# Harnessing the Potential of Deep-learning Algorithms and Generative AI for SoC and Chiplet Design and Verification

**Other titles in Foundations and Trends® in Electronic Design Automation**

*Polynomial Formal Verification of Arithmetic Circuits*
Alireza Mahzoon and Rolf Drechsler
ISBN: 978-1-63828-404-8

*Cloud and Edge Computing for Connected and Automated Vehicles*
Qi Zhu, Bo Yu, Ziran Wang, Jie Tang, Qi Alfred Chen, Zihao Li, Xiangguo Liu, Yunpeng Luo and Lingzi Tu
ISBN: 978-1-63828-302-7

*From CNN to DNN Hardware Accelerators: A Survey on Design, Exploration, Simulation, and Frameworks*
Leonardo Rezende Juracy, Rafael Garibotti and Fernando Gehm Moraes
ISBN: 978-1-63828-162-7

*Self-Powered Wearable IoT Devices for Health and Activity Monitoring*
Ganapati Bhat, Ujjwal Gupta, Yigit Tuncel, Fatih Karabacak, Sule Ozev and Umit Y. Ogras
ISBN: 978-1-68083-748-3

*On-Chip Dynamic Resource Management*
Antonio Miele, Anil Kanduri, Kasra Moazzemi, Dávid Juhász, Amir R. Rahmani, Nikil Dutt, Pasi Liljeberg and Axel Jantsch
ISBN: 978-1-68083-578-6

*Smart Healthcare*
Hongxu Yin, Ayten Ozge Akmandor, Arsalan Mosenia and Niraj K. Jha
ISBN: 978-1-68083-440-6

# Harnessing the Potential of Deep-learning Algorithms and Generative AI for SoC and Chiplet Design and Verification

Edited by

Imed Ben Dhaou

Hannu Tenhunen

Ahmed Abdelgawad

Sree Ranjani Rajendran

Rajat Subhra Chakraborty

now

the essence of knowledge

Boston — Delft

# Foundations and Trends® in Electronic Design Automation

*Published, sold and distributed by:*
now Publishers Inc.
PO Box 1024
Hanover, MA 02339
United States
Tel. +1-781-985-4510
www.nowpublishers.com
sales@nowpublishers.com

*Outside North America:*
now Publishers Inc.
PO Box 179
2600 AD Delft
The Netherlands
Tel. +31-6-51115274

The content of the book was originally published in Foundations and Trends® in Electronic Design Automation, vol. 14, no. 4.

ISBN: 978-1-63828-538-0

**Foundations and Trends® in Electronic Design Automation**
Volume 14, Issue 4, 2025

# Editorial Board

# Editorial Scope

Foundations and Trends® in Electronic Design Automation publishes survey and tutorial articles in the following topics:

- System Level Design
- Behavioral Synthesis
- Logic Design
- Verification
- Test
- Physical Design
- Circuit Level Design
- Reconfigurable Systems
- Analog Design
- Embedded software and parallel programming
- Multicore, GPU, FPGA, and heterogeneous systems
- Distributed, networked embedded systems
- Real-time and cyberphysical systems

## Information for Librarians

Foundations and Trends® in Electronic Design Automation, 2025, Volume 14, 4 issues. ISSN paper version 1551-3939. ISSN online version 1551-3947. Also available as a combined paper and online subscription.

# Contents

**Editorial** 1

**Deep Learning and Generative AI for Monolithic and Chiplet SoC Design and Verification: A Survey** 4
*Imed Ben Dhaou, Syhem Larguech, Sree Ranjani Rajendran, Rajat Subhra Chakraborty, Hannu Tenhunen and Ahmed Abdelgawad*

**Large Language Models for EDA: From Assistants to Agents** 54
*Zhuolun He, Yuan Pu, Haoyuan Wu, Yuhan Qin, Tairu Qiu and Bei Yu*

**Evaluating Large Language Models for Automatic Register Transfer Logic Generation for Combinational Circuits via High-Level Synthesis** 74
*Sneha Swaroopa, Rijoy Mukherjee, Anushka Debnath and Rajat Subhra Chakraborty*

**Biographies** 97

# Editorial

The rapid advancements in semiconductor technology have significantly increased integration density, with modern System-on-Chip (SoC) designs for high-performance computing now exceeding 10 billion transistors. However, as we enter the nanometer regime, the traditional CMOS transistor scaling law has reached its physical limits. This has necessitated the adoption of More-than-Moore (MtM) technology, which integrates novel design methodologies and heterogeneous computing architectures.

Electronic Design Automation (EDA), design reuse, and IP-based methodologies have been instrumental in bridging the productivity gap, reducing time-to-market, and meeting increasingly stringent performance and security requirements. As electronic systems grow in complexity, new design and verification methodologies are emerging to address these challenges effectively.

In the pre-Internet of Things (IoT) era, the security of SoC designs was often an afterthought. However, the widespread adoption of IoT devices has made security a critical concern at every level of deployment. Current EDA tools, while optimizing for performance, inadvertently introduce vulnerabilities that expose circuits to threats such as side-channel attacks, reverse engineering, and hardware Trojans. This increasing focus on security necessitates the development of security-aware EDA tools capable of addressing threats such as fault injection, information leakage, and timing and power-based attacks.

Chiplet technology has evolved from tiled silicon architectures, where processors are divided into clusters to enhance signal propagation, to more advanced configurations where multiple processing elements, including RISC-V cores and application-specific co-processors, are integrated into chiplets. These chiplets leverage various interconnection methods, including 3D stacking and wafer-scale integration, utilizing flexible communication networks such as packet-switched and circuit-switched architectures. However, this shift to chiplet-based architectures presents new challenges in performance estimation and system validation due to the complexity of interconnects and heterogeneous integration.

Deep learning algorithms and Large Language Models (LLMs) have emerged as promising solutions to address the growing challenges in SoC and chiplet design and verification. The application of AI-driven methodologies is revolutionizing RTL generation, hardware security, and design automation, leading to enhanced power-performance-area (PPA) metrics and improved verification efficiency.

This special issue features three works that explore the transformative role of AI in EDA and SoC design:

1. **Deep Learning and Generative AI for Monolithic and Chiplet SoC Design and Verification: A Survey** delves into the impact of deep learning and generative AI, particularly LLMs like GPT, on SoC and chiplet architectures. It highlights AI applications in RTL code generation, hardware verification, and security enhancement, including the mitigation of hardware Trojans. The work also discusses AI-driven optimizations in power, performance, and area (PPA) metrics and the integration of AI in commercial and open-source EDA tools.

2. **Large Language Models for EDA: From Assistants to Agents** focuses on the application of LLMs in Electronic Design Automation (EDA), examining their roles as intelligent assistants and autonomous agents. It discusses how LLMs improve productivity by automating question-answering, script generation, and various design processes. It reviews cutting-edge research and practical implementations demonstrating how LLMs streamline EDA workflows, reduce manual effort, and enhance design accuracy.

3. **Evaluating Large Language Models for Automatic Register Transfer Logic Generation for Combinational Circuits via High-Level Synthesis** evaluates the use of LLMs in automating RTL generation in Verilog through a two-stage High-Level Synthesis (HLS) pipeline. The authors propose an approach where LLMs generate annotated C++ code optimized for HLS, which is subsequently converted to Verilog RTL using tools like Vitis HLS. Benchmarking against the VerilogEval dataset, the study demonstrates that this two-step approach significantly improves functional correctness, achieving a pass@1 score of 0.86, outperforming direct LLM-based Verilog generation methods.

These works collectively underscore the transformative potential of AI and LLMs in revolutionizing SoC design and EDA workflows. By enhancing automation, improving security, and PPA optimizing, AI-driven methodologies pave the way for the next generation of high-performance and secure semiconductor designs. This special issue aims to provide valuable insights for researchers and industry professionals, inspiring further advancements in AI-assisted hardware design and verification.

# Deep Learning and Generative AI for Monolithic and Chiplet SoC Design and Verification: A Survey

Imed Ben Dhaou[1,2], Syhem Larguech[3], Sree Ranjani Rajendran[4], Rajat Subhra Chakraborty[5], Hannu Tenhunen[2] and Ahmed Abdelgawad[6]

[1] *Dar Al-Hekma University, Saudi Arabia*
[2] *University of Turku, Finland; imed.bendhaou@utu.fi*
[3] *Cadence Design Systems, USA*
[4] *Florida Atlantic University, USA*
[5] *Indian Institute of Technology Kharagpur, India*
[6] *Central Michigan University, USA*

ABSTRACT

The rapid development of integrated circuit (IC) technology, driven by the growing demand for Internet of Things (IoT) devices, cloud computing, and cyber-physical systems, has introduced significant challenges in the design and verification of modern System-on-Chip (SoC) systems. Contemporary SoCs, whether used in desktops or servers, are extremely complex with billions of transistors and often use mixed-core technologies. Designing complex SoCs in modern technologies faces issues in scalability, security, verification, and design optimization, especially as the industry transitions to chipset-based architectures. In this work, we explore the potential of deep learning and generative Artificial Intelligence

Imed Ben Dhaou, Syhem Larguech, Sree Ranjani Rajendran, Rajat Subhra Chakraborty, Hannu Tenhunen and Ahmed Abdelgawad (2025), "Deep Learning and Generative AI for Monolithic and Chiplet SoC Design and Verification: A Survey", Foundations and Trends® in Electronic Design Automation: Vol. 14, No. 4, pp 245–294. DOI: 10.1561/1000000063-1.

(AI) to address these challenges, focusing on applications in Register Transfer Level (RTL) code generation, design automation, hardware security, and verification.

In this work, we review the state-of-the-art in AI-driven Electronic Design Automation (EDA) tools, examining both open source and commercial platforms that have integrated AI to enhance design efficiency and performance. The work focuses on AI's role in optimizing power, performance, and area (PPA) metrics, as well as improving hardware security by mitigating threats such as hardware Trojans. In addition, we discuss the implications of adopting AI in SoC workflows and its transformative potential in democratizing hardware design.

# 1

# Introduction

The design of integrated circuits (ICs) has made significant strides in recent years, with modern System-on-Chip (SoC) architectures now containing more than six billion transistors and incorporating hundreds of mixed-core technologies. As highlighted in recent studies, this rapid advancement in IC technology is driven by the growing demand for IoT edge devices, personal augmentation systems, cloud computing, and cyber-physical systems. The proliferation of AI across edge, fog, and cloud computing has intensified the need for robust security and scalable hardware solutions.

In the era of giga-scale integration and multicore technology, circuit designers are faced with increasing challenges in thermal management, reliability, cost, and verification. To mitigate operating expenses (OPEX), semiconductor companies are adopting new methodologies emphasising design reuse and hardware intellectual property (IP) based design. However, while these approaches offer significant benefits, they also introduce new challenges in the development of secure SoCs. As reported in Bhunia *et al.* (2014), Bhunia and Tehranipoor (2017), Hussain *et al.* (2023), and Rama *et al.* (2024), the heavy reliance on third-party IPs has jeopardized the security of embedded systems. Hardware Trojan (HT)

can be inserted through malicious IPs, affecting the SoC's functionality, reliability, and privacy.

Motivated by the need to reduce cost and power consumption, monolithic SoC design has been the primary focus in integrated circuits during the last three decades commonly known as Moore's era. In the post-Moore law era, which is characterized by the integration of heterogeneous components and non-silicon technologies, chiplet-based designs are emerging as a more efficient alternative to traditional monolithic SoC (Ravikumar, 2024; Liu *et al.*, 2024). There are two common approaches for designing chiplets: 2.5D heterogeneous integration, which involves assembling multiple chiplets on an interposer, and 3D integration, which consists of stacking several dies vertically. Figure 1.1 illustrates the 2.5D and 3D heterogeneous integration approach.

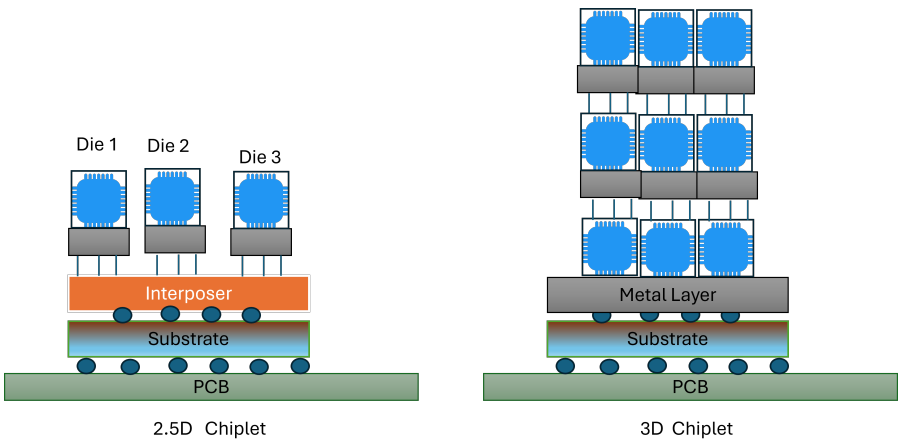


**Figure 1.1:** 2.5D and 3D Chiplets SoC.

The traditional EDA design flow has been typically used to design chiplets independently. However, as suggested in Kabir and Peng (2020), early co-optimization of chiplets significantly improves system performance, signal integrity, and power efficiency compared to traditional methods where chiplets and packages are developed separately. Furthermore, EDA tools must enable secure chiplet design. The proposed secure chiplet design flow, as illustrated in Figure 1.2, involves several key stages. First, the requirements and specifications for the SoC are

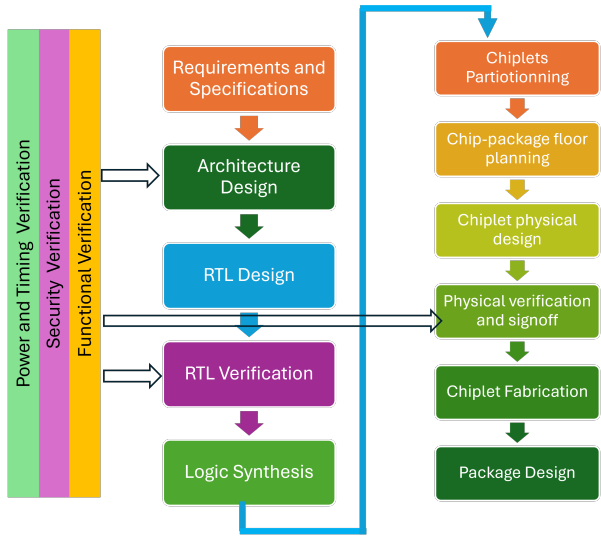


**Figure 1.2:** EDA design flow for designing secure SoC Chiplets.

gathered and processed. Next, these specifications are transformed into an architecture that can be described using a high-level modeling language, such as SystemC. Automated tools then convert this architecture into an RTL (Register Transfer Level) description, typically in VHDL or Verilog, which is verified through simulation. Following this, the logic synthesis phase converts the RTL design into a gate-level netlist. This list is divided into individual chiplets, each of which can be fabricated using the appropriate technology. Finally, the manufactured chiplets are assembled in the last stage of the design flow. Along the design process, after each transformation, the functionality, security, power, and timing of the design should be rigorously verified to ensure performance and security requirements are met.

The machine learning (ML) algorithms most frequently employed in the realm of SoC design and electronic design automation (EDA) tools include several notable techniques. First, reinforcement learning (RL) is particularly effective for applications that demand constant learning and adaptation to the dynamic constraints of design, such as in routing algorithms and design optimization tasks at various design abstraction levels. Next, supervised learning shows efficacy in tasks that

involve identifying patterns, notably in bug detection, Register Transfer Level (RTL) verification, and ensuring hardware security. Additionally, deep learning frameworks, especially Deep Neural Networks (DNN), are capable of capturing and modeling intricate dependencies and complex relationships. These capabilities are especially beneficial in tackling issues like printed circuit board (PCB) routing and the design of chiplets.

This work seeks to explore the potential applications of generative artificial intelligence, alongside advanced deep learning techniques, in both the creation and advancement of SoC architectures, applying both monolithic approaches and chiplet-based configurations. The primary contributions of this work are enumerated as detailed below:

- This study investigates the substantial potential and impacts of deploying large language models (LLMs) within the spheres of RTL code generation, the systems inherent in design automation, and the critical verification processes that are essential to these technologies.
- It provides a comprehensive account of the various design challenges encountered in crafting monolithic SoCs alongside chiplet-based architectures, with particular emphasis on challenges entailing scalability, security, and performance enhancement.
- The work meticulously examines the application of artificial intelligence-driven tools throughout various stages of design, including logic synthesis, complex physical design methodologies, and exhaustive verification procedures.
- It proposes an in-depth discussion regarding the utilization of artificial intelligence to improve strategies that enhance hardware security, focusing on methodologies aimed at reducing susceptibility to threats such as hardware Trojans and side-channel attack mechanisms.
- A detailed overview is included on the successful implementation of deep learning methods in design processes adopting the chiplet-based approach.

Generative AI, encompassing Large Language Models (LLMs) and Deep Learning frameworks, has emerged as a transformative approach to tackle the complexities of modern SoC design and verification. Generative AI systems create new content, such as RTL code or circuit layouts (Mangalagiri *et al.*, 2024), by learning patterns and structures in their training data. LLMs, such as GPT-4, specialize in understanding and generating human-like text, making them invaluable for automating tasks like code and script generation or debugging (Bengesi *et al.*, 2024). Deep Learning, a subset of machine learning, employs multi-layered neural networks to model intricate dependencies, making it particularly effective for tasks such as routing, design optimization, and performance analysis (Dong *et al.*, 2021).

This survey focuses on Large Language Models (LLMs), Deep Neural Networks (DNNs), and RL, as these are particularly well-suited for the complexities of SoC design and verification. Classic machine learning approaches, such as decision trees or clustering, are not included due to their limited applicability in addressing the high scalability and intricate dependencies inherent to SoC workflows

The rest of the work is organized as follows. Section 2 reviews existing survey works and puts our work in context. Section 3 summarizes the research work related to the design and verification of SoC systems. Section 4 describes the algorithms, tools, and techniques for large language models and generative AI. Subsequently, Section 5 discusses the application of generative AI in monolithic SoC design. Section 6 delves into the application of deep learning in chiplet design. Section 7 surveys the adoption of LLM and deep learning by EDA toolchains. Section 8 discusses the open issues in the effective use of LLM for SoC design. Finally, Section 9 concludes the work.

# 2

# Related Work

The inherent statistical variability of parameters in integrated circuits, such as delay, power consumption, and transistor count, has driven the adoption of machine learning, artificial intelligence, and data mining techniques for optimising, designing, and synthesizing analog, Radio Frequency (RF), and digital integrated circuits. Numerous survey papers have extensively explored this growing intersection of AI and IC design, highlighting its potential to address the challenges associated with designing and optimizing SoC systems.

The integration of artificial intelligence into integrated circuit design began four decades ago. Early examples of computer-aided design (CAD) and computer-aided manufacturing (CAM) tools include XCON, an expert system for configuring computer systems (Kirk, 1985). CMU-DA (Carnegie Mellon University Design Automation) generated technology-independent circuit components from data flow graphs (Kowalski and Thomas, 1983), while TALIB was a knowledge-based system for automatic NMOS cell layout (Kim and McDermott, 1983). EMUCS, an extension of CMU-DA, synthesized data paths from behavioral descriptions (Hitchcock and Thomas, 1983).

The complexity of designing very-large-scale integration (VLSI) circuits, classified as an NP-complete problem, soon necessitated heuristic

algorithms and design abstraction levels to address challenges. As detailed in Breuer *et al.* (2000), heuristic algorithms for circuit partitioning during physical synthesis include the Kernighan–Lin algorithm, Fiduccia–Mattheyses algorithm, hMetis (a multilevel partitioning algorithm), and ratio-cut partitioning. For floorplanning and placement, key heuristic techniques discussed are simulated annealing, genetic algorithms, and the quadratic placement algorithm. In the domain of Automatic Test Pattern Generation (ATPG) for single stuck-at faults (SSFs), heuristics such as the D-algorithm, Path-Oriented Decision Making (PODEM), and fan-out-oriented test generation algorithms are highlighted. Finally, fault simulation methodologies explored include concurrent fault simulation, parallel fault simulation, deductive fault simulation, and critical path tracing.

