. Andreou, "Designing silicon brains using llm: Leveraging chatgpt for automated description of a spiking neuron array," 2024.

[3] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2023.

[4] B. Saber Latibari, S. Salehi, H. Homayoun, and A. Sasan, "Iret: Incremental resolution enhancing transformer," in *Proceedings of the Great Lakes Symposium on VLSI 2024*, ser. GLSVLSI '24. New York, NY, USA: Association for Computing Machinery, 2024, p. 620–625. [Online]. Available: https://doi.org/10.1145/3649476.3660380

[5] S. Kardakis, I. Perikos, F. Grivokostopoulou, and I. Hatzilygeroudis, "Examining attention mechanisms in deep learning models for sentiment analysis," *Applied Sciences*, vol. 11, no. 9, 2021. [Online]. Available: https://www.mdpi.com/2076-3417/11/9/3883

[6] B. S. Latibari, N. Nazari, M. A. Chowdhury, K. I. Gubbi, C. Fang, S. Ghimire, E. Hosseini, H. Sayadi, H. Homayoun, S. Salehi *et al.*, "Transformers: A security perspective," *IEEE Access*, 2024.

[7] OpenAI, "Gpt-4 technical report," 2024.

[8] J. Blocklove, S. Garg, R. Karri, and H. Pearce, "Chip-chat: Challenges and opportunities in conversational hardware design," in *2023 ACM/IEEE 5th Workshop on Machine Learning for CAD (MLCAD)*, 2023, pp. 1–6.

[9] S. Thakur, J. Blocklove, H. Pearce, B. Tan, S. Garg, and R. Karri, "Autochip: Automating hdl generation using llm feedback," *arXiv preprint arXiv:2311.04887*, 2023.

[10] B. Yu, "Machine learning in eda: When and how," in *2023 ACM/IEEE 5th Workshop on Machine Learning for CAD (MLCAD)*. IEEE, 2023, pp. 1–6.

[11] 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.

[12] L. Chen, "The dawn of ai-native eda: Promises and challenges of large circuit models," *arXiv preprint arXiv:2403.07257*, 2024.

[13] A. Sharma, T.-D. Ene, K. Kunal, M. Liu, Z. Hasan, and H. Ren, "Assessing economic viability: A comparative analysis of total cost of ownership for domain-adapted large language models versus state-of-the-art counterparts in chip design coding assistance," in *2024 IEEE LLM Aided Design Workshop (LAD)*, 2024, pp. 1–6.

[14] N. Wu, Y. Li, H. Yang, H. Chen, S. Dai, C. Hao, C. Yu, and Y. Xie, "Survey of machine learning for software-assisted hardware design verification: Past, present, and prospect," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 29, no. 4, Jun. 2024. [Online]. Available: https://doi.org/10.1145/3661308

[15] J. Yang, "Harnessing the power of llms in practice: A survey on chatgpt and beyond," 2023.

[16] M. Akyash, "Evolutionary large language models for hardware security: A comparative survey," 2024.

[17] H. Pearce and B. Tan, "Large language models for hardware security (invited, short paper)," in *2024 IEEE 6th International Conference on Trust, Privacy and Security in Intelligent Systems, and Applications (TPS-ISA)*, 2024, pp. 420–423.

[18] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," Jan. 2015, 3rd International Conference on Learning Representations, ICLR 2015 ; Conference date: 07-05-2015 Through 09-05-2015.

[19] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," 2018. [Online]. Available: https://arxiv.org/abs/1710.10903

[20] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, J. Burstein, C. Doran, and T. Solorio, Eds. Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 4171–4186. [Online]. Available: https://aclanthology.org/N19-1423/

[21] M. Tehranipoor, K. Zamiri Azar, N. Asadizanjani, F. Rahman, H. Mardani Kamali, and F. Farahmandi, "Large language models for soc security," in *Hardware Security: A Look into the Future*. Springer, 2024, pp. 255–299.

[22] "Ai is reshaping chip design. but where will it ends?" https://www-forbes-com.cdn.ampproject.org/c/s/www.forbes.com/sites/karlfreund/2023/12/19/ai-is-reshaping-chip-design-but-where-will-it-end/amp/, accessed: 2024-01-22.

[23] Y. Lu, S. Liu, Q. Zhang, and Z. Xie, "Rtllm: An open-source benchmark for design rtl generation with large language model," in *Proceedings of the 29th Asia and South Pacific Design Automation Conference*, ser. ASPDAC '24. IEEE Press, 2024, p. 722–727. [Online]. Available: https://doi.org/10.1109/ASP-DAC58780.2024.10473904

[24] M. Liu, N. Pinckney, B. Khailany, and H. Ren, "Verilogeval: Evaluating large language models for verilog code generation," in *2023 IEEE/ACM International Conference on Computer Aided Design (ICCAD)*. IEEE, 2023, pp. 1–8.