Heuristic algorithms in CAD often produce suboptimal results, as they balance performance against computational complexity. To address these limitations, machine learning (ML) has emerged as a powerful alternative. As highlighted in the survey by Rapp *et al.* (2022), ML surpasses heuristics by learning directly from data, enabling it to capture complex patterns, interactions, and dependencies inherent in modern designs. Moreover, ML leverages advanced techniques such as RL and gradient-based optimization to explore vast design spaces effectively, achieving globally optimal or near-optimal solutions. Machine learning algorithms effectively address a wide range of design challenges in the semiconductor industry, including lithography, physical design, manufacturing, yield, reliability, failure modelling, analog design, and system optimization (Elfadel *et al.*, 2019). Specifically, machine learning algorithms have been proven to be effective in performance prediction, design space exploration, black-box design, decision-making design automation (Huang *et al.*, 2021).

Estimating power at a high level is a crucial aspect of designing complex SoCs. For devices running on batteries, minimizing power consumption has become a vital design goal, aiming to extend the device's operational duration. The power used by the ASIC/SoC is influenced by factors such as operating frequency, switching activities, supply voltage, and switching capacitance. A variety of methods have been put forward to decrease power consumption at different stages like

the system, algorithm, architecture, logic, circuit, and layout levels (Ben Dhaou, 2002; Rabaey, 2009; Reda and Nowroz, 2012). In the realm of low-power design, statistical, probabilistic, and simulation-based power estimation methods have been predominant. These techniques have been reviewed in Nasser *et al.* (2021). In the statistical approach, linear regression which is a supervised machine learning that has been widely used for developing high-level power estimation tools (Ben Dhaou and Tenhunen, 2002; Schuermans and Leupers, 2019).

In traditional machine learning, there is often a significant reliance on the extraction of features manually, along with various pre-processing steps. The scope and depth of these actions can differ substantially depending on the specific algorithm being employed and the characteristics of the dataset. However, with the remarkable progress made in the realm of cloud computing and the analytics of big data, there has been a notable rise in the application of deep learning (DL) as a particularly robust branch of machine learning. Deep learning fundamentally relies on multiple layers of artificial neural networks (ANNs) to facilitate the automatic learning of hierarchical features directly from raw data inputs. These sophisticated architectures are commonly known as Deep Neural Networks (DNNs), and they encompass specialized variants such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs). These variants are particularly well-suited for complex tasks, including but not limited to image recognition applications and sequence modeling endeavors (Dong *et al.*, 2021).

In CAD (Computer-Aided Design), deep learning is a game-changing tool with uses in synthesis, performance prediction, design exploration, run-time management, device and technology evolution, physical design, lithography, and manufacturing. For instance, convolutional neural networks (CNNs) have been employed to predict the performance of integrated circuits during high-level synthesis, enabling designers to explore vast design spaces efficiently. Similarly, generative adversarial networks (GANs) have shown promise in optimizing lithography processes by generating mask patterns that minimize defects and improve yield (Rapp *et al.*, 2022)

Many of the studies mentioned above have neglected the importance of designing secure integrated circuits, which are the foundation of trust

in any computing-based solution. As noted in Koblah *et al.* (2023), major cybersecurity attacks such as *Meltdown* and *Spectre* exploited the vulnerabilities at the processor level. In cases where it is challenging to analytically model security properties or when multiple security conditions must be satisfied, machine learning techniques are invaluable for designing secure Computer Aided Desing (CAD) tools.

In recent years, the application of large language models (LLMs) to SoC design has gained traction. Studies such as Saha *et al.* (2024) explored their potential in hardware security, demonstrating capabilities in vulnerability insertion, security assessment, and countermeasure development. These advancements underscore the transformative role of AI in both optimizing performance and ensuring the security of modern IC designs.

# 3

# Design and Verification of SoC Systems

In the modern era characterized by advanced technology, there is a notable predominance of chip-centric systems. Such systems comprise devices that seamlessly integrate hardware and software components that have been meticulously pre-engineered. Collectively, these components are known in the industry as design intellectual properties, commonly abbreviated as IPs. In the present landscape, the majority of companies either independently develop these IPs through their in-house teams or opt to outsource this development to specialized external firms. When it comes to incorporating these IPs into a fully functional device, a dedicated team responsible for SoC integration takes on the crucial role. This team is charged with the careful collection and meticulous integration of the IPs, ensuring they align with the unique system requirements of the desired device. It is of paramount importance that these IPs communicate effectively amongst themselves, utilizing standardized interfaces to guarantee their successful integration into the system's configuration. For instance, the ARM AMBA bus interface provides an on-chip interconnect specification that facilitates the connection and management of various functional blocks within the system. Within the domain of SoC design, there exist two distinct verification processes:

one dedicated to validating the proper functioning of the individual IPs and ensuring compliance with interface protocols, and another focused on verifying the assembled system as a whole. In contemporary times, the field of hardware verification has reached a relatively advanced stage of maturity, having been the subject of extensive study in both academic research and industrial application for at least three decades, as evidenced by numerous publications in the discipline (Bhadra *et al.*, 2007; Gupta, 1992).

In contemporary industrial practices, verification has become a key, standardized phase integrated within the systems development lifecycle. Despite the advancements achieved in technology, there remains a noteworthy disparity between the existing capabilities of state-of-the-art verification techniques and the demands imposed by contemporary industrial design practices. The dynamic evolution of the design landscape further intensifies these challenges as we transition swiftly and perhaps inevitably into an era characterized by autonomous vehicles, the proliferation of smart cities, and the expansive growth of the Internet of Things (IoT). This profound shift has heralded the advent of an environment wherein electronic devices ubiquitously gather, analyze, and securely store vast amounts of our most personal and private information, such as geographic location, health data, fitness metrics, and sleep cycles. After this information is transmitted across a global network comprised of billions of interconnected computers, the system remarkably sustains uninterrupted operation, even in the presence of millions of potentially malicious or compromised nodes within that network. Therefore, it is crucial that verification processes evolve in tandem with the design and architectural frameworks of these systems to effectively integrate within this novel ecosystem (Ray *et al.*, 2016; Farahmandi *et al.*, 2023a; Nath *et al.*, 2018; Rajendran, 2020).

The availability of resources is a highly significant factor influencing the verification process in contemporary times. The imperative to manufacture vast quantities of varied computing devices has intensified the constraints on time to market for both design and system development. Historically, the typical life cycle of a microprocessor—from the exploratory phase to the commencement of production—spanned approximately three to four years. Conversely, the life cycle duration

for certain Internet of Things (IoT) devices has now been truncated to less than one year (Chen *et al.*, 2017).

Due to this accelerated contraction, there is inadequate time available for comprehensive design evaluations, which can lead to misunderstandings between developers and stakeholders concerning the design's functional decomposition. This, in turn, elevates the risk of errors. Additionally, a condensed life cycle allows for reduced testing duration. Verification teams are consequently tasked with managing designs that may be more prone to errors using limited resources and in a constrained timeframe. Such aggressive scheduling has resulted in an increase in in-field escapes and a heightened demand for post-deployment patches on devices and systems, often necessitated through updates to software and firmware. Moreover, the creation of verification methodologies targeting feasible and highly valuable objectives, including security, networking, and cyber-physical elements, has become imperative.

The increasing complexity of contemporary computing devices necessitates meticulous planning from the outset, which extends through nearly the entire design lifecycle. The process of verification planning is initiated simultaneously with the commencement of product planning and persists throughout the entirety of the system development phase (Chen *et al.*, 2017; Rajendran, 2020). Within the framework of product planning, it becomes essential to delineate the diverse Intellectual Properties (IPs) needed, as well as their breakdown into both hardware and software constituents. This process involves specifying the interfaces required for interconnection and seamless communication. Additionally, it is crucial to establish and understand the variety of objectives concerning power consumption, system performance, security, and energy efficiency. As a result, the phase of verification planning encompasses the development of detailed test plans and corresponding test cards, alongside the generation of *Verification IPs* (VIPs). These Verification IPs represent specialized blocks within the design that play a critical role in debugging activities following the silicon fabrication. Furthermore, the design incorporates instrumentation that is integral to the processes of monitoring, checking, and actively exercising the system.

System architecture is a critical aspect of defining an SoC design, as it establishes the functional parameters, communication protocols

among IPs, and the schemes for managing power and performance. This phase involves an in-depth examination of numerous parameters and design characteristics, such as cache size, pipeline depth, and protocol specifications for security and power management. During this exploration, various "architectural models" are employed to simulate typical workloads and the intended use cases of the device, thus helping to determine parameters that meet the objectives set in the planning phase, which often includes considerations like power, performance, and security. At this juncture in the architectural exploration process, two primary verification activities are paramount (Rashinkar *et al.*, 2007; Lee and Kim, 2011; El Fentis, 2020). The initial activity involves ensuring that all communication protocols function as anticipated. By engaging in this process early, when the design models are abstract and the overall design is still nascent, high-level protocol errors can be detected early, leading to reduced costs; if such errors are discovered later in the product implementation, resolving them might necessitate a significant redesign of multiple IPs.

Due to the abstract nature of design models at this phase, formal analysis is employed to meet this goal. In conjunction with formal analysis, high-level simulation is utilized in practice to obtain the necessary coverage (Talupur *et al.*, 2015). Besides its indispensable contribution to software and firmware verification, verification is instrumental in commencing the creation of hardware prototyping models. This requirement arises because the low-level software and firmware programs must be validated to function correctly alongside the target (and evolving) hardware design developed during the implementation phase. Nevertheless, the verification of software and firmware must proceed without waiting for the hardware implementation to stabilize. Consequently, high-level hardware-software models are generated to expedite the verification process for software and firmware.

The process of formal verification of SoC connectivity typically involves representing specifications in spreadsheets or XML documents. Once these specifications are loaded, the dedicated software tool autonomously generates the properties necessary to carry out formal verification (Liao and Hsiung, 2003). This approach, however, brings about the challenge of establishing a standardized specification framework

capable of encapsulating the myriad of available connection schemes. Furthermore, the intricacy involved in administering the verification process for an extensive number of properties—running into tens of thousands—can be daunting, often necessitating the parallel operation of formal verification engines across a distributed computing environment. In addition to formalizing the connectivity verification as per an existing specification, there exists an additional utility that is adept at extracting and retrieving high-level connectivity specifications from a given design, in instances where no formal specification is provided (Farzana *et al.*, 2019). The design specifications can subsequently be reviewed based on these extracted specifications. Moreover, it is a typical scenario wherein SoC designs undergo modifications aimed at enhancing performance or features, with the connectivity remaining unaffected. Hence, the extracted specifications serve as a valuable resource for formally verifying whether such design modifications have maintained connectivity integrity.

Within an SoC architecture, IP blocks comprise numerous memory-mapped registers that can be accessed by system or user-level software to configure the IP and monitor its operational state. Relative to other system components, these memory-mapped registers exhibit a lower complexity in terms of implementation. However, due to the vast number of registers, each governed by distinct access protocols, verifying them manually using directed or slightly random tests is a labor-intensive and error-prone process, as stated by Kim *et al.* (2013). This challenge is further exacerbated by sophisticated features like register aliasing and remapping, which introduce additional layers of complexity. Verifying memory-mapped registers is also hindered by the lack of precise and dependable specifications; specification documents occasionally overlook critical details, such as scenarios where a memory-mapped register might be internally altered (Kim *et al.*, 2013; Roy, 2007; Liao and Hsiung, 2003). In circumstances where behavior is undocumented, verification engineers are often compelled to resort to trial and error methods to ascertain this behavior.

Due to the increasing complexity and integration of SOC, its development time increases dramatically, and its verification time even more so. As part of the verification process, many directed or randomized

simulation tests are generated and run as regression tests. As far as full chip verification goes, this regression could represent a critical path in a project's schedule if it takes a considerable amount of time to complete. A machine learning (ML) model was useful in automating many parts of the process, which occupied engineers' time and distracted them from adding new coverage metrics. A variety of machine learning models are currently being deployed in areas such as stimulus constraining, test generation, coverage collection, bug detection, and localization (Vaithianathan *et al.*, 2024). By combining AI and Machine Learning with Universal Verification Methodology (UVM), semiconductor design has been thought to become more efficient and accurate. The most effective applications of AI and ML are those concerned with automating time-tapping processes that were previously subject to human error. Using these technologies, it is possible to sift through a huge volume of data and identify what the algorithm believes is likely to be weaknesses in a design not easily noticeable by humans. In addition to supporting the verification task, it also improves the quality of the verification result.

Detecting bugs in software at the Register Transfer Level (RTL) level through a novel AI-driven approach is proposed in Wang *et al.* (2024a). By combining advanced machine learning techniques with domain-specific knowledge of chip design, we address the challenges of increasing complexity and deadline pressures in modern integrated circuits. A comprehensive preprocessing pipeline captures the syntactic and semantic features of RTL code and feeds it into a novel attention-based neural network model. There are still a number of major challenges facing formal verification despite impressive advances. Modeling the environment, insufficient specifications, and complex decision problems are among the most significant. Inductive inference (learning from examples) using hypotheses about system structure is a viable method of tackling these challenges (Seshia, 2012). Researchers (Farahmandi *et al.*, 2023b) examined the benefits of machine learning, inferred new relationships between side-channel measurements, and extracted sensitive data in the survey. The survey also focuses on techniques used to detect and prevent malicious attacks in machine learning systems. Furthermore, it seeks to identify challenges associated with using machine learning for security purposes. In Hughes *et al.* (2019), two DUTs were

tested using both reinforcement ML algorithms and supervised learning algorithms. Constrained-random DV environment tools are leveraged by using supervised learning and reinforcement learning techniques to enhance them. In this way, Design Verification (DV) objectives of full design coverage are achieved on an accelerated timescale and with fewer resources than randomly generated results. Here are two hardware verification examples based on open-source RISCV-Ariane design and Google's RISCV Random Instruction Generator: one using a Cache Controller design and the other using an open-source RISCV-Ariane design.

Employing formal methods to verify records offers the possibility of automating the traditionally manual procedure of analyzing access policies. This is achieved through the formulation of formal properties concerning these policies and subsequently verifying their accuracy and integrity using formal methodologies. The principal challenge encountered in this context is the creation of thorough and robust assumptions about access policies, especially in relation to their application to advanced functional aspects (Grimm *et al.*, 2018).

While classic ML algorithms, such as k-means clustering and support vector machines (SVMs), have been explored in hardware design, they often fall short in tasks requiring scalability and adaptive learning, such as routing and design optimization. This survey emphasizes modern algorithms like LLMs, RL, and DNNs, which better address these challenges.

# 4

# Overview of Generative AI and Large Language Model

Natural Language Processing (NLP) is an important field in computer science and artificial intelligence which lies at the intersection of computational linguistics, statistical engineering, and human language. It has evolved from simple tasks, such as translation in the early 1960s, to solving more complex problems, driven by Large Language Models (LLMs).

LLM represents a category within generative artificial intelligence, often abbreviated as GenAI. While LLMs primarily focus on producing human-like text, GenAI encompasses a broader spectrum of capabilities. Beyond text generation, it can create a variety of other media formats. These include the production of visual media such as videos and images, as well as auditory outputs like speech (Bengesi *et al.*, 2024).

Typical NLP applications include virtual assistant text summarization, sentiment analysis, information retrieval, and speech recognition. As shown in Figure 4.1 a typical NLP chain consists of four main stages. In the first stage, the text or corpus is pre-processed. In the second stage, the features of the document are extracted. In the third stage, the language of the document is modeled. Finally, the last stage performs the intended task on the document.

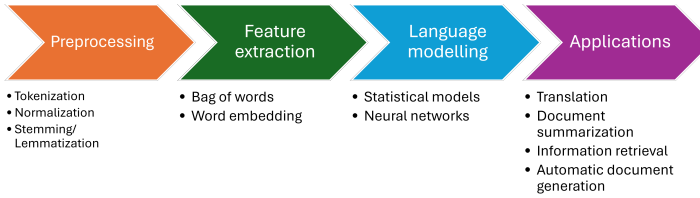


**Figure 4.1:** NLP processing blocks.

Language modeling (LM) plays an instrumental role in the design of effective NLP tasks. Earlier techniques such as *N-gram models* have been based on probabilistic theory, in which it predicts the next word, $w_n$, given the history $(w_1, w_2, \ldots, w_{n-1})$ using conditional probability $P(w_n \mid w_1, w_2, \ldots, w_{n-1})$ (Chang *et al.*, 2024). The N-gram model is computationally efficient and requires processing a corpus to compute the probabilities. However, its precision is limited by the number of words $n$ used to compute the probability, making it unable to capture long-term dependencies in the text. Furthermore, the N-gram model suffers from both sparsity and storage problems.

Neural networks (NN) and their variants, such as Recurrent Neural Networks (RNN) and Long Short-Term Memory (LSTM) networks, have replaced the N-gram model, ushering in a new era of language modeling. As illustrated in Figure 4.2, words or tokens are first converted into word embeddings using algorithms that produce vectors in a high-dimensional space, capturing the semantic relationships between words. These vectors are then fed into the input layer of the neural network. The hidden layers process the embeddings, and the output layer generates the final predictions.

Neural network (NN)-based architectures have significantly transformed language modeling, laying the foundation for advanced techniques in natural language processing (NLP), particularly in tasks such as neural machine translation (NMT) and named entity recognition (NER). The introduction of NNs, particularly recurrent neural networks (RNNs), enabled breakthroughs in sequence processing, while the development of word embeddings provided a method to represent words as continuous vectors in semantic space, capturing relationships

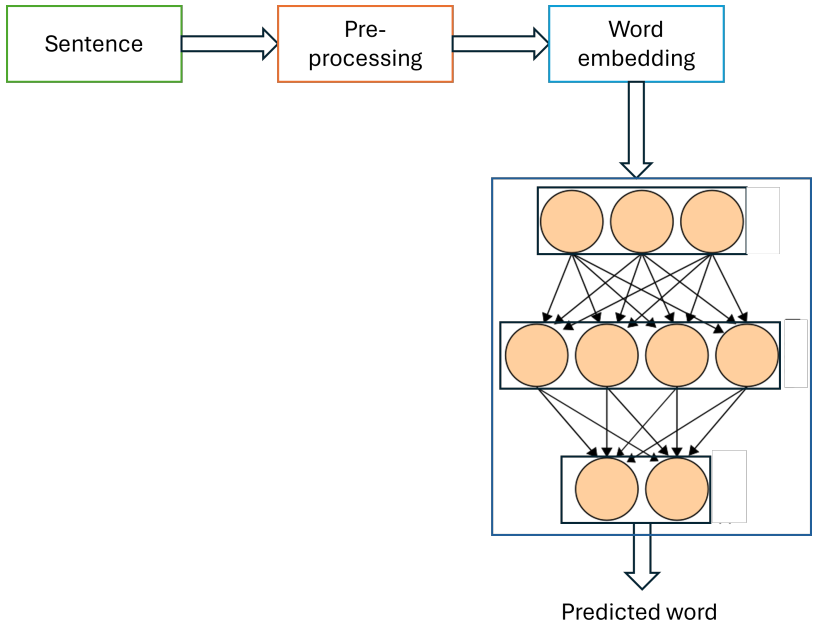


**Figure 4.2:** Architecture of a neural network-based language model.

between words through vector similarity. However, a key limitation of early NN-based translation models, such as those using RNNs, is their reliance on fixed-length vector representations to encode input sequences. This constraint can lead to information loss in long sequences and impose computational inefficiencies, particularly when dealing with long or complex sentences.

The *attention* mechanism, proposed as a solution to this limitation, allows models to selectively focus on relevant parts of the input sequence while generating each output token (Bahdanau *et al.*, 2015). This solution addressed the bottleneck of fixed-length vectors and improved both accuracy and computational efficiency. The chief idea is the dynamic assignment of different weights to different input tokens based on their relevance to the current prediction. Despite these improvements, attention-based models that still rely on RNNs face challenges, notably high memory consumption and slower training times. To address these issues, the *Transformer* architecture was introduced in 2017, replacing

RNNs entirely with attention mechanisms. The Transformer, composed of an encoder-decoder architecture, uses multi-headed self-attention and feed-forward neural networks to process sequences in parallel, drastically improving both sed and scalability. The encoder consists of layers of multi-headed self-attention and feed-forward networks, while the decoder incorporates masked multi-headed self-attention to handle autoregressive generation. This technique, known as the Transformer-based architecture, created another inflection point in NLP (Vaswani *et al.*, 2017).

As summarized by Zhao *et al.* (2023), scaling laws relate the performance and efficiency of LLMs to the model size ($N_M$), the number of tokens ($N_D$), and the computation cost during training ($N_C$). The two most widely used formulas for estimating the performance of LLMs are the KM scaling law and the Chinchilla scaling law, which are capable of estimating the performance of LLMs given $N_M$, $N_DS$, and $N_C$. In their original article, Kaplan *et al.* (2020) found that the scaling law is weakly dependent on the network depth and size. Using the cross-entroy loss ($L$), the expression for the KM scaling law is given in (4.1–4.3). Those equations are valid for model size in the range of 768 million to 1.5 billion non-embedding parameters and the data size in the interval 22 million to 23 billion tokens.

$$L(N_M) = \left(\frac{N_c}{N_M}\right)^{\alpha}_{NM} \tag{4.1}$$

$$L(N_D) = \left(\frac{D_c}{N_D}\right)^{\alpha}_{ND} \tag{4.2}$$

$$L(N_C) = \left(\frac{C_c}{N_C}\right)^{\alpha}_{NC} \tag{4.3}$$

The fitting values for $N_c, \alpha_{NM}, D_c, \alpha_{ND}, C_c$ and $\alpha_{NC}$ are summarized in Table 4.1.

The emerging capabilities of LLMs have been shown to be predicted using these scaling laws, including in-context learning, instruction follow-up, and step-by-step reasoning. The general architecture of LLMs, depicted in Figure 4.3, illustrates the essential operations involved in LLM (Naveed *et al.*, 2023). The architecture includes blocks dedicated to in-context training, which allows the model to adapt and respond

**Table 4.1:** Summary of fitting values for various parameters.

| Parameter | Fitting Value |
|---|---|
| $N_c$ | 88 trillions non-embedding parameters |
| $\alpha_{NM}$ | 0.076 |
| $D_c$ | 54 trillions tokens |
| $\alpha_{ND}$ | 0.095 |
| $C_c$ | $3.110^8$ PF-day |
| $\alpha_{NC}$ | 0.05 |

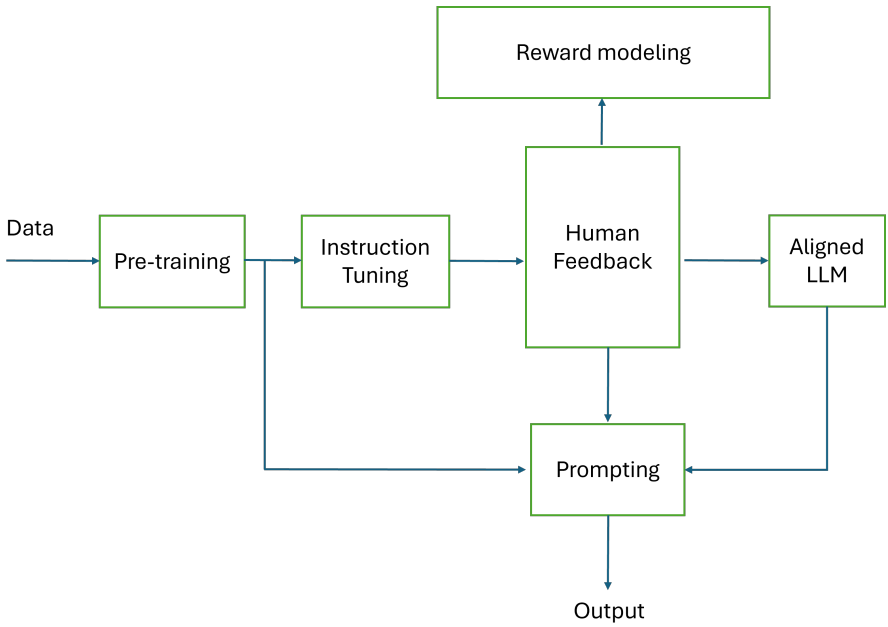


**Figure 4.3:** Generic architecture of LLM with prompting and reward.

based on the context provided within the input data. Furthermore, the architecture incorporates mechanisms for improvement through *Reinforcement Learning with Human Feedback* (RLHF), where the model's performance is iteratively improved based on feedback from humans. The prompt block guides the operation of the LLM by providing specific instructions or contexts that the model uses to generate relevant and coherent responses.

# 5

# Applications of Generative AI in SoC Design

Prior to the development of large language models (LLMs), neural network-based natural language processing (NLP) techniques were widely adopted in tasks such as automatic software development, software testing, and bug fixing for both embedded systems and general-purpose computing (Hai *et al.*, 2022). As highlighted in a survey by Dehaerne *et al.* (2022), popular applications of machine learning in automatic software development include code generation from natural language specifications, software documentation, automatic program repair, and programming by example. The survey also emphasized that transformer-based models significantly outperform earlier architectures like recurrent neural networks (RNNs) and convolutional neural networks (CNNs) for tasks involving code generation from natural language specifications. This is largely due to the transformers' ability better to capture long-range dependencies and contextual relationships in code, making them more effective for complex programming tasks.

Earlier approaches for automatic software development had limited capabilities in handling complex and context-rich tasks. In contrast, large language models have demonstrated impressive performance in software engineering. For example, Codex, a fine-tuned generative pre-

trained transformer developed by OpenAI, can generate standalone Python functions from docstrings which is a string used to document a Python function, class, module, or method (Chen *et al.*, 2021). Codex is used in GitHub Copilot (OpenAI, 2024).

Stemmed by the success in code generation, recently, LLMs' capabilities have been extended to cover SoC design. As summarized in Wang *et al.* (2024b), LLMs have been used in EDA and hardware security. In the EDA realm, LLMs are used, for instance, for RTL code generation, design optimization, bug fixes, script generation for HLS tasks, HDL verification, and analyses (Wu *et al.*, 2024a). For hardware security, the applications of LLMs include fixing hardware security bugs, inserting Hardware Trojan (HT), defending against side-channel attacks, and generating security assertions.

With the objective of democratizing hardware design, the OpenROAD project, an open-source EDA toolchain, has been initiated. The tool is capable of performing logic synthesis and physical design tasks, including floor planning, clock tree synthesis, and global and detailed routing. The OpenRoad design flow is described in Figure 5.1. It combines both logic and physical synthesis. The tool generates GDSII file given the RTL description of the circuit using Verilog hardware description language (Ajayi *et al.*, 2019).

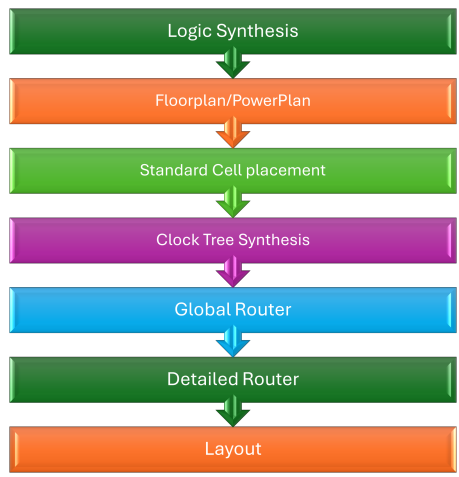


**Figure 5.1:** openROAD design flow.

To further enhance the capabilities of the OpenROAD toolchain, an open-source curated dataset for training large language models has been created. The dataset is designed to generate scripts and answer questions through a tailored chatbot. The question-answer dataset has been collected from the OpenROAD GitHub and is categorized into three groups: general questions pertaining to the use of OpenROAD, specific questions on OpenROAD tools, and questions on the design flow of OpenROAD from RTL to GDS. The dataset has been used to develop an open-source LLM-powered chatbot using the Llama3-8B foundation model. The training process is pictured in Figure 5.2.

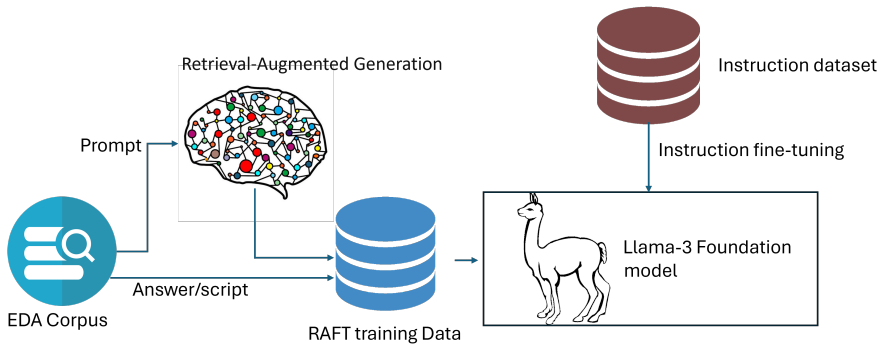


**Figure 5.2:** Training process for the OpenRoad-assistant reported in Sharma *et al.* (2024).

The process is initiated by sending queries to the retriever model, which searches through a domain-specific database. The context retrieved is merged with the supervised training data to form a RAFT (Retrieval Augmented Fine-Tuning) dataset. The Llama3 foundation model is then trained using the new RAFT dataset. Compared with ChatGPT-3.5, ChatGPT-4, Code Llama, and Claude3, the OpenROAD-Assistant scored high in script generation (77% pass@1 and 80% pass@3) and in question-answering tasks (98% BERTScore and 96% BARTScore).

The work presented in Chen *et al.* (2024) explores the shift from traditional AI-enhanced Electronic Design Automation (AI4EDA) towards AI-native EDA systems, driven by the development of Large Circuit Models (LCMs). The authors argue that existing AI4EDA solutions, which repurpose models from domains like vision and language process-

ing, are insufficient for the unique complexities of circuit design. Instead, they propose LCMs—multimodal models tailored to EDA workflows that can unify diverse design stages such as specification, RTL, netlists, and physical layouts.

Motivated by the need to have a specialized LLM for automating the design of SoC, the work by Wu *et al.* (2024b) elaborates on an autonomous agent named ChatEDA, which uses a fine-tuned open-source LLM model, Llama2. The agent, AutoMage, is capable of autonomously decomposing tasks, generating scripts, and executing tasks. The work details the integration of EDA tools to streamline the design flow from RTL to GDSII. The capabilities of AutoMage have been compared with existing foundation models such as GPT-3.5, Claude2, and GPT-4, demonstrating superior performance in handling EDA tasks. Additionally, an upgraded version, AutoMage2, incorporates enriched training data, instruction tuning with explanations, and chain of thoughts prompting to further enhance its capabilities.

# 6

# AI Techniques for Chiplet Design and Heterogeneous Integration Packaging

The semiconductor sector is transitioning significantly from traditional monolithic chips to the arena of three-dimensional integrated circuits (3D-IC), chiplets, as well as vertically stacked silicon and wafers. Presently, advanced SoCs are encountering the constraints imposed by reticle size limits. Many enterprises have come to the consensus that solely adhering to Moore's Law, often referred to as "More Moore," no longer represents the most viable path forward in terms of both technical and economic aspects, especially for the upcoming generation of designs. As the industry approaches the finite scaling limits inherent to advanced nodes, the pressure for enhanced computational performance along with efficient data transfer has reached unprecedented levels. This scenario has necessitated the exploration and development of pioneering solutions, aiming to maintain the trajectory of Moore's Law scaling while achieving notable improvements in performance, coupled with a reduction in power consumption.

The semiconductor packaging sector is gearing up to assume a more prominent and essential role in the design of future electronic products. The integration of chips through vertical stacking within a single package (known as 3D integration) and the employment of a

multi-chiplet structure utilizing a silicon interposer within the same package (referred to as 2.5D integration) are becoming favored solutions, each presenting distinct challenges. As the demand for heterogeneous chiplet-based architectures increases, there is a necessity to develop innovative system-level design methodologies that focus on optimizing Power, Performance, and Area (PPA) metrics at the system level. The Cadence Integrity 3D-IC platform stands as the industry's pioneering comprehensive solution for system planning, execution, and precise preliminary analysis. This platform makes use of Cadence's leading-edge implementation and validation technologies across digital, analog, and packaging domains, all coordinated within a cohesive hierarchical database.

In the context of heterogeneous integration, an IC package now includes various elements such as bare dies/chiplets, packaged devices, interposers, bridges, and passive components. Numerous complex IPs must be routed within the package, including die-to-die connections and links between dies and SMD components to the board, such as DDR, PCIe, and UCIe. A range of strategies must be explored to find the optimal routing solution, which can be time-consuming. The chosen strategy will directly affect the routing quality of both the IC package and the board, as well as the overall system cost. AI approaches have demonstrated their ability to outperform humans in problems requiring multiple strategies to win.

In the design flow, many advanced routing algorithms are adopted to automatically route all nets. Physical design routing was traditionally solved using heuristics and optimization. Below is an example of some work related to physical design routing.

The algorithms are classified into escape routing and area routing. Escape routing is more complex to achieve compared to area routing because of the high routing density. It is an important problem in package and PCB design. Several works were conducted to develop escape routing solutions mainly for PCB designs. Network flow (Yan and Wong, 2012) and nonflow (Gandhi *et al.*, 2019) are pervasively used to model this problem. The proposed escape routing models are classified into three categories; unordered escape (Yan and Wong, 2012) (Chen *et al.*, 2023), ordered escape (Lin *et al.*, 2021) and simultaneous escape (Ozdal and Wong, 2004).

For instance, the DDR macro is considered the most complex IP to escape route in IC package design. It often determines the required total number of layers of the substrate. For this macro type, net grouping is needed, requiring an ordered escape routing approach with significant consideration given to power and ground connections. Each of the above methods has limitations, such as not considering different net types (power, ground, and Differential Pairs (DP)) or net ordering. Crosstalk is a major issue for DDR macro routing. To achieve the IP's requirements in terms of crosstalk values, adjusting the line spacing outside the escape region is often a key element. The line spacing outside the escape routing region also influences the escape routing strategy. The more severe the crosstalk requirements, the larger the spacing between the lines both outside and inside the macro.

In the study outlined in Yu and Wei-Ming Dai (1995), the authors present a pioneering routing framework known as the *Alpha-PD-Router*, which is underpinned by a RL methodology that is independent of specific data architectures. This model is adept at learning to route circuits while simultaneously resolving short violations. The underpinning RL strategy, named *Alpha-Go Zero*, distinguishes itself by negating the necessity for extensive training datasets, which are notoriously challenging and costly to compile. Rather than relying on such datasets, this approach empowers an RL agent to acquire routing efficiencies and improve routing strategies through dynamic interactions within the design environment. The designed Alpha-PD-Router leverages a sophisticated cooperative min-max game framework synergistically integrated with physical design routing algorithms. This novel framework derives its conceptual foundation from the principles of Alpha-Go Zero, an innovation by Google that has demonstrated the capability to master the intricate game of Go autonomously, without human guidance.

The work presented by McMurchie and Ebeling (1995) introduces a comprehensive routing algorithm designed to address a variety of complex, real-world constraints essential for effective printed circuit board routing. For this routing task (including Signal, power, and ground distribution, and DP), the Pathfinder algorithm (Fang *et al.*, 2009), known for its negotiation-based approach, is utilized. This algorithm aims at minimizing violations of design rules while achieving comprehensive

routing solutions. Notably, the pathfinder is capable of executing both escape routing and area routing operations. Originally devised for routing circuits on FPGAs, Pathfinder functions as an iterative algorithm adept at managing and reconciling the conflicting objectives of reducing congestion and minimizing the delay across critical paths within an iterative process. This process begins by permitting signals to utilize shared routing resources but transitions into a phase where signals must negotiate to ascertain which requires the shared resources the most. The authors further suggest an innovative cost function specifically designed for managing obstacles encountered by routed nets during each iteration of the routing process.

The core innovation of Fang *et al.* (2007) involves the use of non-local criss-cross attention networks, a type of neural network architecture that captures long-range dependencies and spatial relationships more effectively than traditional methods. This allows the model to consider thermal effects across the entire PCB layout, rather than just local areas. The criss-cross attention module enables the network to focus on critical areas that influence thermal distribution, allowing for more informed routing decisions. The model incorporates thermal metrics directly into the routing optimization process, using data on thermal conductivity, power dissipation, and other relevant factors. The authors use a comprehensive dataset of PCB designs, including thermal simulations and routing solutions, to train and evaluate the T-Router model. Performance is assessed using metrics such as thermal uniformity, routing efficiency, and computational time. The thermal-driven routing approach shows high efficiency in achieving a low-temperature PCB design compared to the classic verification-then-fix approach.

# 7

# Adoption of Generative AI and Deep-learning by Open Source and Commercial EDA Tools

Leading Electronic Design Automation (EDA) companies have been expanding their capabilities by integrating AI and generative AI technologies into their toolsets. The following section explores this significant surge and its impact on the evolution of EDA solutions.

## 7.1 Cadence Design Systems

The Cadence Joint Enterprise Data and AI (JedAI) platform is designed to accelerate intelligent design processes from chips to complete systems. It features a variety of generative AI applications, including Allegro X AI, Optimality Intelligent System Explorer, Verisium AI-Driven Verification, and Cadence Cerebrus Intelligent Chip Explorer (Cadence Design Systems, 2025).

Allegro X AI utilizes cloud scalability to significantly reduce PCB design cycle times. By automating component placement, power plane creation, and critical net routing, it boosts engineers' productivity. This allows for more iteration and exploration without sacrificing quality, ensuring electrical correctness and manufacturability.

The Cadence Optimality Intelligent System Explorer facilitates the analysis and optimization of electronic systems, transcending traditional,

labour-intensive optimization processes. It replaces the conventional design-test-refine loop with generative AI technology, quickly delivering optimal system design solutions without compromising accuracy. Additionally, Cadence Cerebrus Intelligent Chip Explorer offers an AI-driven, automated approach to chip design flow optimization. Block engineers define design goals, and the generative AI features intelligently optimize the design to meet power, performance, and area (PPA) objectives in a fully automated manner.

Moreover, the Cadence Verisium AI-Driven Platform represents a significant leap forward by harnessing big data and generative AI across multiple runs of multiple engines in a comprehensive SoC verification campaign. Verisium optimizes verification workloads, enhances coverage, and accelerates root-cause analysis of bugs. Cadence Virtuoso Studio, a key component of the Cadence.AI Generative AI Platform, builds on 30 years of industry expertise in custom/analog design, extending support to systems including RF, mixed-signal, photonics, and advanced heterogeneous designs. Innovative AI techniques, cloud enablement, infrastructure improvements, and seamless integration across Cadence products further streamline these design flows, creating an efficient hub for delivering real-world designs.

In a significant advancement, Cadence is rolling out its inaugural robust demonstration of a large language model (LLM) tool tailored for chip design, named Cadence ChipGPT. This innovative LLM is particularly advantageous in the design-cleanup phase, allowing engineers to cross-verify designs against specifications, diagnose and rectify potential discrepancies, and initiate analytical tasks with comprehensive explanations in human-readable language. This sophisticated tool facilitates various review processes, conserves substantial engineering hours, and diminishes the frequency of extensive group discussions. Additionally, it plays a critical role in detecting numerous bugs that may elude identification prior to regression verification (Freund, 2023).

The proof-of-concept for Cadence's ChipGPT signifies the commencement of a potentially protracted journey towards implementing large language models (LLMs) in the sphere of chip design. Nevertheless, those clients leveraging the JedAI platform have already demonstrated remarkable outcomes through this initiative, achieving a substantial

reduction in the time needed to transition from initial specifications to final designs while simultaneously improving the degree of design control.