[25] Y. Ye, T. Chen, Y. Gao, H. Yan, B. Yu, and L. Shi, "Graph-learning-driven path-based timing analysis results predictor from graph-based timing analysis," in *Proceedings of the 28th Asia and South Pacific Design Automation Conference*, 2023, pp. 547–552.

[26] T. Chowdhury, A. Vakil, B. Saber Latibari, S. Aresh Beheshti Shirazi, A. Mirzaeian, X. Guo, S. Manoj PD, H. Homayoun, I. Savidis, L. Zhao *et al.*, "Rapta: A hierarchical representation learning solution for real-time prediction of path-based static timing analysis," in *Proceedings of the Great Lakes Symposium on VLSI 2022*, 2022, pp. 493–500.

[27] S. A. Beheshti-Shirazi, N. Nazari, K. I. Gubbi, B. S. Latibari, S. Rafatirad, H. Homayoun, A. Sasan, and P. S. Manoj, "Advanced reinforcement learning solution for clock skew engineering: Modified q-table update technique for peak current and ir drop minimization," *IEEE Access*, vol. 11, pp. 87 869–87 886, 2023.

[28] B. Yang, "Floorplanning with edge-aware graph attention network and hindsight experience replay," *ACM Transactions on Design Automation of Electronic Systems*, 2024.

[29] W. Fang, M. Li, M. Li, Z. Yan, S. Liu, H. Zhang, and Z. Xie, "Assertllm: Generating and evaluating hardware verification assertions from design specifications via multi-llms," 2024.

[30] K. Chang, Y. Wang, H. Ren, M. Wang, S. Liang, Y. Han, H. Li, and X. Li, "Chipgpt: How far are we from natural language hardware design," 2023. [Online]. Available: https://arxiv.org/abs/2305.14019

[31] C. Lv, Z. Wei, W. Qian, J. Ye, C. Feng, and Z. He, "Gpt-ls: Generative pre-trained transformer with offline reinforcement learning for logic synthesis," in *2023 IEEE 41st International Conference on Computer Design (ICCD)*, 2023, pp. 320–326.

[32] Z. Guo, M. Liu, J. Gu, S. Zhang, D. Z. Pan, and Y. Lin, "A timing engine inspired graph neural network model for pre-routing slack prediction," in *Proceedings of the 59th ACM/IEEE Design Automation Conference*, ser. DAC '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 1207–1212. [Online]. Available: https://doi.org/10.1145/3489517.3530597

[33] J. Maynard and A. Rezaei, "Reconfigurable run-time hardware trojan mitigation for logic-locked circuits," in *2024 IEEE 17th Dallas Circuits and Systems Conference (DCAS)*. IEEE, 2024, pp. 1–6.

[34] R. Vishwakarma and A. Rezaei, "Uncertainty-aware hardware trojan detection using multimodal deep learning," in *2024 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2024, pp. 1–6.

[35] Y. Aghamohammadi and A. Rezaei, "Lipstick: Corruptibility-aware and explainable graph neural network-based oracle-less attack on logic locking," in *2024 29th Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE, 2024, pp. 606–611.

[36] R. Vishwakarma and A. Rezaei, "Risk-aware and explainable framework for ensuring guaranteed coverage in evolving hardware trojan detection," in *2023 IEEE/ACM International Conference on Computer Aided Design (ICCAD)*. IEEE, 2023, pp. 01–09.

[37] K. I. Gubbi, B. S. Latibari, M. A. Chowdhury, A. Jalilzadeh, E. Y. Hamedani, S. Rafatirad, A. Sasan, H. Homayoun, and S. Salehi, "Optimized and automated secure ic design flow: A defense-in-depth approach," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 71, no. 5, pp. 2031–2044, 2024.

[38] S. Liu, W. Fang, Y. Lu, Q. Zhang, H. Zhang, and Z. Xie, "Rtlcoder: Outperforming gpt-3.5 in design rtl generation with our open-source dataset and lightweight solution," in *2024 IEEE LLM Aided Design Workshop (LAD)*, 2024, pp. 1–5.

[39] Y. Tsai, M. Liu, and H. Ren, "Rtlfixer: Automatically fixing rtl syntax errors with large language model," in *Proceedings of the 61st ACM/IEEE Design Automation Conference*, ser. DAC '24. New York, NY, USA: Association for Computing Machinery, 2024. [Online]. Available: https://doi.org/10.1145/3649329.3657353

[40] A. Sarihi, A. Patooghy, P. Jamieson, and A.-H. A. Badawy, "Hardware trojan insertion using reinforcement learning," 2022.