Cadence emphasizes data security in scenarios where AI algorithms need to interact with sensitive intellectual property. The implementation of Cadence's LLM is conducted wholly on-premises, ensuring that all data is securely stored and managed within the confines of the Cadence.AI platform, safeguarded by the enterprise's firewall. The LLM's computational operations are executed on the server infrastructure provided by the customers, which can be based on either CPU or GPU architectures.

## 7.2 Siemens EDA Software

The company offers a full suite of EDA tools for designing PCBs, integrated circuits, and packaging. According to Siemens (2023), the integration of AI into its EDA tools began in 2018, driven by a shortage of experienced design engineers and the need to reduce time-to-market. AI has been deployed to automate various IC design processes, ensuring correct-by-design hardware and significantly improving design efficiency and accuracy.

The high-level synthesis and verification tool, Catapult, automatically transforms a SoC described in SystemC or C++ into a PPA-optimized RTL description. To enhance designer productivity, Catapult AI uses machine learning to convert Python code into designs for neural network (NN) hardware or AI accelerators. The design flow for Catapult AI NN is shown in Figure 7.1.

## 7.3 Synopsys EDA Software

Synopsys offers a wide range of EDA tools for designing monolithic SoCs and chiplet-based SoCs. It also provides access to its EDA toolchain through Software as a Service (SaaS) via cloud computing. The Fusion Design Platform is accessible through a hybrid cloud model. The platform uses AI to analyze and optimize design parameters and to automate intricate design tasks.

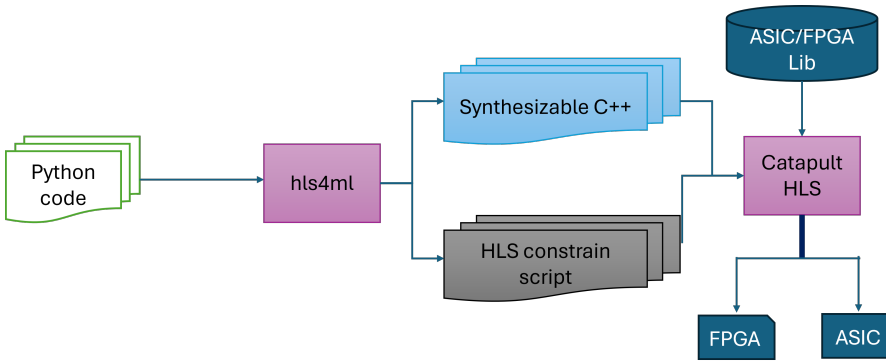


**Figure 7.1:** Design flow of Catapult NN.

**Table 7.1:** Summary of Synopsys AI solution.

| Application | Description |
|---|---|
| VSO.ai | AI-based verification space optimization |
| DSO.ai | Design space optimization AI |
| Design.da | End-to-End AI-driven data analytics platform |
| TSO.ai | AI-driven autonomous system for ATPG configuration and QoR optimization |
| Silicon.da | A comprehensive data analytics solution from design through manufacturing |
| Fab.da | Comprehensive AI-driven process analytics and control solution |

The company has a full stack of AI-driven EDA tools that span system architecture, design, and manufacturing. AI in the toolchain is used to optimize the design, data analytics, and generative AI. Generative AI facilitates the use of the EDA tools, better analyses reports, and streamlines workflows. It is also used to reduce the time spent on developing RTL and in the universal verification methodology. Figure 7.2 summarizes the use of generative AI, machine learning, and data analytics solutions in Synopsys's design tools. These solutions are summarized in Table 7.1.

## 7.4 Start-up AI EDA Companies

An unprecedented increase in the emergence of startup enterprises, often referred to as AI EDA firms, has been significantly driven by recent progress in artificial intelligence technologies.

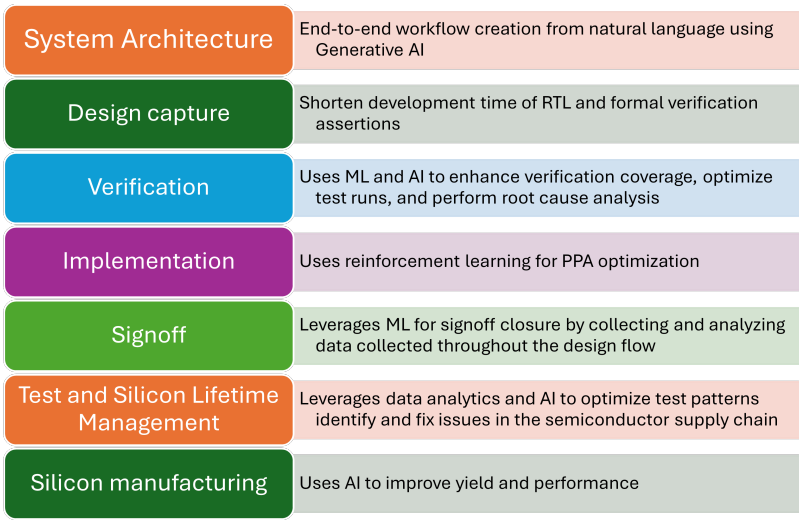


**Figure 7.2:** Integration of Generative AI, ML, and data analytics in Synopys's toolchain.

The company Primis AI (2025) has developed an AI-driven EDA tool known as RapidGPT. This tool greatly aids engineers by providing capabilities to write, optimize, and troubleshoot HDL code with elevated efficiency. RapidGPT extends support to VHDL, Verilog, and SystemVerilog languages. Additionally, the tool is equipped with a Retrieval Augmented Generation (RAG) database, which is a significant asset for hardware designers seeking to integrate intellectual properties (IPs) within their design processes.

Silimate, established in 2023, represents another emerging company in the field of AI-driven electronic design automation (EDA). The firm is actively working on bringing to market a co-pilot tool designed to assist designers in crafting Register Transfer Level (RTL) code that is both free from bugs and optimal in terms of power, performance, and area (PPA) efficiency (Silimate, Inc., 2025).

# 8

# Discussions

The integration of deep learning and generative AI into SoC and chiplet-based designs is rapidly transforming the landscape of hardware design and verification. This work has demonstrated the potential of these technologies in addressing some of the most critical challenges in design scalability, performance optimization, and hardware security. In particular, the application of AI-driven EDA tools like Siemens' Catapult, Cadence Cerebrus, and the emerging OpenROAD-Assistant illustrates how AI can enhance traditional design workflows, improving time-to-market and reducing the complexity of verification processes.

One of the most significant contributions of AI is its ability to optimize RTL code generation, physical design, and logic synthesis through automation. By leveraging machine learning models, tools can now handle vast design spaces, offering design space exploration with a level of efficiency that manual processes cannot match. AI-driven optimizations have shown promise in achieving improvements in power, performance, and area (PPA) metrics, making them particularly valuable in the era of heterogeneous chiplet architectures. For example, the use of deep learning for routing algorithms in chiplets, as discussed, has the potential to streamline complex design constraints while minimizing signal integrity and thermal issues.

Despite the progress, several challenges remain. One of the most critical issues is the interpretability of AI models in hardware design. While large language models (LLMs) and deep learning algorithms can optimize design processes, the lack of transparency in how decisions are made can be problematic, especially in security-critical applications. Explainable AI approaches will need to be integrated into the design flow to ensure that engineers can understand and trust AI-generated solutions.

Another challenge is the scalability of AI models, particularly when applied to increasingly complex SoC architectures. As chiplets and 3D-IC designs become more widespread, AI models must scale efficiently to handle the larger datasets and more intricate design requirements. Furthermore, integrating AI-generated designs into existing workflows, especially in environments with legacy systems, poses another hurdle. Ensuring compatibility and seamless integration with traditional EDA tools will be key to broader adoption.

In the process of validating Verilog designs, Large Language Models (LLMs) have exhibited significant potential in automating the generation of test stimuli. For example, LLMs have successfully generated test cases for simple combinational and sequential circuits. Nevertheless, the reliability of LLMs in verifying more complex designs, such as those with extensive concurrent processes and intricate module hierarchies, remains a challenge. This domain still demands additional exploration and detailed study. Future research should focus on enhancing LLMs' comprehension of large-scale Verilog code and integrating them with other AI techniques, such as Graph Neural Networks (GNNs), to improve their scalability and accuracy (Ma *et al.*, 2024).

The deployment of foundational models such as GPT-4 in the development of SoCs offers both significant potential benefits and certain challenges. These models are characterized by their unmatched flexibility, rendering them appropriate for a plethora of applications, including the generation of Hardware Description Language (HDL) via natural language, optimizing logic synthesis, and enhancing verification procedures. Moreover, their ability to integrate diverse types of data and deliver high-caliber solutions matches well with the growing complexity inherent in contemporary Electronic Design Automation (EDA) processes.

Despite their broad applicability, foundational models often fall short in terms of the domain-specific precision that custom language models, refined using hardware-specific datasets, can provide. These tailored models have the advantage of utilizing historical design data to boost outcomes in key tasks such as Register-Transfer Level (RTL) generation, verification, and Power-Performance-Area (PPA) optimization. As recent studies underline, meticulous fine-tuning and careful prompt engineering can greatly improve both the accuracy and utility of these models in the EDA domain.

Nevertheless, the shift towards using specialized models highlights a gap in accessibility. Bigger corporations, equipped with substantial internal datasets, ample computational power, and proprietary resources, are more capable of capitalizing on these benefits, potentially deepening the divide between leading industry players and smaller companies. Mitigating this disparity by promoting open-source datasets, fostering cooperative development, and ensuring democratized access to advanced models constitutes a vital avenue for further investigation (Zhong *et al.*, 2023).

# 9

# Conclusion

At the onset of the fourth industrial revolution and in the More than Moore technology the design of complex SoCs is drifting from monolithic towards chiplet-based designs. In the traditional design techniques of integrated circuits, the optimization focused on three metrics: area, speed, and power consumption. However, the explosion in IoT devices (low, middle, and high-end devices) has put critical infrastructure at increasing risk of cyber threats. This has been further exacerbated by the heavy reliance on third parties for the design of complex SoCs. These and other factors have put the security of SoCs at the forefront.

In this work, we have conducted a comprehensive review of the use of deep-learning models and LLMs for the design and verification of SoC systems. We have considered both monolithic and chiplet-based designs. LLMs, especially fine-tuned ones, can increase the productivity of the design process. However, the challenge in the wide acceptance of LLMs lies in the transparency of training or fine-tuning the LLMs. In addition, the security and privacy concerns related to the use of cloud computing to run the LLMs can pose a serious barrier.

# References

Ajayi, T., V. A. Chhabria, M. Fogaça, S. Hashemi, A. Hosny, A. B. Kahng, M. Kim, J. Lee, U. Mallappa, M. Neseem, G. Pradipta, S. Reda, M. Saligane, S. S. Sapatnekar, C. Sechen, M. Shalan, W. Swartz, L. Wang, Z. Wang, M. Woo, and B. Xu. (2019). "Toward an Open-Source Digital Flow: First Learnings from the OpenROAD Project". In: *Proceedings of the 56th Annual Design Automation Conference 2019. DAC '19*. Las Vegas, NV, USA: Association for Computing Machinery. DOI: 10.1145/3316781.3326334.

Bahdanau, D., K. Cho, and Y. Bengio. (2015). "Neural machine translation by jointly learning to align and translate". In: *3rd International Conference on Learning Representations, ICLR 2015*. 1–15. URL: https://arxiv.org/abs/1409.0473.

Ben Dhaou, I. and H. Tenhunen. (2002). "HIPED: a tool for high-level power estimation of digital signal processing algorithms". In: *9th International Conference on Electronics, Circuits and Systems*. Vol. 2. 729–732 vol.2. DOI: 10.1109/ICECS.2002.1046272.

Ben Dhaou, I. (2002). "Low Power Design Techniques for Deep Submicron Technology with Application to Wireless Transceiver Design". *PhD thesis*. KTH, Microelectronics and Information Technology, IMIT.

Bengesi, S., H. El-Sayed, M. K. Sarker, Y. Houkpati, J. Irungu, and T. Oladunni. (2024). "Advancements in Generative AI: A Comprehensive Review of GANs, GPT, Autoencoders, Diffusion Model, and Transformers". *IEEE Access*. 12: 69812–69837. DOI: 10.1109/ACCESS.2024.3397775.

Bhadra, J., M. S. Abadir, L.-C. Wang, and S. Ray. (2007). "A survey of hybrid techniques for functional verification". *IEEE Design & Test of Computers*. 24(02): 112–122.

Bhunia, S., M. S. Hsiao, M. Banga, and S. Narasimhan. (2014). "Hardware Trojan Attacks: Threat Analysis and Countermeasures". *Proceedings of the IEEE*. 102(8): 1229–1247. DOI: 10.1109/JPROC.2014.2334493.

Bhunia, S. and M. M. Tehranipoor. (2017). *The Hardware Trojan War: Attacks, Myths, and Defenses*. eng. 1st ed. Cham: Springer International Publishing AG.

Breuer, M., M. Sarrafzadeh, and F. Somenzi. (2000). "Fundamental CAD algorithms". *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*. 19(12): 1449–1475. DOI: 10.1109/43.898826.

Cadence Design Systems. (2025). "AI for Intelligent Design". URL: https://www.cadence.com/en_US/home/ai/overview.html.

Chang, Y., X. Wang, J. Wang, Y. Wu, L. Yang, K. Zhu, H. Chen, X. Yi, C. Wang, Y. Wang, W. Ye, Y. Zhang, Y. Chang, P. S. Yu, Q. Yang, and X. Xie. (2024). "A Survey on Evaluation of Large Language Models". *ACM Trans. Intell. Syst. Technol.* 15(3). DOI: 10.1145/3641289.

Chen, L., Y. Chen, Z. Chu, W. Fang, T.-Y. Ho, R. Huang, Y. Huang, S. Khan, M. Li, X. Li, Y. Li, Y. Liang, J. Liu, Y. Liu, Y. Lin, G. Luo, Z. Shi, G. Sun, D. Tsaras, R. Wang, Z. Wang, X. Wei, Z. Xie, Q. Xu, C. Xue, J. Yan, J. Yang, B. Yu, M. Yuan, E. F. Y. Young, X. Zeng, H. Zhang, Z. Zhang, Y. Zhao, H.-L. Zhen, Z. Zheng, B. Zhu, K. Zhu, and S. Zou. (2024). "The Dawn of AI-Native EDA: Opportunities and Challenges of Large Circuit Models". URL: https://arxiv.org/abs/2403.07257.

Chen, M., J. Tworek, H. Jun, Q. Yuan, H. P. de Oliveira Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman, A. Ray, R. Puri, G. Krueger, M. Petrov, H. Khlaaf, G. Sastry, P. Mishkin, B. Chan, S. Gray, N. Ryder, M. Pavlov, A. Power, L. Kaiser, M. Bavarian, C. Winter, P. Tillet, F. P. Such, D. Cummings, M. Plappert, F. Chantzis, E. Barnes, A. Herbert-Voss, W. H. Guss, A. Nichol, A. Paino, N. Tezak, J. Tang, I. Babuschkin, S. Balaji, S. Jain, W. Saunders, C. Hesse, A. N. Carr, J. Leike, J. Achiam, V. Misra, E. Morikawa, A. Radford, M. Knight, M. Brundage, M. Murati, K. Mayer, P. Welinder, B. McGrew, D. Amodei, S. McCandlish, I. Sutskever, and W. Zaremba. (2021). "Evaluating Large Language Models Trained on Code". *CoRR*. abs/2107.03374. URL: https://arxiv.org/abs/2107.03374.

Chen, T., S. Xiong, H. He, and B. Yu. (2023). "TRouter: Thermal-Driven PCB Routing via Nonlocal Crisscross Attention Networks". *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*. 42(10): 3388–3401. DOI: 10.1109/TCAD.2023.3243544.

Chen, W., S. Ray, J. Bhadra, M. Abadir, and L.-C. Wang. (2017). "Challenges and trends in modern SoC design verification". *IEEE Design & Test*. 34(5): 7–22.

Dehaerne, E., B. Dey, S. Halder, S. De Gendt, and W. Meert. (2022). "Code Generation Using Machine Learning: A Systematic Review". *IEEE Access*. 10: 82434–82455. DOI: 10.1109/ACCESS.2022.3196347.

Dong, S., P. Wang, and K. Abbas. (2021). "A survey on deep learning and its applications". *Computer Science Review*. 40: 100379. DOI: https://doi.org/10.1016/j.cosrev.2021.100379.

El Fentis, I. (2020). "Methodologies for SOC verification". *PhD thesis*. Politecnico di Torino.

Elfadel, I. M., D. S. Boning, and X. Li. (2019). *Machine learning in VLSI computer-aided design*. Springer.

Fang, J.-W., C.-H. Hsu, and Y.-W. Chang. (2009). "An Integer-Linear-Programming-Based Routing Algorithm for Flip-Chip Designs". *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*. 28(1): 98–110. DOI: 10.1109/TCAD.2008.2009151.

Fang, J.-W., I.-J. Lin, Y.-W. Chang, and J.-H. Wang. (2007). "A Network-Flow-Based RDL Routing Algorithmz for Flip-Chip Design". *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*. 26(8): 1417–1429. DOI: 10.1109/TCAD.2007.891364.

Farahmandi, F., M. S. Rahman, S. R. Rajendran, and M. Tehranipoor. (2023a). *CAD for hardware security*. Springer.

Farahmandi, F., M. S. Rahman, S. R. Rajendran, and M. Tehranipoor. (2023b). "CAD for Machine Learning in Hardware Security". In: *CAD for Hardware Security*. Springer. 211–230.

Farzana, N., F. Rahman, M. Tehranipoor, and F. Farahmandi. (2019). "Soc security verification using property checking". In: *2019 IEEE International Test Conference (ITC)*. IEEE. 1–10.

Freund, K. (2023). "Cadence Design is Working with Renesas to Build the World's First LLM Tool for Up-Front Chip Design". URL: https://www.forbes.com/sites/karlfreund/2023/09/11/cadence-design-is-working-with-renesas-to-build-the-worlds-first-llm-tool-for-up-front-chip-design/?sh=4da0c67b21a4.

Gandhi, U., I. Bustany, W. Swartz, and L. Behjat. (2019). "A Reinforcement Learning-Based Framework for Solving Physical Design Routing Problem in the Absence of Large Test Sets". In: *2019 ACM/IEEE 1st Workshop on Machine Learning for CAD (MLCAD)*. 1–6. DOI: 10.1109/MLCAD48534.2019.9142109.

Grimm, T., D. Lettnin, and M. Hübner. (2018). "A survey on formal verification techniques for safety-critical systems-on-chip". *Electronics*. 7(6).

Gupta, A. (1992). "Formal hardware verification methods: A survey". *Formal Methods in System Design*. 1: 151–238.

Hai, T., J. Zhou, N. Li, S. K. Jain, S. Agrawal, and I. Ben Dhaou. (2022). "Cloud-based bug tracking software defects analysis using deep learning". *Journal of Cloud Computing*. 11(1): 32. DOI: 10.1186/s13677-022-00311-8.

Hitchcock, C. and D. Thomas. (1983). "A Method of Automatic Data Path Synthesis". In: *20th Design Automation Conference Proceedings*. 484–489. DOI: 10.1109/DAC.1983.1585697.

Huang, G., J. Hu, Y. He, J. Liu, M. Ma, Z. Shen, J. Wu, Y. Xu, H. Zhang, K. Zhong, X. Ning, Y. Ma, H. Yang, B. Yu, H. Yang, and Y. Wang. (2021). "Machine Learning for Electronic Design Automation: A Survey". *ACM Trans. Des. Autom. Electron. Syst.* 26(5). DOI: 10.1145/3451179.

Hughes, W., S. Srinivasan, R. Suvarna, and M. Kulkarni. (2019). "Optimizing design verification using machine learning: Doing better than random". *arXiv preprint arXiv:1909.13168*.

Hussain, M., N. K. Baloach, G. Ali, M. ElAffendi, I. B. Dhaou, S. S. Ullah, and M. Uddin. (2023). "Hardware Trojan Mitigation Technique in Network-on-Chip (NoC)". *Micromachines*. 14(4). DOI: 10.3390/mi14040828.

Kabir, M. A. and Y. Peng. (2020). "Chiplet-Package Co-Design For 2.5D Systems Using Standard ASIC CAD Tools". In: *2020 25th Asia and South Pacific Design Automation Conference (ASP-DAC)*. 351–356. DOI: 10.1109/ASP-DAC47756.2020.9045734.

Kaplan, J., S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, and D. Amodei. (2020). "Scaling laws for neural language models". *CoRR*. abs/2001.08361. URL: https://arxiv.org/abs/2001.08361.

Kim, J. and J. McDermott. (1983). "TALIB: an IC layout design assistant". In: *Proceedings of the Third AAAI Conference on Artificial Intelligence. AAAI'83*. Washington, D.C.: AAAI Press. 197–201.

Kim, N., J. Park, B. Min, and W. Park. (2013). "Register verification: Do we have reliable specification?" In: *Design and Verification Conference and Exhibition*.

Kirk, R. S. (1985). "The impact of AI technology on VLSI design". In: *Managing Requirements Knowledge, International Workshop on*. Los Alamitos, CA, USA: IEEE Computer Society. 125. DOI: 10.1109/AFIPS.1985.63.

Koblah, D., R. Acharya, D. Capecci, O. Dizon-Paradis, S. Tajik, F. Ganji, D. Woodard, and D. Forte. (2023). "A Survey and Perspective on Artificial Intelligence for Security-Aware Electronic Design Automation". eng. *ACM transactions on design automation of electronic systems*. 28(2): 1–57.

Kowalski, T. and D. Thomas. (1983). "The VLSI Design Automation Assistant: Prototype System". In: *20th Design Automation Conference Proceedings*. 479–483. DOI: 10.1109/DAC.1983.1585696.

Lee, J.-H. and S.-C. Kim. (2011). "Analysis of Verification Methodologies Based on a SoC Platform Design". *International Journal of Contents*. 7(1): 23–28.

Liao, W.-S. and P.-A. Hsiung. (2003). "FVP: A formal verification platform for SoC". In: *IEEE International [Systems-on-Chip] SOC Conference, 2003. Proceedings*. IEEE. 21–24.

Lin, T.-C., D. Merrill, Y.-Y. Wu, C. Holtz, and C.-K. Cheng. (2021). "A Unified Printed Circuit Board Routing Algorithm With Complicated Constraints and Differential Pairs". In: *2021 26th Asia and South Pacific Design Automation Conference (ASP-DAC)*. 170–175.

Liu, Y., X. Li, and S. Yin. (2024). "Review of chiplet-based design: system architecture and interconnection". eng. *Science China. Information sciences*. 67(10).

Ma, R., Y. Yang, Z. Liu, J. Zhang, M. Li, J. Huang, and G. Luo. (2024). "VerilogReader: LLM-Aided Hardware Test Generation". In: *2024 IEEE LLM Aided Design Workshop (LAD)*. 1–5. DOI: 10.1109/LAD62341.2024.10691801.

Mangalagiri, P., L. Qian, F. Zafar, P. Mosalikanti, P. Chang, A. Kurian, and V. Saripalli. (2024). "CDLS: Constraint Driven Generative AI Framework for Analog Layout Synthesis". In: *Proceedings of the 61st ACM/IEEE Design Automation Conference. DAC '24*. San Francisco, CA, USA: Association for Computing Machinery. DOI: 10.1145/3649329.3656232.

McMurchie, L. and C. Ebeling. (1995). "PathFinder: A Negotiation-Based Performance-Driven Router for FPGAs". In: *Third International ACM Symposium on Field-Programmable Gate Arrays*. 111–117. DOI: 10.1109/FPGA.1995.242049.

Nasser, Y., J. Lorandel, J.-C. Prévotet, and M. Hélard. (2021). "RTL to Transistor Level Power Modeling and Estimation Techniques for FPGA and ASIC: A Survey". *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*. 40(3): 479–493. DOI: 10.1109/TCAD.2020.3003276.

Nath, A. P. D., S. Ray, A. Basak, and S. Bhunia. (2018). "System-on-chip security architecture and CAD framework for hardware patch". In: *2018 23rd Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE. 733–738.

Naveed, H., A. U. Khan, S. Qiu, M. Saqib, S. Anwar, M. Usman, N. Akhtar, N. Barnes, and A. Mian. (2023). "A Comprehensive Overview of Large Language Models". *arXiv preprint arXiv:2307.06435*. URL: https://arxiv.org/abs/2307.06435.

OpenAI. (2024). "OpenAI Codex". URL: https://openai.com/index/openai-codex/.

Ozdal, M. and M. Wong. (2004). "Simultaneous escape routing and layer assignment for dense PCBs". In: *IEEE/ACM International Conference on Computer Aided Design, 2004. ICCAD-2004.* 822–829. DOI: 10.1109/ICCAD.2004.1382689.

Primis AI. (2025). "Primis AI: Welcome to the Future of Hardware Design". URL: https://primis.ai/.

Rabaey, J. M. (2009). *Low Power Design Essentials*. Boston, MA: Springer. DOI: 10.1007/978-0-387-71713-5.

Rajendran, S. R. (2020). "Security challenges in hardware used for smart environments". In: *Internet of Things and Secure Smart Environments*. Chapman and Hall/CRC. 363–381.

Rama, E., M. Ayache, R. Buchty, B. Bauer, M. Korb, M. Berekovic, and S. Mulhem. (2024). "Trustworthy Integrated Circuits: From Safety to Security and Beyond". *IEEE Access*. 12: 69603–69632. DOI: 10.1109/ACCESS.2024.3400685.

Rapp, M., H. Amrouch, Y. Lin, B. Yu, D. Z. Pan, M. Wolf, and J. Henkel. (2022). "MLCAD: A Survey of Research in Machine Learning for CAD Keynote Paper". *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*. 41(10): 3162–3181. DOI: 10.1109/TCAD.2021.3124762.

Rashinkar, P., P. Paterson, and L. Singh. (2007). *System-on-a-chip Verification: Methodology and Techniques*. Springer Science & Business Media.

Ravikumar, C. (2024). "Chiplets for Integration of Electronic Systems". *IETE Journal of Education*. 0(0): 1–17. DOI: 10.1080/09747338.2024.2311372.

Ray, S., Y. Jin, and A. Raychowdhury. (2016). "The changing computing paradigm with internet of things: A tutorial introduction". *IEEE Design & Test*. 33(2): 76–96.

Reda, S. and A. N. Nowroz. (2012). "Power Modeling and Characterization of Computing Devices". *Foundations and Trends® in Electronic Design Automation*. 6(2): 121–216. DOI: 10.1561/1000000022.

Roy, S. K. (2007). "Top level SOC interconnectivity verification using formal techniques". In: *2007 Eighth International Workshop on Microprocessor Test and Verification*. IEEE. 63–70.

Saha, D., S. Tarek, K. Yahyaei, S. K. Saha, J. Zhou, M. Tehranipoor, and F. Farahmandi. (2024). "LLM for SoC Security: A Paradigm Shift". *IEEE Access*: 1–1. DOI: 10.1109/ACCESS.2024.3427369.

Schuermans, S. and R. Leupers. (2019). *Power Estimation on Electronic System Level Using Linear Power Models*. Cham, Switzerland: Springer. DOI: 10.1007/978-3-030-01875-7.

Seshia, S. A. (2012). "Sciduction: Combining induction, deduction, and structure for verification and synthesis". In: *Proceedings of the 49th Annual Design Automation Conference*. 356–365.

Sharma, U., B.-Y. Wu, S. R. D. Kankipati, V. A. Chhabria, and A. Rovinski. (2024). "OpenROAD-Assistant: An Open-Source Large Language Model for Physical Design Tasks". In: *Proceedings of the 2024 ACM/IEEE International Symposium on Machine Learning for CAD. MLCAD '24*. Salt Lake City, UT, USA: Association for Computing Machinery. DOI: 10.1145/3670474.3685960.

Siemens. (2023). "A New Era of EDA Powered by AI". URL: https://resources.sw.siemens.com/en-US/white-paper-a-new-era-of-eda-powered-by-AI.

Silimate, Inc. (2025). "Silimate: Streamlining Chip Design Workflows". URL: %7Bhttps://www.silimate.com/%7D.

Talupur, M., S. Ray, and J. Erickson. (2015). "Transaction flows and executable models: Formalization and analysis of message-passing protocols". In: *2015 Formal Methods in Computer-Aided Design (FMCAD)*. IEEE. 168–175.

Vaithianathan, M., M. Patil, S. F. Ng, and S. Udkar. (2024). "Integrating AI and Machine Learning with UVM in Semiconductor Design". *ESP International Journal of Advancements in Computational Technology (ESP-IJACT) Volume*. 2: 37–51.

Vaswani, A., N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. (2017). "Attention is all you need". In: *Advances in Neural Information Processing Systems*. Vol. 30. 5998–6008. URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Wang, S., H. Zheng, X. Wen, K. Xu, and H. Tan. (2024a). "Enhancing chip design verification through AI-powered bug detection in RTL code". *Applied and Computational Engineering*. 92: 27–33.

Wang, Z., L. Alrahis, L. Mankali, J. Knechtel, and O. Sinanoglu. (2024b). "LLMs and the Future of Chip Design: Unveiling Security Risks and Building Trust". In: *2024 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*. 385–390. DOI: 10.1109/ISVLSI61997.2024.00076.

Wu, B.-Y., U. Sharma, S. R. D. Kankipati, A. Yadav, B. K. George, S. R. Guntupalli, A. Rovinski, and V. A. Chhabria. (2024a). "EDA Corpus: A Large Language Model Dataset for Enhanced Interaction with OpenROAD". In: *2024 IEEE LLM Aided Design Workshop (LAD)*. 1–5. DOI: 10.1109/LAD62341.2024.10691774.

Wu, H., Z. He, X. Zhang, X. Yao, S. Zheng, H. Zheng, and B. Yu. (2024b). "ChatEDA: A Large Language Model Powered Autonomous Agent for EDA". *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*. 43(10): 3184–3197. DOI: 10.1109/TCAD.2024.3383347.

Yan, T. and M. D. F. Wong. (2012). "Correctly Modeling the Diagonal Capacity in Escape Routing". *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*. 31(2): 285–293. DOI: 10.1109/TCAD.2011.2169258.

Yu, M.-F. and W. Wei-Ming Dai. (1995). "Single-layer fanout routing and routability analysis for ball grid arrays". In: *Proceedings of IEEE International Conference on Computer Aided Design (ICCAD)*. 581–586. DOI: 10.1109/ICCAD.1995.480175.

Zhao, W. X., K. Zhou, J. Li, T. Tang, X. Wang, Y. Hou, Y. Min, B. Zhang, J. Zhang, Z. Dong, Y. Du, C. Yang, Y. Chen, Z. Chen, and J. Jiang. (2023). "A Survey of Large Language Models". *arXiv preprint arXiv:2303.18223*.

Zhong, R., X. Du, S. Kai, Z. Tang, S. Xu, H.-L. Zhen, J. Hao, Q. Xu, M. Yuan, and J. Yan. (2023). "LLM4EDA: Emerging Progress in Large Language Models for Electronic Design Automation". URL: https://arxiv.org/abs/2401.12224.

# Large Language Models for EDA: From Assistants to Agents

Zhuolun He[1*], Yuan Pu[1], Haoyuan Wu[1], Yuhan Qin[2], Tairu Qiu[2] and Bei Yu[1*]

[1] *The Chinese University of Hong Kong, Hong Kong*
[2] *ChatEDA Tech, Hong Kong*

ABSTRACT

This survey explores the application of Large Language Models (LLMs) in Electronic Design Automation (EDA), covering their roles as both assistants and autonomous agents. We review current research and practical implementations where LLMs are utilized for tasks such as question answering, script generation, and automated design processes. This work highlights the benefits of LLMs, including enhanced productivity and innovation, while also addressing challenges like accuracy and integration with traditional EDA tools. Furthermore, we discuss the evolution from LLMs as supportive assistants to more sophisticated agents capable of handling complex EDA workflows. This work aims to provide a comprehensive overview and guide future advancements in the integration of LLMs within the EDA domain.

*Corresponding authors: Zhuolun He, zleonhe@gmail.com, and Bei Yu, byu@cse.cuhk.edu.hk.
Zhuolun He, Yuan Pu, Haoyuan Wu, Yuhan Qin, Tairu Qiu and Bei Yu (2025), "Large Language Models for EDA: From Assistants to Agents", Foundations and Trends® in Electronic Design Automation: Vol. 14, No. 4, pp 295–314. DOI: 10.1561/1000000063.

# 1

# Introduction

Electronic Design Automation (EDA) is a critical discipline within the field of electronics and computer engineering, encompassing the tools and methodologies used for designing, verifying, and testing electronic systems. As integrated circuits (ICs) have grown in complexity, so too has the need for sophisticated EDA tools to manage this complexity. These tools are essential for automating the design process, ensuring that ICs meet performance, power, and area (PPA) requirements, and enabling the rapid development of new technologies.

The advent of Artificial Intelligence (AI) and, more specifically, Machine Learning (ML) techniques, has brought about a paradigm shift in the capabilities of EDA tools. AI for EDA leverages learning algorithms to improve various aspects of the design flow, from synthesis and placement to timing analysis and power optimization. By harnessing the power of data, these AI-driven solutions can predict outcomes, optimize designs, and even learn from past projects, thereby significantly accelerating the design cycle and improving the quality of results (QoR).

In recent years, Large Language Models (LLMs) have emerged as a transformative force in the domain of natural language processing (NLP), with applications spanning a wide array of fields. LLMs, such

as GPT-4, BERT, and their successors, are characterized by their vast parameter counts and ability to generate or understand human-like text based on the context provided. Figure 1.1 shows a timeline of existing large language models, cited from Zhao *et al.* (2023). When applied to the realm of EDA, LLMs have the potential to revolutionize the way designers interact with EDA tools, transforming them from passive assistants into active agents capable of performing complex tasks autonomously.

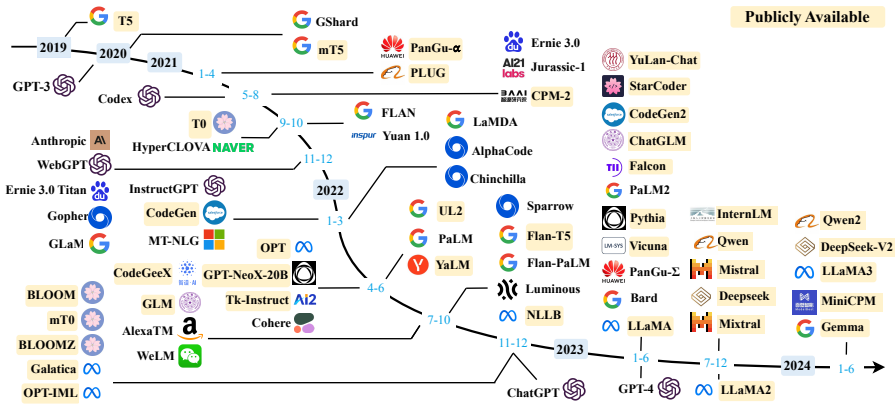


**Figure 1.1** A timeline of existing large language models (Zhao *et al.*, 2023).

This survey aims to provide a comprehensive overview of the current state of research and application of LLMs in the context of EDA. We will explore how LLMs function as assistants, where they augment the capabilities of designers by offering instant access to information, automating routine inquiries, and facilitating knowledge management. Furthermore, we will delve into their role as agents, where they go beyond simple assistance and actively participate in the design process, from generating and optimizing scripts to performing complex analysis and suggesting innovative solutions.

Through this exploration, we hope to highlight the significant contributions that LLMs can make to the EDA community, and to inspire further research and development in this exciting and rapidly evolving area.

# 2

# LLMs as Design Assistants

In the dynamic and increasingly complex field of Electronic Design Automation (EDA), the integration of Large Language Models (LLMs) is ushering in a new era of intelligent assistance, fundamentally transforming the way engineers approach design, verification, and debugging. These advanced AI models, with their profound natural language understanding and extensive knowledge base, are becoming indispensable tools that augment the capabilities of EDA professionals. By serving as sophisticated assistants, LLMs can provide instant and accurate responses to a wide range of technical queries, enhancing the question-answering (QA) process and facilitating more informed decision-making. Additionally, they can meticulously review design specifications, ensuring compliance and identifying potential discrepancies or ambiguities early in the design cycle. Furthermore, LLMs are proving to be valuable in bug analysis, where they can help diagnose and resolve issues by offering detailed and contextually relevant insights. This section introduces the role of LLMs as intelligent assistants in EDA, setting the stage for a comprehensive exploration of their applications in QA, specification review, and bug analysis in the following sections.

## 2.1 Question-Answering

HDLCopilot (Abdelatty and Reda, 2024), introduced by Manar Abdelatty and Sherief Reda, is an LLM-powered system designed to facilitate the querying of Process Design Kits (PDKs) using natural language. PDKs, which contain standard cell libraries and various design rules, are essential for the synthesis of abstract circuit definitions into manufacturable chips. Navigating these complex libraries is often time-consuming and error-prone. HDLCopilot leverages LLMs to enable hardware de-

sign engineers to interact with PDKs more efficiently, allowing them to retrieve specific information about gates or design rules through natural language queries. The system achieves an accuracy of 94.23% on an evaluation set of diverse and complex queries, positioning itself as a powerful tool for enhancing productivity and reducing human errors in the hardware design process.

RAG-EDA (Pu *et al.*, 2024) presents a customized retrieval augmented generation (RAG) framework tailored for EDA tool documentation question-answering (QA). Their approach includes a contrastive learning scheme for fine-tuning text embedding models, a reranker distilled from proprietary LLMs, and a generative LLM fine-tuned with high-quality domain-specific data. They also introduce ORD-QA, a benchmark for evaluating QA performance on OpenROAD, an advanced RTL-to-GDSII design platform. Experimental results demonstrate that their proposed RAG flow and techniques achieve superior performance on both the ORD-QA benchmark and a commercial EDA tool, outperforming state-of-the-art methods. This work underscores the importance of customizing LLMs for specialized domains to enhance their effectiveness and reliability. Figure 2.1 illustrates the overall flow of RAG-EDA.

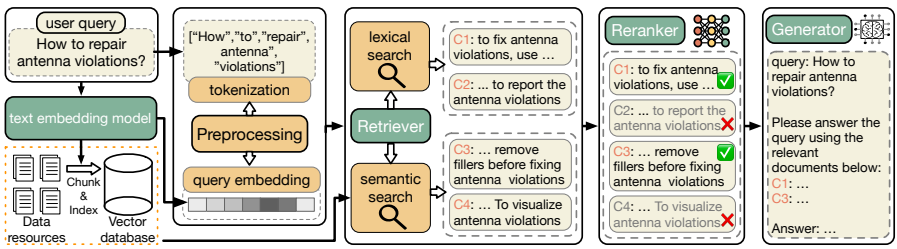


**Figure 2.1** Overall flow of RAG-EDA (Pu *et al.*, 2024).

Wu *et al.* (2024a) introduce EDA Corpus, an open-source dataset tailored for OpenROAD, a widely adopted open-source EDA toolchain. The dataset features over 1,000 data points and is structured in two formats: a pairwise set of question prompts with prose answers and a pairwise set of code prompts with corresponding OpenROAD scripts. By providing this dataset, the authors aim to facilitate LLM-focused research within the EDA domain. The dataset is designed to train LLMs

on answering questions about physical design methods and generating OpenROAD scripts, thereby enhancing the interaction between designers and EDA tools. The evaluation shows that fine-tuning LLMs with EDA Corpus leads to improved performance on physical design-specific tasks, highlighting the critical role of tailored datasets in advancing LLM-assisted EDA.