[41] J. Cruz, P. Gaikwad, A. Nair, P. Chakraborty, and S. Bhunia, "Automatic hardware trojan insertion using machine learning," 2022.

[42] S. R. Zantout, *Hardware Trojan Detection in FPGA through Side-Channel Power Analysis and Machine Learning*. University of California, Irvine, 2018.

[43] M. Mushtaq, M. A. Mukhtar, V. Lapotre, M. K. Bhatti, and G. Gogniat, "Winter is here! a decade of cache-based side-channel attacks, detection & mitigation for rsa," *Information Systems*, vol. 92, p. 101524, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0306437920300338

[44] H. Wang, H. Sayadi, A. Sasan, S. Rafatirad, T. Mohsenin, and H. Homayoun, "Comprehensive evaluation of machine learning countermeasures for detecting microarchitectural side-channel attacks," in *Proceedings of the 2020 on Great Lakes Symposium on VLSI*, ser. GLSVLSI '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 181–186. [Online]. Available: https://doi.org/10.1145/3386263.3407586

[45] T. Lavanya and K. Rajalakshmi, "A review paper on machine learning based trojan detection in the iot chips," in *Internet of Things and Connected Technologies*, R. Misra, N. Kesswani, M. Rajarajan, B. Veeravalli, and A. Patel, Eds. Cham: Springer International Publishing, 2022, pp. 225–238.

[46] K. Liakos, G. Georgakilas, and F. Plessas, *Hardware and System Security: Attacks and Countermeasures Against Hardware Trojans*. Cham: Springer International Publishing, 2023, pp. 501–549. [Online]. Available: https://doi.org/10.1007/978-3-031-16344-9_13

[47] Z. Pan and P. Mishra, "A survey on hardware vulnerability analysis using machine learning," *IEEE Access*, vol. 10, pp. 49 508–49 527, 2022.

[48] R. Mukherjee, S. R. Rajendran, and R. S. Chakraborty, "A comprehensive survey of physical and logic testing techniques for hardware trojan detection and prevention," *Journal of Cryptographic Engineering*, vol. 12, no. 4, pp. 495–522, Nov 2022. [Online]. Available: https://doi.org/10.1007/s13389-022-00295-w

[49] K. I. Gubbi, B. Saber Latibari, A. Srikanth, T. Sheaves, S. A. Beheshti-Shirazi, S. M. PD, S. Rafatirad, A. Sasan, H. Homayoun, and S. Salehi, "Hardware trojan detection using machine learning: A tutorial," *ACM Trans. Embed. Comput. Syst.*, vol. 22, no. 3, apr 2023. [Online]. Available: https://doi.org/10.1145/3579823

[50] V. Gohil, H. Guo, S. Patnaik, and J. Rajendran, "Attrition: Attacking static hardware trojan detection techniques using reinforcement learning," in *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 1275–1289. [Online]. Available: https://doi.org/10.1145/3548606.3560690

[51] K. G. Liakos, G. K. Georgakilas, S. Moustakidis, N. Sklavos, and F. C. Plessas, "Conventional and machine learning approaches as countermeasures against hardware trojan attacks," *Microprocessors and Microsystems*, vol. 79, p. 103295, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0141933120304543

[52] A. Tauhid, L. Xu, M. Rahman, and E. Tomai, "A survey on security analysis of machine learning-oriented hardware and software intellectual property," *High-Confidence Computing*, vol. 3, no. 2, p. 100114, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2667295223000120

[53] 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," 2022.

[54] K. Thorat, J. Zhao, Y. Liu, H. Peng, X. Xie, B. Lei, J. Zhang, and C. Ding, "Advanced language model-driven verilog development: Enhancing power, performance, and area optimization in code synthesis," *arXiv preprint arXiv:2312.01022*, 2023.

[55] M. E. Yanik, I. Cicek, and E. Afacan, "Shortcircuit: An open-source chatgpt driven digital integrated circuit front-end design automation tool," in *2023 30th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, 2023, pp. 1–4.

[56] 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," 2023.

[57] M. Li, W. Fang, Q. Zhang, and Z. Xie, "Specllm: Exploring generation and review of vlsi design specification with large language model," *arXiv preprint arXiv:2401.13266*, 2024.

[58] B. Mali, K. Maddala, S. Reddy, V. Gupta, C. Karfa, and R. Karri, "Chiraag: Chatgpt informed rapid and automated assertion generation," 2024.