ORAssistant (Kaintura *et al.*, 2024) is a Retrieval-Augmented Generation (RAG) based conversational assistant designed to support users of OpenROAD. Google Gemini serves as the base LLM, which has been fine-tuned for the specific domain of EDA and chip design. To build the dataset, ORAssistant extracts conversations from GitHub Issues and Discussions related to OpenROAD and OpenROAD-flow-scripts, categorizing them into JSONL datasets using an LLM-based categorization approach. These categories include bug reports, feature requests, runtime issues, and more, which are then used to train the system to provide relevant and accurate information. For evaluation, the team used two QA datasets: a custom curated HumanEval dataset with 50 OpenROAD-related questions and 100 questions from the EDA Corpus dataset. Evaluation results showed that ORAssistant significantly outperformed base pre-trained LLMs, achieving high precision and recall scores, and a notably higher LLMScore. Additionally, ORAssistant demonstrated faster response times compared to GPT-4o, while being slightly slower than the base Gemini 1.5 Flash.

Xu *et al.* (2024) introduce ChipExpert, the first open-source, instructional LLM specifically tailored for the IC design field. Trained on the Llama-3 8B model, ChipExpert undergoes a rigorous training process that includes data preparation, continued pre-training, instruction-guided supervised fine-tuning, and preference alignment. The dataset used for training is constructed through manual selection and data synthesis, ensuring high quality and relevance. ChipExpert is designed to respond to user queries professionally, acquiring a vast amount of IC design knowledge. To mitigate hallucinations, the model is integrated with a Retrieval-Augmented Generation (RAG) system, making it a robust and reliable assistant for IC design tasks. The availability of ChipExpert and its associated dataset marks a significant step towards democratizing access to advanced LLMs in the IC design community.

Han *et al.* (2023) propose a new interaction paradigm for complex EDA software, leveraging GPT and other LLMs. They developed SmartonAI, an AI interaction assist plugin for EDA software, with KiCad as the initial example. SmartonAI breaks down designer requests into subtasks, such as analyzing help documentation and executing different plugins, and leverages the built-in schematic and PCB manipulation functions. The preliminary results show that SmartonAI can significantly streamline the PCB design process by simplifying complex commands into intuitive language-based interactions. This work demonstrates the potential of LLMs to bridge the gap between complex EDA software and user-friendly interaction, enhancing the productivity of EDA engineers.

Ask-EDA (Shi *et al.*, 2024) is a chat agent designed to serve as a 24/7 expert available to provide guidance to design engineers. Ask-EDA leverages LLMs, hybrid retrieval augmented generation (RAG), and abbreviation de-hallucination (ADH) techniques to deliver more relevant and accurate responses. The authors curated three evaluation datasets — q2a-100, cmds-100, and abbr-100 — to assess general design question answering, design command handling, and abbreviation resolution, respectively. The evaluation results show that hybrid RAG offers significant improvements in recall, while ADH enhances the resolution of abbreviations, making Ask-EDA an effective tool for addressing design-related inquiries.

## 2.2 Specification and Documentation Generation

Fernando *et al.* (2024) explore using LLMs to generate human-readable design documentation from formal hardware models. In their proposed workflow, a formal specification of a hardware system is pre-processed into prompts that are then fed to an LLM. These prompts are crafted using a generic template, which includes all necessary instructions but leaves out technical details. The technical information, such as port and bit widths, is extracted directly from the formal Unified Specification Format (USF) model and inserted into the template. This method allows the prompt to be repurposed across different USF models, facilitating a more scalable solution. The generated documentation is evaluated based on the manual effort required to review and revise it. The analysis

reveals that only 37.4% of the effort needed for fully manual writing is required when using LLM-generated documentation. This translates to a 2.7-fold increase in productivity. The quality of the generated text is also assessed, with approximately 62% of the documentation needing no corrections, 14% requiring minor revisions, and 19% necessitating major changes. An additional 5% falls under the category of unusable, requiring complete rewrites. To account for the reading and classification of parts that need correction, the researchers added an estimated 10% work effort, resulting in the aforementioned 37.4% total effort.

Li *et al.* (2024) introduce SpecLLM, a framework that leverages LLMs for the generation and review of VLSI design specifications, addressing the time-consuming and error-prone nature of manual specification development. They propose a structured definition of architecture specifications, categorizing them into three abstraction levels: Highest-level (HAS), Middle-level (MAS), and Lowest-level (LAS). To support this, they have compiled and released a dataset comprising 46 architecture specification documents from various sources. The authors explore two primary applications of LLMs: generating new architecture specifications, which includes writing specifications from scratch and converting RTL code into detailed specifications, and reviewing existing specifications to identify errors and inconsistencies. The study demonstrates that LLMs can effectively generate and review specifications, with the potential to significantly streamline the design process. However, they also highlight the importance of evaluating LLM-generated feedback, suggesting that training a specialized model for this purpose could improve the reliability of the review outcomes. Additionally, the work addresses challenges such as the lack of unified writing standards in the industry, the varying lengths of specifications, and the token limitations of current LLMs, which can affect the accuracy of the generated output.

## 2.3 Feedback Analysis

Qiu *et al.* (2024) explore the use of LLMs to generate novice-friendly explanations of compile-time synthesis error messages from EDA tools like Quartus Prime and Vivado (Figure 2.2). Training new engineers in digi-

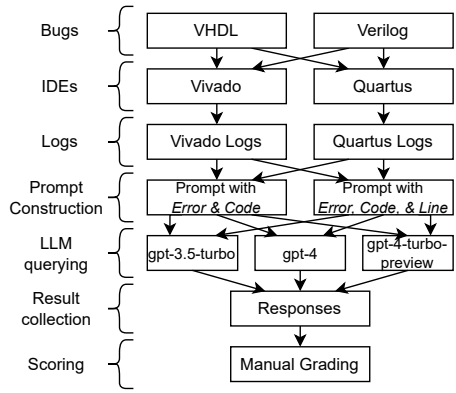


**Figure 2.2** LLM improves EDA tool usability (Qiu *et al.*, 2024).

tal design, particularly in the use of complex EDA tools, is a significant challenge. LLMs, with their text comprehension and question-answering capabilities, can provide clear and understandable explanations of synthesis errors, helping novices overcome common hurdles. The study generates 936 error message explanations using three OpenAI LLMs over 21 different buggy code samples and finds that in approximately 71% of cases, the LLMs provide correct and complete explanations suitable for novice learners. This work highlights the potential of LLMs in improving the learning and usability of EDA tools.

Wang (2024) present an innovative approach to semiconductor test data analytics by incorporating LLMs into an AI agent named IEA-Plot. The work advocates for an end-to-end methodology that places a Knowledge Graph (KG) at its core, facilitating the analysis of complex wafermap data. By leveraging LLMs, the system is capable of advanced natural language-driven analytics, which utilizes GPT-3 and subsequent models for instruction parsing, KG construction, and so on. The research highlights the practical implementation of this AI agent, showcasing its ability to conduct comprehensive data exploration and failure pattern analysis, while also addressing the challenges of reducing manual effort, especially in the context of concept-centric grounding. Despite these advancements, the work acknowledges the ongoing necessity for further research to fully automate the process, particularly in areas where human expertise remains indispensable.

# 3

# LLMs as Autonomous Agents

In the rapidly advancing field of EDA, Large Language Models (LLMs) are not only serving as powerful assistants but are also emerging as autonomous agents capable of performing complex and specialized tasks. As agents, LLMs go beyond merely providing information and support; they actively participate in the design and automation processes, significantly enhancing the efficiency and effectiveness of EDA workflows. These models leverage their deep understanding of natural language and extensive knowledge bases to generate flow scripts, assist in layout design, and even contribute to the intricate domain of analog design. By automating these tasks, LLMs can reduce the manual effort required, minimize errors, and accelerate the design cycle. This section introduces the role of LLMs as agents in EDA, setting the stage for a detailed exploration of their applications in flow script generation, layout assistance, and analog design in the following sections.

## 3.1 Flow Script Generation

ChatEDA (He *et al.*, 2023; Wu *et al.*, 2024b) introduces an innovative autonomous agent framework leveraging large language models to streamline EDA processes. As shown in Figure 3.1, this system leverages

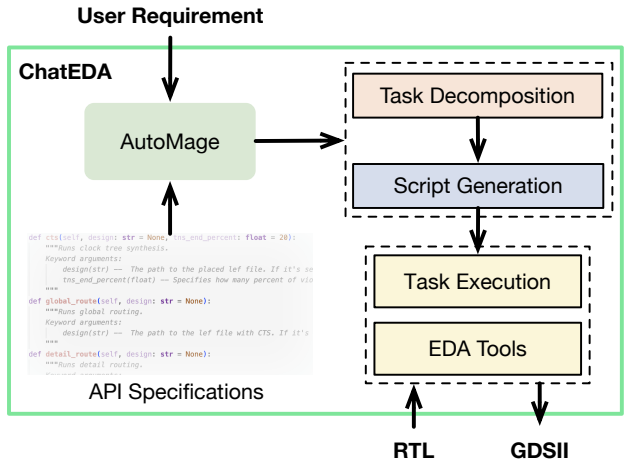


**Figure 3.1** LLM as agents: ChatEDA (He *et al.*, 2023)

a fine-tuned LLM (AutoMage) to interpret natural language instructions and generate executable scripts for EDA tools, streamlining the design flow from RTL to GDSII. This system, enhanced through instruction tuning and low-rank adaptation, demonstrates superior performance over generic LLMs like GPT-4, significantly improving the reliability and efficiency of EDA workflows.

IICPilot (Jiang *et al.*, 2024) introduces an innovative LLM-based multi-agent system designed to automate the backend design of integrated circuits using open-source EDA tools. This framework leverages the advanced reasoning and natural language understanding capabilities of large language models to significantly reduce the complexity and expertise required for IC backend design. Key contributions include the development of specialized agents for tasks such as script generation, EDA tool invocation, design space exploration, and computing resource allocation, all orchestrated through a unified EDA calling interface. By automating these processes, IICPilot not only enhances the accessibility and usability of open-source EDA tools but also improves the efficiency and performance of IC design workflows.

OpenROAD-Assistant (Sharma *et al.*, 2024) showcases the application of large language models as autonomous agents in EDA, specifically

for physical design tasks. Leveraging the Llama3-8B foundation model and employing retrieval-aware fine-tuning (RAFT), this open-source chatbot generates natural language responses and Python scripts using OpenROAD APIs. Unlike previous approaches that relied on proprietary tools, OpenROAD-Assistant uses only public data, making it accessible to new and experienced chip designers alike. The model outperforms other foundational models, achieving high scores in both scripting (77% pass@1, 80% pass@3) and question-answering (98% BERTScore, 96% BARTScore), demonstrating its potential to enhance productivity and accessibility in the EDA domain.

## 3.2 Layout Helper

ChatPattern (Wang *et al.*, 2024) introduces a novel framework leveraging Large Language Models as autonomous agents for flexible layout pattern customization. This system, composed of an expert LLM agent and a controllable layout pattern generator, interprets natural language instructions to perform complex design tasks, such as generating high-quality, large-scale VLSI layout patterns. The LLM agent not only understands user requirements but also operates design tools to meet specific design needs, demonstrating significant advancements in automating the creation of diverse and scalable pattern libraries, which are crucial for various DFM studies and machine learning applications in lithography design.

Chen *et al.* (2024a) present a novel framework that integrates reinforcement learning (RL) and large language models to automate the development of optical proximity correction (OPC) recipes in semiconductor manufacturing. The paper demonstrates how RL can optimize the design of EPE measurement and fragmentation recipes, while LLMs are employed to summarize and generalize these RL-generated recipes into practical, rule-based guidelines. This two-stage approach not only achieves significant improvements in key error metrics but also maintains the efficiency of the OPC process, making it immediately applicable to new layouts without additional optimization time. The study highlights the potential of LLMs as autonomous agents in layout design, enhancing both the effectiveness and efficiency of complex design tasks.

### 3.3 Analog Design

ADO-LLM (Yin *et al.*, 2024) introduces a novel framework that integrates Large Language Models with Bayesian Optimization (BO) to enhance the efficiency and effectiveness of analog circuit design. By leveraging the rich domain knowledge embedded in LLMs and their capability for in-context learning, ADO-LLM addresses the limitations of traditional BO methods, such as the inefficiency in exploring high-dimensional design spaces and the tendency to produce non-viable solutions. The framework consists of a Gaussian Process-based BO proposer, an LLM agent, a high-quality data sampler, and an HSPICE simulator. The LLM agent is initialized with circuit definitions and design specifications, and through iterative learning, it generates innovative and feasible design points that are then evaluated and used to refine future iterations. This cooperative approach not only accelerates the design convergence but also enables the exploration of a broader design space, demonstrating significant improvements in design efficiency and effectiveness on two different analog circuits.

Chen *et al.* (2024b) introduce the LLANA framework, which also leverages Large Language Models to enhance Bayesian Optimization for the automated generation of analog layout constraints. Addressing the limitations of traditional BO methods, such as slow convergence and high data requirements, LLANA exploits the few-shot learning capabilities of LLMs to efficiently explore the analog circuit design space and generate design-dependent parameters. By integrating LLMs' contextual understanding and learning efficiency, LLANA not only matches the performance of state-of-the-art BO methods but also demonstrates superior exploration of the design space. This innovative approach underscores the potential of LLMs as autonomous agents in accelerating the design process and improving the accuracy and reliability of analog layout synthesis. The authors provide an open-source implementation of their framework, facilitating further research and practical applications in the field.

Liu *et al.* (2024b) introduce AmpAgent, a pioneering multi-agent system leveraging large language models for the autonomous design of multi-stage amplifiers. AmpAgent is composed of three specialized

agents: the Literature Analysis Agent, which extracts critical information from academic literature; the Mathematics Reasoning Agent, which performs complex calculations and derives key design parameters; and the Device Sizing Agent, which integrates with simulation tools to optimize device parameters. The system addresses the challenges posed by the complexity and data scarcity in analog circuit design, demonstrating significant improvements over conventional methods in terms of design efficiency, iteration reduction, and performance enhancement. By effectively decomposing the design process and leveraging the reasoning capabilities of LLMs, AmpAgent showcases the potential of LLMs in advancing EDA for analog circuits, particularly in the rapid and accurate design of multi-stage amplifiers.

LayoutCopilot (Liu *et al.*, 2024a) introduces a pioneering framework that leverages LLMs as autonomous agents to enhance the efficiency and usability of analog layout design. Addressing the steep learning curve and cumbersome user experience associated with traditional design tools, LayoutCopilot simplifies human-tool interaction by translating natural language instructions into executable commands and interpreting high-level design intents into actionable suggestions. Utilizing a multi-agent collaborative system, the framework progressively transforms designer requirements into precise layout adjustments, demonstrating high accuracy and flexibility in handling real-world analog designs. Extensive testing reveals that LayoutCopilot significantly improves the functional correctness and analytical reasoning of layout outputs, particularly when guided by specific instructions.

GLayout (Hammoud *et al.*, 2024) introduces a novel approach to automating analog layout design. Unlike conventional methods that utilize Graph Convolutional Neural Networks (GCNs) and specific constraint files, GLayout employs LLMs to interpret human language prompts and transform them into compact, technology-generic text representations of analog layouts. This framework uses fine-tuning and in-context learning with Retrieval Augmented Generation (RAG) to enable the LLM to generate layouts for unseen circuits effectively, as shown in Figure 3.2. The models range from 3.8 to 22 billion parameters, showing enhanced performance with larger models, and have been validated on an open-source evaluation set that includes circuits from

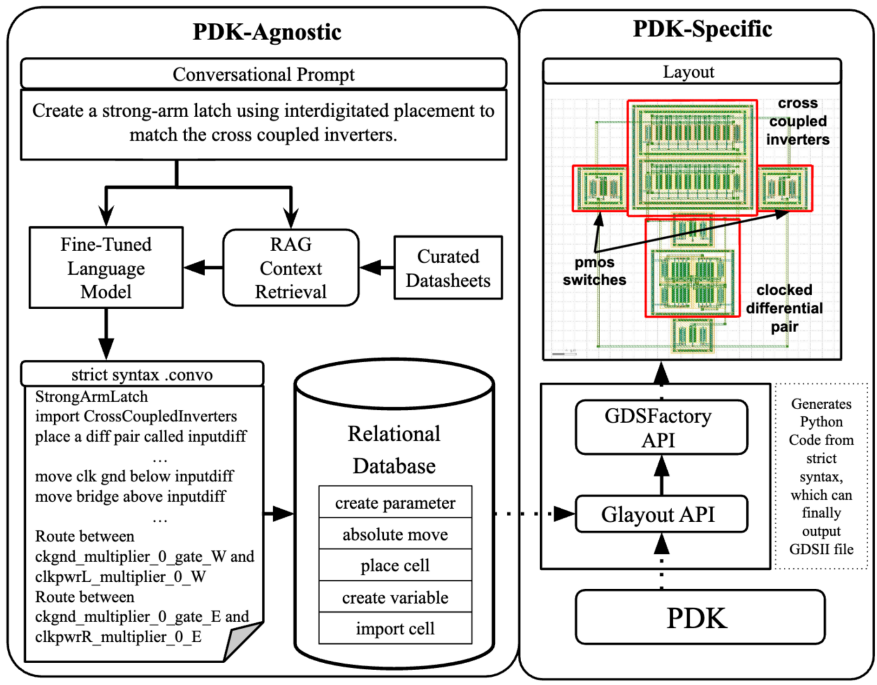


**Figure 3.2** LLM as agents: GLayout (Hammoud *et al.*, 2024)

simple 2-transistor designs to complex $\Delta\Sigma$ ADCs. This approach not only simplifies the design process but also sets a standard benchmark for evaluating LLM-based EDA tools, advancing the field towards more intuitive and versatile automated design solutions.

SPICED (Chaudhuri *et al.*, 2024) introduces a novel framework which operates entirely in the software domain, eliminating the need for hardware modifications and achieving high accuracy in detecting and localizing both syntactical bugs and analog Trojans in circuit netlists. By employing chain-of-thought reasoning and few-shot examples, SPICED teaches LLMs to identify anomalies without explicit training, resulting in zero area overhead. The framework demonstrates an average Trojan coverage of 93.32% and a true positive rate of 93.4% across various analog benchmark circuits, validating the effectiveness of LLMs in enhancing EDA security and reliability.

AnalogCoder (Lai *et al.*, 2024) is a pioneering training-free Large Language Model agent designed for automating analog circuit design through Python code generation. Unlike previous approaches that require extensive training on specialized datasets, AnalogCoder leverages a feedback-enhanced design flow and a circuit tool library to enable self-correcting and efficient design processes. The agent interprets natural language descriptions of circuit design tasks and generates functionally correct Python code using the PySpice library, significantly reducing the complexity and manual effort involved. Through extensive experiments, AnalogCoder demonstrates superior performance compared to other LLMs, successfully designing 20 out of 24 benchmark circuits.

LADAC (Liu *et al.*, 2024c), a Large Language Model-driven Auto-Designer for Analog Circuits, presents a novel approach to automating analog circuit design by leveraging a domain-specific LLM as a decision-making agent. This work addresses the limitations of current LLMs in comprehending the intricate nature of analog circuits by integrating a specialized knowledge library and interactive tools that enable the LLM to autonomously perform tasks such as transistor sizing and simulation. Through successful designs of amplifiers and a ring oscillator, LADAC demonstrates the promising future of using LLMs for more complex analog circuit design challenges.

# 4

# Conclusion

In this survey, we have explored the transformative impact of Large Language Models (LLMs) on the field of Electronic Design Automation (EDA), highlighting their dual roles as assistants and agents. The benefits of LLMs in terms of productivity, efficiency, and innovation are compelling. By automating routine tasks, providing expert-level insights, and facilitating more intuitive and user-friendly interactions with EDA tools, LLMs are poised to revolutionize the way electronic systems are designed, verified, and manufactured. However, the integration of LLMs into EDA workflows is not without its challenges. Ensuring the accuracy and reliability of LLM-generated content, mitigating the risk of hallucinations, and addressing the computational demands of training and deploying these models are ongoing areas of research.