[59] M. DeLorenzo, A. B. Chowdhury, V. Gohil, S. Thakur, R. Karri, S. Garg, and J. Rajendran, "Make every move count: Llm-based high-quality rtl code generation using mcts," 2024.

[60] M. Hassan, S. Ahmadi-Pour, K. Qayyum, C. K. Jha, and R. Drechsler, "Llm-guided formal verification coupled with mutation testing," in *2024 Design, Automation and Test in Europe Conference and Exhibition (DATE)*, 2024, pp. 1–2.

[61] L. J. Wan, Y. Huang, Y. Li, H. Ye, J. Wang, X. Zhang, and D. Chen, "Invited paper: Software/hardware co-design for llm and its application for design verification," in *2024 29th Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2024, pp. 435–441.

[62] X. Yao, "Hdldebugger: Streamlining hdl debugging with large language models," *arXiv preprint arXiv:2403.11671*, 2024.

[63] K. Chang, "Data is all you need: Finetuning llms for chip design via an automated design-data augmentation framework," *arXiv preprint arXiv:2403.11202*, 2024.

[64] Z. Wang, Y. Shen, X. Yao, W. Zhao, Y. Bai, F. Farnia, and B. Yu, "Chatpattern: Layout pattern customization via natural language," in *Proceedings of the 61st ACM/IEEE Design Automation Conference*, ser. DAC '24. New York, NY, USA: Association for Computing Machinery, 2024. [Online]. Available: https://doi.org/10.1145/3649329.3657361

[65] Anonymous, "Improving large language model hardware generating quality through post-LLM search," in *Machine Learning for Systems 2023*, 2023. [Online]. Available: https://openreview.net/forum?id=IY7M6sqCxq

[66] M. DeLorenzo, "Creativeval: Evaluating creativity of llm-based hardware code generation," *arXiv preprint arXiv:2404.08806*, 2024.

[67] M. Nazzal, "A dataset for large language model-driven ai accelerator generation," 2024.

[68] P. Vitolo, "Natural language to verilog: Design of a recurrent spiking neural network using large language models and chatgpt," *arXiv preprint arXiv:2405.01419*, 2024.

[69] M. Xiang, "Digital asic design with ongoing llms: Strategies and prospects," *arXiv preprint arXiv:2405.02329*, 2024.

[70] J. Blocklove, S. Garg, R. Karri, and H. Pearce, "Evaluating llms for hardware design and test," in *2024 IEEE LLM Aided Design Workshop (LAD)*, 2024, pp. 1–6.

[71] S. Liu, "Rtlcoder: Outperforming gpt-3.5 in design rtl generation with our open-source dataset and lightweight solution," 2024.

[72] B. Nadimi, "A multi-expert large language model architecture for verilog code generation," 2024.

[73] B. Ahmad, S. Thakur, B. Tan, R. Karri, and H. Pearce, "On hardware security bug code fixes by prompting large language models," *IEEE Transactions on Information Forensics and Security*, vol. 19, pp. 4043–4057, 2024.

[74] W. Fu, S. Li, Y. Zhao, H. Ma, R. Dutta, X. Zhang, K. Yang, Y. Jin, and X. Guo, "Hardware phi-1.5b: A large language model encodes hardware domain specific knowledge," in *Proceedings of the 29th Asia and South Pacific Design Automation Conference*, ser. ASPDAC '24. IEEE Press, 2024, p. 349–354. [Online]. Available: https://doi.org/10.1109/ASP-DAC58780.2024.10473927

[75] J. T. Meech, "Leveraging high-level synthesis and large language models to generate, simulate, and deploy a uniform random number generator hardware design," 2024. [Online]. Available: https://arxiv.org/abs/2311.03489

[76] J. Liu, A. Sharma, C. Doumbia, and J. K. S. Poon, "Towards large-language model assisted layout of silicon photonic integrated circuits," in *The 25th European Conference on Integrated Optics*, J. Witzens, J. Poon, L. Zimmermann, and W. Freude, Eds. Cham: Springer Nature Switzerland, 2024, pp. 441–447.

[77] K. Chang, Z. Chen, Y. Zhou, W. Zhu, kun wang, H. Xu, C. Li, M. Wang, S. Liang, H. Li, Y. Han, and Y. Wang, "Natural language is not enough: Benchmarking multi-modal generative ai for verilog generation," 2024. [Online]. Available: https://arxiv.org/abs/2407.08473

[78] Y. Zhao, D. Huang, C. Li, P. Jin, Z. Nan, T. Ma, L. Qi, Y. Pan, Z. Zhang, R. Zhang, X. Zhang, Z. Du, Q. Guo, X. Hu, and Y. Chen, "Codev: Empowering llms for verilog generation through multi-level summarization," 2024. [Online]. Available: https://arxiv.org/abs/2407.10424

[79] Z. Jiang, Q. Zhang, C. Liu, H. Li, and X. Li, "Iicpilot: An intelligent integrated circuit backend design framework using open eda," 2024. [Online]. Available: https://arxiv.org/abs/2407.12576

[80] M. Abdelatty and S. Reda, "Hdlcopilot: Hardware design library querying with natural language," 2024. [Online]. Available: https://arxiv.org/abs/2407.12749

[81] Z. He, H. Wu, X. Zhang, X. Yao, S. Zheng, H. Zheng, and B. Yu, "Chateda: A large language model powered autonomous agent for eda," 2024. [Online]. Available: https://arxiv.org/abs/2308.10204

[82] C.-T. Ho and H. Ren, "Large language model (llm) for standard cell layout design optimization," 2024. [Online]. Available: https://arxiv.org/abs/2406.06549

[83] M. Gao, J. Zhao, Z. Lin, W. Ding, X. Hou, Y. Feng, C. Li, and M. Guo, "Autovcoder: A systematic framework for automated verilog code generation using llms," 2024. [Online]. Available: https://arxiv.org/abs/2407.18333

[84] A. Nakkab, S. Q. Zhang, R. Karri, and S. Garg, "Rome was not built in a single step: Hierarchical prompting for llm-based chip design," 2024. [Online]. Available: https://arxiv.org/abs/2407.18276

[85] L. Collini, S. Garg, and R. Karri, "C2hlsc: Can llms bridge the software-to-hardware design gap?" *arXiv preprint arXiv:2406.09233*, 2024.

[86] A. B. Chowdhury, M. Romanelli, B. Tan, R. Karri, and S. Garg, "Retrieval-guided reinforcement learning for boolean circuit minimization," in *The Twelfth International Conference on Learning Representations*, 2024. [Online]. Available: https://openreview.net/forum?id=0t1O8ziRZp

[87] R. Qiu, G. L. Zhang, R. Drechsler, U. Schlichtmann, and B. Li, "Autobench: Automatic testbench generation and evaluation using llms for hdl design," in *Proceedings of the 2024 ACM/IEEE International Symposium on Machine Learning for CAD*. New York, NY, USA: Association for Computing Machinery, 2024. [Online]. Available: https://doi.org/10.1145/3670474.3685956

[88] S. Qiu, B. Tan, and H. Pearce, "Llm-aided explanations of eda synthesis errors," in *2024 IEEE LLM Aided Design Workshop (LAD)*, 2024, pp. 1–6.

[89] 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," 2024. [Online]. Available: https://arxiv.org/abs/2405.14918

[90] C.-C. Chang, Y. Shan, S. Fan, J. Li, S. Zhang, N. Cao, Y. Chen, and X. Zhang, "Lamagic: Language-model-based topology generation for analog integrated circuits," 2024. [Online]. Available: https://arxiv.org/abs/2407.18269

[91] G. Chen, K. Zhu, S. Kim, H. Zhu, Y. Lai, B. Yu, and D. Z. Pan, "Llm-enhanced bayesian optimization for efficient analog layout constraint generation," 2024. [Online]. Available: https://arxiv.org/abs/2406.05250

[92] Y. Yin, Y. Wang, B. Xu, and P. Li, "Ado-llm: Analog design bayesian optimization with in-context learning of large language models," 2024. [Online]. Available: https://arxiv.org/abs/2406.18770

[93] 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," 2024. [Online]. Available: https://arxiv.org/abs/2406.18873

[94] Y. Liu, Z. Ju, Z. Li, M. Dong, H. Zhou, J. Wang, F. Yang, X. Zeng, and L. Shang, "Floorplanning with graph attention," in *Proceedings of the 59th ACM/IEEE Design Automation Conference*, ser. DAC '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 1303–1308. [Online]. Available: https://doi.org/10.1145/3489517.3530484

[95] T. Mohamed, V. M. van Santen, L. Alrahis, O. Sinanoglu, and H. Amrouch, "Graph attention networks to identify the impact of transistor degradation on circuit reliability," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 71, no. 7, pp. 3269–3281, 2024.