As the field continues to evolve, it is clear that LLMs will play an increasingly central role in EDA. Future research will likely focus on further customizing and fine-tuning these models for specific EDA applications, improving their interpretability and explainability, and integrating them more seamlessly into existing design environments. The development of open-source datasets and benchmarks will be crucial for advancing the state of the art and ensuring that the benefits of LLMs are accessible to a broader community of designers and researchers. As these models continue to mature and integrate more deeply into EDA workflows, they will undoubtedly drive new innovations and efficiencies, ultimately contributing to the development of more advanced and reliable electronic systems. The journey from LLMs as assistants to agents is well underway, and the future of EDA is set to be profoundly shaped by these powerful AI-driven tools.

# References

Abdelatty, M. and S. Reda. (2024). "HDLCopilot: Hardware Design Library Querying with Natural Language". *arXiv preprint arXiv:2407.12749*.

Chaudhuri, J., D. Thapar, A. Chaudhuri, F. Firouzi, and K. Chakrabarty. (2024). "SPICED: Syntactical Bug and Trojan Pattern Identification in A/MS Circuits using LLM-Enhanced Detection". *arXiv preprint arXiv:2408.16018*.

Chen, G., H. Yang, B. Yu, and H. Ren. (2024a). "Intelligent OPC Engineer Assistant for Semiconductor Manufacturing". *arXiv preprint arXiv:2408.12775*.

Chen, G., K. Zhu, S. Kim, H. Zhu, Y. Lai, B. Yu, and D. Z. Pan. (2024b). "LLM-Enhanced Bayesian Optimization for Efficient Analog Layout Constraint Generation". *arXiv preprint arXiv:2406.05250*.

Fernando, S., R. Kunzelmann, D. S. Lopera, J. Al Halabi, and W. Ecker. (2024). "Leveraging Large Language Models for the Automated Documentation of Hardware Designs". In: *2024 13th Mediterranean Conference on Embedded Computing (MECO)*. IEEE. 1–6.

Hammoud, A., C. Goyal, S. Pathen, A. Dai, A. Li, G. Kielian, and M. Saligane. (2024). "Human Language to Analog Layout Using GLayout Layout Automation Framework". In: *Proceedings of the 2024 ACM/IEEE International Symposium on Machine Learning for CAD*. 1–7.

Han, B., X. Wang, Y. Wang, J. Yan, and Y. Tian. (2023). "New interaction paradigm for complex eda software leveraging gpt". *arXiv preprint arXiv:2307.14740*.

He, Z., H. Wu, X. Zhang, X. Yao, S. Zheng, H. Zheng, and B. Yu. (2023). "ChatEDA: A Large Language Model Powered Autonomous Agent for EDA". In: *2023 ACM/IEEE 5th Workshop on Machine Learning for CAD (MLCAD)*. IEEE. 1–6.

Jiang, Z., Q. Zhang, C. Liu, H. Li, and X. Li. (2024). "IICPilot: An Intelligent Integrated Circuit Backend Design Framework Using Open EDA". *arXiv preprint arXiv:2407.12576*.

Kaintura, A., S. S. Luar, I. I. Almeida, *et al.* (2024). "ORAssistant: A Custom RAG-based Conversational Assistant for OpenROAD". *arXiv preprint arXiv:2410.03845*.

Lai, Y., S. Lee, G. Chen, S. Poddar, M. Hu, D. Z. Pan, and P. Luo. (2024). "AnalogCoder: Analog Circuit Design via Training-Free Code Generation". *arXiv preprint arXiv:2405.14918*.

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

Liu, B., H. Zhang, X. Gao, Z. Kong, X. Tang, Y. Lin, R. Wang, and R. Huang. (2024a). "LayoutCopilot: An LLM-powered Multi-agent Collaborative Framework for Interactive Analog Layout Design". *arXiv preprint arXiv:2406.18873*.

Liu, C., W. Chen, A. Peng, Y. Du, L. Du, and J. Yang. (2024b). "AmpAgent: An LLM-based Multi-Agent System for Multi-stage Amplifier Schematic Design from Literature for Process and Performance Porting". *arXiv preprint arXiv:2409.14739*.

Liu, C., Y. Liu, Y. Du, and L. Du. (2024c). "LADAC: Large Language Model-driven Auto-Designer for Analog Circuits". *Authorea Preprints*.

Pu, Y., Z. He, T. Qiu, H. Wu, and B. Yu. (2024). "Customized Retrieval Augmented Generation and Benchmarking for EDA Tool Documentation QA". *arXiv preprint arXiv:2407.15353*.

Qiu, S., B. Tan, and H. Pearce. (2024). "Explaining EDA synthesis errors with LLMs". *arXiv preprint arXiv:2404.07235*.

Sharma, U., B.-Y. Wu, S. R. D. Kankipati, V. A. Chhabria, and A. Rovinski. (2024). "OpenROAD-Assistant: An Open-Source Large Language Model for Physical Design Tasks". In: *Proceedings of the 2024 ACM/IEEE International Symposium on Machine Learning for CAD*. 1–7.

Shi, L., M. Kazda, B. Sears, N. Shropshire, and R. Puri. (2024). "Ask-EDA: A Design Assistant Empowered by LLM, Hybrid RAG and Abbreviation De-hallucination". *arXiv preprint arXiv:2406.06575*.

Wang, L.-C. (2024). "LLM-Assisted Analytics in Semiconductor Test". In: *Proceedings of the 2024 ACM/IEEE International Symposium on Machine Learning for CAD*. 1–7.

Wang, Z., Y. Shen, X. Yao, W. Zhao, Y. Bai, F. Farnia, and B. Yu. (2024). "ChatPattern: Layout Pattern Customization via Natural Language". *ACM/IEEE Design Automation Conference (DAC)*.

Wu, B.-Y., U. Sharma, S. R. D. Kankipati, A. Yadav, B. K. George, S. R. Guntupalli, A. Rovinski, and V. A. Chhabria. (2024a). "EDA Corpus: A Large Language Model Dataset for Enhanced Interaction with OpenROAD". *arXiv preprint arXiv:2405.06676*.

Wu, H., Z. He, X. Zhang, X. Yao, S. Zheng, H. Zheng, and B. Yu. (2024b). "Chateda: A large language model powered autonomous agent for eda". *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*.

Xu, N., Z. Zhang, L. Qi, W. Wang, C. Zhang, Z. Ren, H. Zhang, X. Cheng, Y. Zhang, Z. Liu, *et al.* (2024). "ChipExpert: The Open-Source Integrated-Circuit-Design-Specific Large Language Model". *arXiv preprint arXiv:2408.00804*.

Yin, Y., Y. Wang, B. Xu, and P. Li. (2024). "ADO-LLM: Analog Design Bayesian Optimization with In-Context Learning of Large Language Models". *arXiv preprint arXiv:2406.18770*.

Zhao, W. X., K. Zhou, J. Li, T. Tang, X. Wang, Y. Hou, Y. Min, B. Zhang, J. Zhang, Z. Dong, *et al.* (2023). "A survey of large language models". *arXiv preprint arXiv:2303.18223*.

# Evaluating Large Language Models for Automatic Register Transfer Logic Generation for Combinational Circuits via High-Level Synthesis

Sneha Swaroopa[1], Rijoy Mukherjee[2], Anushka Debnath[3] and Rajat Subhra Chakraborty[4]

[1]*Indian Institute of Technology Kharagpur, India; swaroopasneha202@kgpian.iitkgp.ac.in*
[2]*Indian Institute of Technology Kharagpur, India; rijoy.mukherjee@iitkgp.ac.in*
[3]*National Institute of Technology Durgapur, India; anushkadebnath77777@gmail.com*
[4]*Indian Institute of Technology Kharagpur, India; rschakraborty@cse.iitkgp.ac.in*

ABSTRACT

The ever-growing popularity of large language models (LLMs) has resulted in their increasing adoption for hardware design and verification. Prior research has attempted to assess the capability of LLMs to automate digital hardware design by producing superior-quality Register Transfer Logic (RTL) descriptions, particularly in Verilog. However, these tests have revealed that Verilog code production using LLMs at current state-of-the-art lack sufficient functional correctness to be practically viable, compared to automatic generation

Sneha Swaroopa, Rijoy Mukherjee, Anushka Debnath and Rajat Subhra Chakraborty (2025), "Evaluating Large Language Models for Automatic Register Transfer Logic Generation for Combinational Circuits via High-Level Synthesis", Foundations and Trends® in Electronic Design Automation: Vol. 14, No. 4, pp 315–337. DOI: 10.1561/1000000063-3.

of programs in general-purpose programming languages such as C, C++, Python, etc. With this as the key insight, in this work we assess the performance of a two-stage software pipeline for automated Verilog RTL generation for combinational circuits: LLM based automatic generation of annotated C++ code suitable for high-level synthesis (HLS), followed by HLS to generate Verilog RTL. We have benchmarked the performance of our proposed scheme using the open-source *VerilogEval* dataset, for four different industry-scale LLMs, and the *Vitis HLS* tool. Our experimental results demonstrate that our two-step technique substantially outperforms previous proposed techniques of direct Verilog RTL generation by LLMs in terms of average functional correctness rates, reaching a score of 0.86 in *pass*@1 metric.

# 1

# Introduction

Digital hardware design using a Hardware Description Language (HDL) like Verilog has traditionally been a labor-intensive and challenging endeavor. It warrants significant expertise, and is error-prone, necessitating intense validation efforts for large industry-scale designs. Historically, digital hardware design using HDLs has been a specialized skill-set, and far lesser number of expert designers using HDLs like Verilog exist than expert programmers in mainstream programming languages such as C, C++, Java, Python, etc.

Naturally, there is an increasing interest in developing more accessible approaches for producing HDL, intending to simplify design and testing processes. This would make constructing functionally correct hardware easier for those with pre-existing software development experience in mainstream programming languages. Towards this end, High-level synthesis (HLS) has emerged and matured over the last few decades, as a paradigm that helps software developers to design correct-by-construction hardware, by directly translating code from a high-level programming language like C or C++ (the two most common options), to a target HDL like Verilog (Cong *et al.*, 2011). For instance, *Vitis HLS* (AMD, n.d.) and *Bambu* (Ferrandi *et al.*, 2021) are free-of-

cost software tools that facilitate rapid hardware logic prototyping by generating RTL in HDL, based on hardware specifications in (supersets of) C, C++, and System C.

Recent efforts have been aimed at elevating the abstraction level and ease of digital hardware design even further, where the goal is to allow designers to articulate functional specifications of the logic to be described in a natural language (NL) (Pearce *et al.*, 2020). State-of-the-art Large Language Models (LLMs) have the potential to revolutionize digital hardware design and validation methodology, by automatically generating correct-by-construction circuit descriptions in HDL, and HDL code for hardware validation ("testbench"), when prompted by the functional description of the circuit in a natural language, e.g. English.

However, recent studies reported by multiple research groups to evaluate the potential of LLMs at the current state-of-the-art in producing functionally correct HDL descriptions (mostly RTL in Verilog) have identified limitations regarding their ability to generate correct and complete Verilog (Liu *et al.*, 2023). Input "prompt engineering" targeting LLMs for automatic Verilog generation also lacks structure and is sometimes ambiguous. Another constraint is that currently, LLMs cannot perform top-down digital design, necessitating (relatively) much more laborious bottom-up design (Liu *et al.*, 2023). LLMs often generate HDL descriptions without ensuring functional correctness at the hardware level and usually do not perform any design space exploration (Chang *et al.*, 2023). The main reason for the low effectiveness of LLMs in HDL generation tasks is the relatively small amount of open-source codebases in HDLs like Verilog and VHDL available for training the LLMs. These limitations currently prevent the seamless integration of LLMs into EDA workflows.

On the other hand, advances in LLMs have led to phenomenal results on automatic code generation tasks (Rozière *et al.*, 2024). This is made possible by the availability of large collections of open-source codebases in common general-purpose programming languages, which can be used for fine-tuned training of the LLMs. LLMs have exhibited unprecedented success in diverse tasks like suggesting code snippets ("copilots"), solving complex algorithms, and explaining programming

concepts. Therefore, combining HLS which allows logic design through higher-level programming interfaces, with the emerging LLMs, represents progress toward the vision of automatic and error-free hardware logic design via natural language interaction.

In this work, we explore the capabilities of LLM in generating sophisticated C++ programs suitable for HLS processing, starting with natural language problem descriptions. We then use a HLS tool to convert this generated C++ code to Verilog RTL. Since the reliability of hardware design is a key factor, we enable the evaluation of the generated final Verilog code through a robust validation procedure (Liu *et al.*, 2024b). Figure 1.1 compares the proposed software pipeline of LLM-assisted automated hardware logic design, with that of the existing works. We experimented with four different well-known industry-standard LLMs (`gpt-3.5 turbo` (OpenAI, 2024a), `gpt-4o` (OpenAI, 2024c), `Claude 3 Haiku` (Anthropic, 2024a), `Claude 3.5 Sonnet` (Anthropic, 2024b)), and *Vitis HLS*.

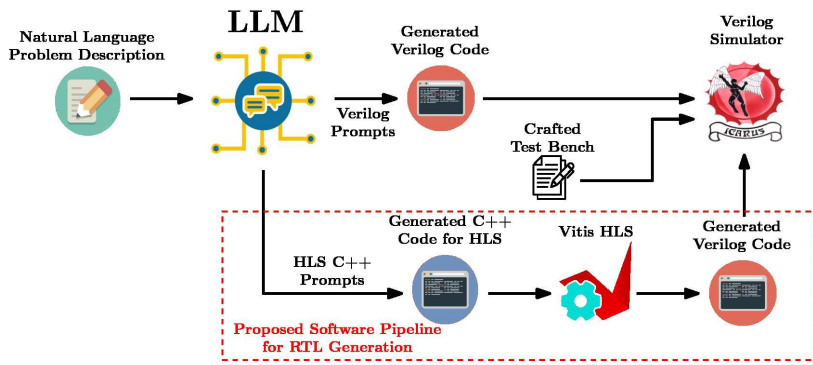


**Figure 1.1:** Comparison of existing RTL Generation via LLM with the proposed software pipeline with HLS.

We use a subset of the *VerilogEval* dataset (Liu *et al.*, 2023), comprising 70 problems sourced from the *HDLBits* platform, to evaluate the approaches (the proposed software pipeline vs. existing techniques). These problems constitute a wide variety of combinational circuits and are versatile enough to serve as the foundation for creating larger, more complex systems. Consequently, our chosen HDL dataset is smaller in scale compared to those of higher-level programming languages, the 70

combinational circuit problems in this dataset are both adequate and representative for evaluating the diverse challenges of combinational circuit design in HDL. We call this the **HLSEval** dataset, which establishes the superior accuracy (up to *pass*@1 value of 0.86) for our proposed approach over existing LLM-assisted automatic Verilog RTL generation. We have made the **HLSEval** benchmarks and LLM scripts open-source on Github (Mukherjee, 2024).

In the next section, we describe some related work on automated RTL generation and validation using LLMs.

# 2

# LLM-based Automated Digital Hardware Design and Validation

With the escalating popularity of LLMs, several research works have aimed at harnessing their power to generate HDL (primarily Verilog) code, to automate digital hardware design and validation. Both Dehaerne *et al.* (2023) and Thakur *et al.* (2024a) fine-tuned the open-source `CodeGen` model (Nijkamp *et al.*, 2023) using training data that consists of Verilog code from GitHub and machine-readable Verilog code snippets from electronic copies of Verilog textbooks. Evaluation of curated tasks concluded that the resulting `VeriGen` model (Thakur *et al.*, 2024a) delivered functionally accurate code only 41.9% of the time. In Liu *et al.* (2023), the authors designed a benchmarking framework to assess LLM performance in Verilog code generation tasks in hardware design. They also introduced a robust, comprehensive dataset called *VerilogEval*, comprising 156 problems sourced from HDLBits (Wong, 2024), an online Verilog learning platform. The evaluation demonstrated the Verilog code generation capabilities of pre-trained language models like `gpt-3.5` (OpenAI, 2024a) and `gpt-4` (OpenAI, 2024b), and the fine-tuned `VeriGen` model, which incidentally achieved the best performance. However, the improvement wasn't much from the numbers achieved previously. To improve the quality of Verilog generation by pre-trained

models, Thakur *et al.* (2024b) developed a workflow with a chain of feedback mechanisms called *AutoChip* to prompt the LLMs for iterative hardware development. Chang *et al.* (2024) introduced `ChipGPT-FT`, a Verilog design model, by fine-tuning the `Llama-2` (Touvron *et al.*, 2023) model with the Verilog dataset augmented from GitHub. When compared with performance of `VeriGen` (Thakur *et al.*, 2024a) on similar benchmarks, `ChipGPT-FT` increases functional correctness of the generated code from 58.8% to 70.6%.

Another branch of work targeting LLMs has focused on the quality of the generated Verilog code regarding creativity, optimization, resource utilization and flexibility (Chang *et al.*, 2023; DeLorenzo *et al.*, 2024a; DeLorenzo *et al.*, 2024b; Blocklove *et al.*, 2023). Other prominent works in this domain have focused on using LLMs for hardware testing (Qiu *et al.*, 2024; Blocklove *et al.*, 2024), bug fixing (Ahmad *et al.*, 2024), generating security assertions (Paria *et al.*, 2023; Kande *et al.*, 2024; Meng *et al.*, 2023) and formal verification of RTL (Orenes-Vera *et al.*, 2023). Works such as Wu *et al.* (2024) and Liu *et al.* (2024a) use LLM to enhance hardware design productivity by providing a conversational interface like an engineering chatbot that helps in task planning, EDA script generation, bug analysis and task execution through the invocation of EDA tool APIs.

Very few studies have been conducted in the LLM realm regarding high-level synthesis. Collini *et al.* (2024) proposed a workflow called *C2HLSC*, enabling seamless refactoring of a traditional C program to HLS-compatible C, to produce optimized hardware architectures. Fu *et al.* (2023) developed a framework called *GPT4AIGChip*, to design an AI accelerator, starting from human language specifications. Specifically, it used `gpt-4` (OpenAI, 2024b) to design the AI accelerator by prompting for decoupled HLS C++ modules. Though the authors investigated the viability of HLS with the conclusion that current LLMs struggle with understanding long dependencies, the experimentation was not thorough and comprehensive enough to account for various types of tasks frequently seen for hardware design in HLS.

# 3

# Proposed Software Pipeline for Automated Verilog Generation and its Validation

In this section, we discuss in detail the *HLSEval* dataset; our proposed software pipeline for automated Verilog generation, and its evaluation technique.

## 3.1 The HLSEval Dataset

We assess the functional correctness of the proposed hardware design pipeline on problems selected from the **VerilogEval** (Liu *et al.*, 2023) dataset. *VerilogEval* encompasses diverse coding tasks, ranging from simple combinational circuits to complex finite state machines. Our proposed **HLSEval** dataset consists of concise hardware design tasks that demand problem-solving skills in circuit optimization, boolean logic reduction, state transitions, and more. We only consider a subset of 70 combinational logic circuits for this evaluation process. Sequential circuits are realized in HLS in the form of *streams* (AMD, n.d.), whereas in RTL, they are realized with one or more user-defined clock signal(s) controlling a module. Thus, a fair comparison between our proposed two-step approach with that of direct LLM-generated Verilog is difficult due to contrasting underlying concepts (Cong *et al.*, 2022; AMD, n.d.). So, we did not include those 62 sequential problems. Remaining problems in

*VerilogEval* were related to bug-finding in Verilog codes and text-based question answers, and are hence unrelated to our scope.

For **HLSEval**, the problem descriptions from the **VerilogEval** are converted into a schema suitable for HLS C++ prompting. A sample for problem statement `xnor` is shown in Figure 3.1. It has two parts: 1) the problem description representing the hardware generation task in simple English, and, 2) the function description providing a snapshot of the function signature we are expecting from the LLM. Representing Finite State Machines (FSMs) and Karnaugh maps (K-maps) into textual forms was a challenge, and for this, we queried `gpt-3.5 turbo` (OpenAI, 2024a) as shown in Figure 3.2 and Figure 3.3. We adopted the textual structure as returned by `gpt-3.5 turbo`, for further prompting by LLM.