[96] V. Lee, C. Deng, L. Elzeiny, P. Abbeel, and J. Wawrzynek, "Chip placement with diffusion," 2024. [Online]. Available: https://arxiv.org/abs/2407.12282

[97] X. Li, X. Li, L. Chen, X. Zhang, M. Yuan, and J. Wang, "Circuit transformer: End-to-end circuit design by predicting the next gate," 2024. [Online]. Available: https://arxiv.org/abs/2403.13838

[98] C.-T. Ho, A. Chandna, D. Guan, A. Ho, M. Kim, Y. Li, and H. Ren, "Novel transformer model based clustering method for standard cell design automation," in *Proceedings of the 2024 International Symposium on Physical Design*, ser. ISPD '24. New York, NY, USA: Association for Computing Machinery, 2024, p. 195–203. [Online]. Available: https://doi.org/10.1145/3626184.3633314

[99] C. Deng, Z. Yue, C. Yu, G. Sarar, R. Carey, R. Jain, and Z. Zhang, "Less is more: Hop-wise graph attention for scalable and generalizable learning on circuits," 2024. [Online]. Available: https://arxiv.org/abs/2403.01317

[100] B. Zhu, R. Chen, X. Zhang, F. Yang, X. Zeng, B. Yu, and M. D. Wong, "Hotspot detection via multi-task learning and transformer encoder," in *2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, 2021, pp. 1–8.

[101] Y. Lai, J. Liu, Z. Tang, B. Wang, J. Hao, and P. Luo, "Chipformer: Transferable chip placement via offline decision transformer," 2023. [Online]. Available: https://arxiv.org/abs/2306.14744

[102] A. Mehradfar, X. Zhao, Y. Niu, S. Babakniya, M. Alesheikh, H. Aghasi, and S. Avestimehr, "Aicircuit: A multi-level dataset and benchmark for ai-driven analog integrated circuit design," 2024. [Online]. Available: https://arxiv.org/abs/2407.18272

[103] Y.-Z. Lin, M. Mamun, M. A. Chowdhury, S. Cai, M. Zhu, B. S. Latibari, K. I. Gubbi, N. N. Bavarsad, A. Caputo, A. Sasan *et al.*, "Hw-v2w-map: Hardware vulnerability to weakness mapping framework for root cause analysis with gpt-assisted mitigation suggestion," 2023.

[104] 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 *2024 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, 2024, pp. 233–243.
[105] M. Akyash, "Self-hwdebug: Automation of llm self-instructing for hardware security verification," 2024.
[106] D. Saha, "Llm for soc security: A paradigm shift," 2023.
[107] D. N. Gadde, A. Kumar, T. Nalapat, E. Rezunov, and F. Cappellini, "All artificial, less intelligence: Genai through the lens of formal verification," 2024. [Online]. Available: https://arxiv.org/abs/2403.16750
[108] R. Kande, H. Pearce, B. Tan, B. Dolan-Gavitt, S. Thakur, R. Karri, and J. Rajendran, "Llm-assisted generation of hardware assertions," 2023.
[109] 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*, vol. 33, no. 4, pp. 1118–1131, 2025.
[110] X. Meng, "Unlocking hardware security assurance: The potential of llms," 2023.
[111] S. Tarek, D. Saha, S. K. Saha, M. Tehranipoor, and F. Farahmandi, "Socurellm: An llm-driven approach for large-scale system-on-chip security verification and policy generation," *Cryptology ePrint Archive*, 2024.
[112] 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.
[113] V. T. Hayashi and W. V. Ruggiero, "Hardware trojan dataset of risc-v and web3 generated with chatgpt-4," *Data*, vol. 9, no. 6, p. 82, 2024.
[114] J. Bhandari, R. Sadhukhan, P. Krishnamurthy, F. Khorrami, and R. Karri, "Sentaur: Security enhanced trojan assessment using llms against undesirable revisions," 2024. [Online]. Available: https://arxiv.org/abs/2407.12352
[115] J. Chaudhuri, A. Chaudhuri, and K. Chakrabarty, "Latent: Llm-augmented trojan insertion and evaluation framework for analog netlist topologies," 2025. [Online]. Available: https://arxiv.org/abs/2505.06364
[116] B. S. Latibari, S. Ghimire, M. A. Chowdhury, N. Nazari, K. I. Gubbi, H. Homayoun, A. Sasan, and S. Salehi, "Automated hardware logic obfuscation framework using gpt," in *2024 IEEE 17th Dallas Circuits and Systems Conference (DCAS)*, 2024, pp. 1–5.
[117] A. Srivastava, S. Das, N. Choudhury, R. Psiakis, P. H. Silva, D. Pal, and K. Basu, "Scar: Power side-channel analysis at rtl level," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 32, no. 6, pp. 1110–1123, 2024.
[118] Z. He and R. B. Lee, "How secure is your cache against side-channel attacks?" in *Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture*, 2017, pp. 341–353.
[119] M. Chen, X. Kou, and G. Zhang, "Trojanformer: Resource-efficient hardware trojan detection using graph transformer network," in *2024 7th International Conference on Electronics Technology (ICET)*, 2024, pp. 165–170.
[120] Y. Li, S. Li, and H. Shen, "Htrans: Transformer-based method for hardware trojan detection and localization," in *2023 IEEE 32nd Asian Test Symposium (ATS)*, 2023, pp. 1–6.
[121] H. Zhang, Z. Fan, Y. Zhou, and Y. Li, "B-htrecognizer: Bit-wise hardware trojan localization using graph attention networks," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pp. 1–1, 2024.
[122] R. Afsharmazayejani, M. M. Shahmiri, P. Link, H. Pearce, and B. Tan, "Toward hardware security benchmarking of llms," in *2024 IEEE LLM Aided Design Workshop (LAD)*, 2024, pp. 1–7.
[123] M. U. Hadi, al tashi, R. Qureshi, A. Shah, M. Irfan, A. Zafar, M. B. Shaikh, N. Akhtar, J. Wu, S. Mirjalili, Q. Al-Tashi, A. Muneer, M. A. Al-garadi, G. Cnn, and T. RoBERTa, "Large language models: A comprehensive survey of its applications, challenges, limitations, and future prospects." [Online]. Available: https://api.semanticscholar.org/CorpusID:266378240
[124] Z. Wang, L. Alrahis, L. Mankali, J. Knechtel, and O. Sinanoglu, "Llms and the future of chip design: Unveiling security risks and building trust," 2024. [Online]. Available: https://arxiv.org/abs/2405.07061
[125] M. Liu, T.-D. Ene, R. Kirby, C. Cheng, N. Pinckney, R. Liang, J. Alben, H. Anand, S. Banerjee, I. Bayraktaroglu, B. Bhaskaran, B. Catanzaro, A. Chaudhuri, S. Clay, B. Dally, L. Dang, P. Deshpande, S. Dhodhi, S. Halepete, E. Hill, J. Hu, S. Jain, A. Jindal, B. Khailany, G. Kokai, K. Kunal, X. Li, C. Lind, H. Liu, S. Oberman, S. Omar, G. Pasandi, S. Pratty, J. Raiman, A. Sarkar, Z. Shao, H. Sun, P. P. Suthar, V. Tej, W. Turner, K. Xu, and H. Ren, "Chipnemo: Domain-adapted llms for chip design," 2024.
[126] R. Kande, V. Gohil, M. DeLorenzo, C. Chen, and J. Rajendran, "Llms for hardware security: Boon or bane?" in *2024 IEEE 42nd VLSI Test Symposium (VTS)*. Los Alamitos, CA, USA: IEEE Computer Society, apr 2024, pp. 1–4. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/VTS60656.2024.10538871
[127] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," 2020.
[128] Z. Wei, "Towards transferable adversarial attacks on vision transformers," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 3, 2022, pp. 2668–2676.
[129] Y.-C. Lin, "Novel preprocessing technique for data embedding in engineering code generation using large language model," 2024.
[130] B.-Y. Wu, "Eda corpus: A large language model dataset for enhanced interaction with openroad," 2024.
[131] 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," 2024.
[132] S. A. Sheikholeslam and A. Ivanov, "Synthai: A multi agent generative ai framework for automated modular hls design generation," 2024.
[133] M. A. Khair, J. R. P. K. Ande, D. R. Goda, and S. R. Yerram, "Secure vlsi design: Countermeasures against hardware trojans and side-channel attacks," *Engineering International*, vol. 7, p. 147–160, 12 2019.
[134] H. Huang, Z. Lin, Z. Wang, X. Chen, K. Ding, and J. Zhao, "Towards llm-powered verilog rtl assistant: Self-verification and self-correction," 2024.
[135] A. Alaql, S. Chattopadhyay, P. Chakraborty, T. Hoque, and S. Bhunia, "Lego: A learning-guided obfuscation framework for hardware ip protection," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 41, no. 4, pp. 854–867, 2022.
[136] A. Nahiyan, K. Xiao, K. Yang, Y. Jin, D. Forte, and M. Tehranipoor, "Avfsm: A framework for identifying and mitigating vulnerabilities in fsms," in *Proceedings of the 53rd Annual Design Automation Conference*, 2016, pp. 1–6.
[137] R. Hassan, G. Kolhe, S. Rafatirad, H. Homayoun, and S. M. P. Dinakarrao, "A neural network-based cognitive obfuscation toward enhanced logic locking," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 41, no. 11, pp. 4587–4599, 2021.
[138] S. Paria, A. Dasgupta, and S. Bhunia, "Navigating soc security landscape on llm-guided paths," in *Proceedings of the Great Lakes Symposium on VLSI 2024*, ser. GLSVLSI '24. New York, NY, USA: Association for Computing Machinery, 2024, p. 252–257. [Online]. Available: https://doi.org/10.1145/3649476.3660393
[139] M. Shao, S. Jancheska, M. Udeshi, B. Dolan-Gavitt, H. Xi, K. Milner, B. Chen, M. Yin, S. Garg, P. Krishnamurthy, F. Khorrami, R. Karri, and M. Shafique, "Nyu ctf dataset: A scalable open-source benchmark dataset for evaluating llms in offensive security," 2024. [Online]. Available: https://arxiv.org/abs/2406.05590
[140] S. S. Miftah, A. Srivastava, H. Kim, and K. Basu, "Assert-o: Context-based assertion optimization using llms," in *Proceedings of the Great Lakes Symposium on VLSI 2024*, ser. GLSVLSI '24. New York, NY, USA: Association for Computing Machinery, 2024, p. 233–239. [Online]. Available: https://doi.org/10.1145/3649476.3660378
[141] F. Cui, C. Yin, K. Zhou, Y. Xiao, G. Sun, Q. Xu, Q. Guo, D. Song, D. Lin, X. Zhang, Yun, and Liang, "Origen:enhancing rtl code generation with code-to-code augmentation and self-reflection," 2024. [Online]. Available: https://arxiv.org/abs/2407.16237
[142] Y. Zhang, Z. Yu, Y. Fu, C. Wan, and Y. C. Lin, "Mg-verilog: Multi-grained dataset towards enhanced llm-assisted verilog generation," 2024. [Online]. Available: https://arxiv.org/abs/2407.01910
[143] Z. He and B. Yu, "Large language models for eda: Future or mirage?" in *Proceedings of the 2024 International Symposium on Physical Design*, ser. ISPD '24. New York, NY, USA: Association for Computing Machinery, 2024, p. 65–66. [Online]. Available: https://doi.org/10.1145/3626184.3639700
[144] N. Rouf, F. Amin, and P. D. Franzon, "Can low-rank knowledge distillation in llms be useful for microelectronic reasoning?" 2024. [Online]. Available: https://arxiv.org/abs/2406.13808
[145] X. Li, X. Li, L. Chen, X. Zhang, M. Yuan, and J. Wang, "Logic synthesis with generative deep neural networks," 2024. [Online]. Available: https://arxiv.org/abs/2406.04699

[146] A. B. Chowdhury, M. Romanelli, B. Tan, R. Karri, and S. Garg, "Retrieval-guided reinforcement learning for boolean circuit minimization," 2024. [Online]. Available: https://arxiv.org/abs/2401.12205

[147] D. Saha, S. Tarek, K. Yahyaei, S. K. Saha, J. Zhou, M. Tehranipoor, and F. Farahmandi, "Llm for soc security: A paradigm shift," 2023. [Online]. Available: https://arxiv.org/abs/2310.06046

[148] Y. Pu, Z. He, T. Qiu, H. Wu, and B. Yu, "Customized retrieval augmented generation and benchmarking for eda tool documentation qa," 2024. [Online]. Available: https://arxiv.org/abs/2407.15353

[149] N. Pinckney, C. Batten, M. Liu, H. Ren, and B. Khailany, "Revisiting verilogeval: Newer llms, in-context learning, and specification-to-rtl tasks," 2024. [Online]. Available: https://arxiv.org/abs/2408.11053

[150] E. Nijkamp, B. Pang, H. Hayashi, L. Tu, H. Wang, Y. Zhou, S. Savarese, and C. Xiong, "Codegen: An open large language model for code with multi-turn program synthesis," 2023. [Online]. Available: https://arxiv.org/abs/2203.13474

[151] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun, "Graph neural networks: A review of methods and applications," *AI Open*, vol. 1, pp. 57–81, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2666651021000012

[152] N. Nazari, H. M. Makrani, C. Fang, H. Sayadi, S. Rafatirad, K. N. Khasawneh, and H. Homayoun, "Forget and rewire: Enhancing the resilience of transformer-based models against Bit-Flip attacks," in *33rd USENIX Security Symposium (USENIX Security 24)*. Philadelphia, PA: USENIX Association, Aug. 2024, pp. 1349–1366. [Online]. Available: https://www.usenix.org/conference/usenixsecurity24/presentation/nazari