**User Prompt:**

```
########problem description:
Create a module that implements an XNOR gate.

########function description:
#include <ap_int.h>
void xnorgate(ap_uint<1>a, ap_uint<1> b, ap_uint<1>& out){
    #pragma HLS INTERFACE ap_none port=out
    #pragma HLS INTERFACE ap_ctrl_none port=return
    <FILL>
}

########solution:
```

**Figure 3.1:** Problem desciption of `xnor` in **HLSEval**.

## 3.2 Proposed Software Pipeline for Automated Verilog Generation

The first stage of the proposed pipeline, as shown in Figure 1.1, consists of prompting the LLM properly with well-crafted problem statements in a natural language (English in our case). Figure 3.4 shows an example of such prompting for the problem `fadd` (a binary full adder). The system prompt is concatenated with an "one-shot" example and user prompt, and sent to the LLM for inference. The one-shot example helps to address ambiguity within the problem descriptions, particularly when there is a precise requirement for certain attributes in the LLM response.

**User Prompt:**
I want you to write code from a state transition diagram. How can I give you this diagram in a text format?

**LLM Response:**
Here's an example format you can follow:
States:
S0
S1
S2
Inputs:
IN1
IN2
Transitions:
S0: (IN1) -> S1
S0: (IN2) -> S2
S1: (IN1) -> S0
S1: (IN2) -> S2
S2: (IN1) -> S2
S2: (IN2) -> S0
Outputs:
S0: OUT0
S1: OUT1
S2: OUT2

**Figure 3.2:** `gpt-3.5 turbo` response for Finite State Machine (FSM) specifications.

**User Prompt:**
I want you to write code from a Karnaugh map. How can I give you this diagram in a text format?

**LLM Response:**
Here's an example format for a 4-variable K-map:

Variables: A, B, C, D
K-map:
AB\CD | 00 | 01 | 11 | 10
---
00 | 0 | 1 | 1 | 0
01 | 1 | 1 | 0 | 0
11 | 0 | 0 | 1 | 1
10 | 1 | 0 | 0 | 1

**Figure 3.3:** `gpt-3.5 turbo` response for Karnaugh map (K-map) specification.

The pragmas introduced for each problem are optional. In our case, we use it to align the downstream Verilog generated with those generated by the existing pipeline of Liu *et al.* (2023) for better comparison. Figure 3.4 also shows the response produced by LLM, an annotated HLS-compliant C++ code snippet with the provided function signature, and correct functionality. Next, the generated HLS C++ code produced by LLM is converted to Verilog using *Vitis HLS* (AMD, n.d.). Figure 3.5 shows the generated Verilog for `fadd` by *Vitis HLS*, corresponding to the C++ code in Figure 3.4. In the full adder circuit description, the

**System Prompt:**
Your task is to complete <FILL> portion in the given high-level synthesis (HLS) C++ program. The problem description and the function description is provided. Do not include any additional HLS pragma. A reference example is provided which is delimited by triple backticks. Do not output any unnecessary statements and no need for explanations. Do not forget to add #include <ap_int.h>.

**One-shot Example:**

```
########problem description:
Create a 2-1 multiplexer. When sel=0, choose a. When sel=1, choose b.
########function description:
#include <ap_int.h>
void mux2to1v(ap_uint<100> a, ap_uint<100> b, ap_uint<1> sel,
              ap_uint<100>& out){
    #pragma HLS INTERFACE ap_none port=out
    #pragma HLS INTERFACE ap_ctrl_none port=return
    <FILL>
}
########solution:
#include <ap_int.h>
void mux2to1v(ap_uint<100> a, ap_uint<100> b, ap_uint<1> sel,
              ap_uint<100>& out){
    #pragma HLS INTERFACE ap_none port=out
    #pragma HLS INTERFACE ap_ctrl_none port=return
    out = sel ? b : a;
}
```

**User Prompt:**

```
########problem description:
Create a full adder. A full adder adds three bits (including carry-in) and produces a
sum and carry-out.
########function description:
#include <ap_int.h>
void fadd (ap_uint<1> a, ap_uint<1> b, ap_uint<1> cin, ap_uint<1>& cout,
           ap_uint<1>& sum){
    #pragma HLS INTERFACE ap_none port=cout
    #pragma HLS INTERFACE ap_none port=sum
    #pragma HLS INTERFACE ap_ctrl_none port=return
    <FILL>
}
########solution:
```

**LLM Response:**

```
#include <ap_int.h>
void fadd (ap_uint<1> a, ap_uint<1> b, ap_uint<1> cin, ap_uint<1>& cout,
           ap_uint<1>& sum){
    #pragma HLS INTERFACE ap_none port=cout
    #pragma HLS INTERFACE ap_none port=sum
    #pragma HLS INTERFACE ap_ctrl_none port=return
    sum = a ^ b ^ cin;
    cout = (a & b) | (b & cin) | (a & cin);
}
```

**Figure 3.4:** Example of prompting for `fadd` (full adder) in **HLSEval**. The logic description includes a description of the problem in natural language, a function description, and a sample one-shot input and output definition.

inputs are $a$, $b$ and $cin$, and we need to evaluate the outputs $cout$ and $sum$. These are expressed as logic functions of two intermediate signals (for $cout$), and one intermediate signal and a primary input (for $sum$).

For a different circuit, the same system prompt and One-shot-example as mentioned previously can be used. The user would just need

**Generated Verilog:**

```
module fadd (
      a,
      b,
      cin,
      cout,
      sum
);
input  [0:0] a;
input  [0:0] b;
input  [0:0] cin;
output  [0:0] cout;
output  [0:0] sum;
wire   [0:0] xor_ln1545_fu_66_p2;
wire   [0:0] or_ln1544_fu_79_p2;
wire   [0:0] ret_V_4_fu_85_p2;
wire   [0:0] ret_V_5_fu_91_p2;
assign cout = (ret_V_5_fu_91_p2 | ret_V_4_fu_85_p2);
assign or_ln1544_fu_79_p2 = (cin | a);
assign ret_V_4_fu_85_p2 = (or_ln1544_fu_79_p2 & b);
assign ret_V_5_fu_91_p2 = (cin & a);
assign sum = (xor_ln1545_fu_66_p2 ^ a);
assign xor_ln1545_fu_66_p2 = (cin ^ b);
endmodule //fadd
```

**Figure 3.5:** Verilog RTL of `fadd` generated by *Vitis HLS*.

to provide a new user prompt with a problem description and functional description based on the new problem.

## 3.3 Evaluation Methodology for Proposed Technique

We use the open-source evaluation harness described in Liu *et al.* (2024b) to validate the Verilog files generated by our pipeline. It compares simulation results between the generated Verilog, and "golden reference" Verilog implementations. It uses the open-source *Icarus Verilog* (Williams, 2024) simulator adapted in a sandbox environment to safely run the generated Verilog files. We use the popular *pass@k* metric (Chen *et al.*, 2021) to measure functional correctness of the generated Verilog RTL, where a problem is considered solved if any one of $k$ randomly selected samples pass the unit tests:

$$pass@k := \mathbb{E}_{\text{Problems}} \left[ 1 - \frac{\binom{n-c}{k}}{\binom{n}{k}} \right], \tag{3.1}$$

where we generate $n \geq k$ samples per problem among which $c \leq n$ samples pass testing. In practice, the number of samples $n$ must be sufficiently large (at least twice the value of $k$) to produce low-variance estimates for $pass@k$.

In the next section, we describe comparative experimental evaluation results.

# 4

# Experimental Results

For testing, we used four of the best state-of-the-art commercial LLMs, namely, `gpt-3.5 turbo` (OpenAI, 2024a) and `gpt-4o` (OpenAI, 2024c) from *OpenAI*, and `Claude 3 Haiku` (Anthropic, 2024a) and `Claude 3.5 Sonnet` (Anthropic, 2024b) by *Anthropic* (the last two were hosted on Amazon Web Services). We set the parameters to be: *top* $p = 0.95$, temperature $= 0.8$, and context length $= 2048$ for all the experiments. For measuring $pass@k = \{1, 5, 10\}$ we use (3.1), with $n = 20$ code completions for each problem. We used *Xilinx* (*AMD*) *Vitis HLS* for high level synthesis, and *Xilinx* (*AMD*) *Vivado* for synthesis of the generated RTL designs targeting the *Xilinx Artix-7* FPGA. The entire flow was automated to a "push-button" mode using custom Python scripts (for the LLM query and associated steps) and TCL scripts (for the HLS and logic-synthesis steps). Given the ease of automating the flow, the proposed methodology is amenable to being integrated with existing industry-standard VLSI flows.

## 4.1 English to Verilog RTL: Direct Conversion vs. Two-stage Proposed Software Pipeline

Table 4.1 illustrates the *pass*@ rates for the **HLSEval** tasks using the various LLMs, and compares both techniques. **NL to Verilog** refers to existing technique (Liu *et al.*, 2023), where LLM generates Verilog based on natural language, whereas **NL to Verilog via HLS** refers to our proposed approach where LLM generates HLS-compatible C++ based on natural language, that is then converted to Verilog. The terminology *pass*@1 refers to the probability that one sample is generated, and it is found to be correct – a limiting case for the *pass*@$k$ metric with $k = 1$. It is computed by evaluating the expected fraction of problems where the randomly chosen sample passes all unit tests, reflecting the functional correctness of the generated code when only one sample is evaluated per problem. Similarly, *pass*@5 and *pass*@10 measure the probabilities that at least one of five or ten randomly selected samples, respectively, passes all unit tests for a given problem.

These metrics reflect the functional correctness of the generated code when evaluating up to five or ten samples per problem. From the study in Liu *et al.* (2023), more capable and larger LLMs generally result in better Verilog code generation capability. This observation is consistent with our results for the four LLMs, both for the **NL to Verilog** scheme, as well as the **NL to Verilog via HLS** scheme. However, **we obtained significant improvement in *pass*@ values** for the proposed **NL to Verilog via HLS** software pipeline, compared to the **NL to Verilog** technique.

Table 4.2 shows the average percentage overheads incurred (post-synthesis and routing) by implementations obtained by synthesizing Verilog RTL generated following our proposed methodology, over the "golden reference" designs of the *VerilogEval* dataset (Liu *et al.*, 2023). From these results, it is clear that overheads for most of the designs are within acceptable limits, and even better (i.e. negative overhead) for our proposed methodology. Notably, average power dissipation overhead was negative for all the LLM models used by us.

**Table 4.1:** Comparison of different models on **HLSEval** dataset (average *pass*@*k* values)

| Model | NL to Verilog | | | NL to Verilog via HLS (This Work) | | |
|---|---|---|---|---|---|---|
| | *pass*@1 | *pass*@5 | *pass*@10 | *pass*@1 | *pass*@5 | *pass*@10 |
| `gpt-3.5 turbo` | 0.36 | 0.53 | 0.56 | 0.60 | 0.66 | 0.68 |
| `gpt-4o` | 0.67 | 0.75 | 0.77 | 0.80 | 0.83 | 0.84 |
| `Claude 3 Haiku` | 0.50 | 0.63 | 0.66 | 0.64 | 0.72 | 0.74 |
| `Claude 3.5 Sonnet` | 0.70 | 0.80 | 0.83 | **0.86** | **0.87** | **0.87** |

**Table 4.2:** Comparison of different models on **HLSEval** dataset (average overhead values, functionally correct Verilog only)

| Model | Average Overhead (%) | | |
|---|---|---|---|
| | Hardware (# LUTs) | Power | Delay |
| `gpt-3.5 turbo` | 1.55 | -0.30 | 0.20 |
| `gpt-4o` | 0.80 | -0.45 | -0.05 |
| `Claude 3 Haiku` | 1.22 | -0.15 | 0.15 |
| `Claude 3.5 Sonnet` | 0.85 | -0.10 | -0.10 |

## 4.2 Current Limitations and Discussions

In automated logic design from natural language, we found that realizing circuits corresponding to K-map problems, such as those in Figure 4.1, is a key challenge. Even though we prompt the LLM with the schema provided in Figure 3.3, none of the four LLMs generated correct code. By analyzing the generated (incorrect) Verilog RTL codes, we found that the LLMs could not minimize the K-maps, or correctly identify the minterms or maxterms. This could be a future work domain for LLMs to solve, by fine-tuning with more data or advanced prompting techniques. However, in our opinion, the inability of our methodology to handle Karnaugh maps (K-maps) is not a major shortcoming of our technique, since K-maps are usually difficult to employ to minimize logic functions with more than four variables. This leaves K-maps to be useful instructional tools, but little more than that. Also, equivalent textual descriptions (e.g. functional descriptions or truth tables) exist as alternatives for K-maps, which can be used in our methodology.

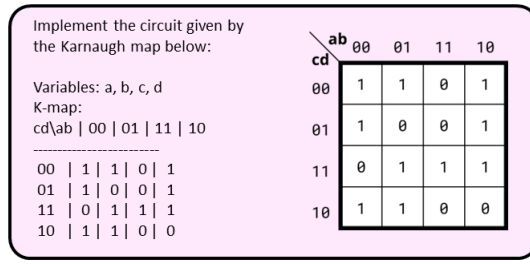


**Figure 4.1:** An example Karnaugh Map (K-map) problem.

The scope of the current work is relatively limited because of its focus on combinational circuits. Sequential circuits come in different variants (e.g. controllers and finite state machines, instruction execution pipelines and pipelined datapaths, registers and clocked buffers, etc.). In particular, pipelined datapath circuit hardware architectures can vary widely depending on whether the focus is on throughput or on hardware resource footprint. Given a C++ code segment to describe an algorithm at a substantially higher abstraction level, its equivalent sequential circuit implementation during HLS requires substantial design-space exploration, based on input user directives and constraints. In our proposed methodology, we found it challenging to input these hardware architectural specifications while querying the LLM to generate C++, because of the gap in the level of abstraction. Our current research is focused on overcoming this challenge.

Our usage of ready-made pre-trained LLM without any fine-tuning may seem to be a disadvantage. However, fine-tuning requires an open-source LLM, availability of sufficient computational resources, and the expense of substantial computational effort. Unfortunately, currently we do not have access to such extensive computational resources, which prevented us from experimenting with fine-tuning an open-source LLM model. Also, given that even current pre-trained LLMs demonstrate extremely good capabilities for automatic C++ code generation (the main observation on which the current work is based), we envisage little improvement over the results that we have obtained.

# 5

# Conclusions

Researchers have applied LLMs to generate RTL descriptions for digital hardware from hardware specifications written in natural language. This work proposes an alternative as a software pipeline for automatic hardware logic design, where LLMs are prompted to produce HLS-compatible C++ code, followed by HLS of the generated C++ code to Verilog RTL. Detailed experiments on a standard benchmark suite demonstrated that Verilog code generation through the proposed software pipeline achieves much higher functional correctness rate, at small resource, delay and power overheads. Our future research efforts will be directed towards exploring the ability of the proposed methodology for sequential circuits, and to improve security, power efficiency, thermal management and other important aspects of VLSI system design.

# References

Ahmad, B., S. Thakur, B. Tan, R. Karri, and H. Pearce. (2024). "On Hardware Security Bug Code Fixes by Prompting Large Language Models". *IEEE Transactions on Information Forensics and Security*. 19: 4043–4057. DOI: 10.1109/TIFS.2024.3374558.

AMD. "Vitis HLS". URL: https://docs.amd.com/r/en-US/ug1399-vitis-hls.

Anthropic. (2024a). "Claude 3 Haiku". URL: https://www.anthropic.com/news/claude-3-haiku.

Anthropic. (2024b). "Claude 3.5 Sonnet". URL: https://www.anthropic.com/news/claude-3-5-sonnet.

Blocklove, J. *et al.* (2023). "Chip-Chat: Challenges and Opportunities in Conversational Hardware Design". In: *2023 ACM/IEEE 5th Workshop on Machine Learning for CAD (MLCAD)*. IEEE.

Blocklove, J. *et al.* (2024). "Evaluating LLMs for Hardware Design and Test". URL: https://arxiv.org/abs/2405.02326.

Chang, K. *et al.* (2023). "ChipGPT: How far are we from natural language hardware design". URL: https://arxiv.org/abs/2305.14019.

Chang, K. *et al.* (2024). "Data is all you need: Finetuning LLMs for Chip Design via an Automated design-data augmentation framework". URL: https://arxiv.org/abs/2403.11202.

Chen, M. *et al.* (2021). "Evaluating Large Language Models Trained on Code". URL: https://arxiv.org/abs/2107.03374.

Collini, L., S. Garg, and R. Karri. (2024). "C2HLSC: Can LLMs Bridge the Software-to-Hardware Design Gap?" URL: https://arxiv.org/abs/2406.09233.

Cong, J. *et al.* (2011). "High-Level Synthesis for FPGAs: From Prototyping to Deployment". *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*. 30(4): 473–491.

Cong, J. *et al.* (2022). "FPGA HLS Today: Successes, Challenges, and Opportunities". *ACM Transactions on Reconfigurable Technology and Systems*. 15(4).

Dehaerne, E. *et al.* (2023). "A Deep Learning Framework for Verilog Autocompletion Towards Design and Verification Automation". URL: https://arxiv.org/abs/2304.13840.

DeLorenzo, M. *et al.* (2024a). "Make Every Move Count: LLM-based High-Quality RTL Code Generation Using MCTS". URL: https://arxiv.org/abs/2402.03289.

DeLorenzo, M., V. Gohil, and J. Rajendran. (2024b). "CreativEval: Evaluating Creativity of LLM-Based Hardware Code Generation". URL: https://arxiv.org/abs/2404.08806.

Ferrandi, F. *et al.* (2021). "Bambu: an Open-Source Research Framework for the High-Level Synthesis of Complex Applications". In: *58th ACM/IEEE Design Automation Conference (DAC)*. IEEE. 1327–1330.

Fu, Y. *et al.* (2023). "GPT4AIGChip: Towards Next-Generation AI Accelerator Design Automation via Large Language Models". In: *IEEE/ACM International Conference on Computer Aided Design (ICCAD)*. 1–9.

Kande, R. *et al.* (2024). "(Security) Assertions by Large Language Models". *IEEE Transactions on Information Forensics and Security*. 19: 4374–4389. DOI: 10.1109/TIFS.2024.3372809.

Liu, M. *et al.* (2023). "VerilogEval: Evaluating Large Language Models for Verilog Code Generation". In: *2023 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*.

Liu, M. *et al.* (2024a). "ChipNeMo: Domain-Adapted LLMs for Chip Design". URL: https://arxiv.org/abs/2311.00176.

Liu, M. *et al.* (2024b). "VerilogEval". URL: https://github.com/NVlabs/verilog-eval.

Meng, X. *et al.* (2023). "Unlocking Hardware Security Assurance: The Potential of LLMs". URL: https://arxiv.org/abs/2308.11042.

Mukherjee, R. (2024). "HLSEval". URL: https://github.com/rijoym/HLSEval.

Nijkamp, E. *et al.* (2023). "CodeGen: An Open Large Language Model for Code with Multi-Turn Program Synthesis". URL: https://arxiv.org/abs/2203.13474.

OpenAI. (2024a). "gpt-3.5". URL: https://openai.com/index/chatgpt/.

OpenAI. (2024b). "gpt-4". URL: https://openai.com/index/gpt-4-research/.

OpenAI. (2024c). "gpt-4o". URL: https://openai.com/index/hello-gpt-4o/.

Orenes-Vera, M., M. Martonosi, and D. Wentzlaff. (2023). "Using LLMs to Facilitate Formal Verification of RTL". URL: https://arxiv.org/abs/2309.09437.

Paria, S., A. Dasgupta, and S. Bhunia. (2023). "DIVAS: An LLM-based End-to-End Framework for SoC Security Analysis and Policy-based Protection". URL: https://arxiv.org/abs/2308.06932.

Pearce, H., B. Tan, and R. Karri. (2020). "DAVE: Deriving Automatically Verilog from English". In: *Proceedings of the 2020 ACM/IEEE Workshop on Machine Learning for CAD. MLCAD '20.* Virtual Event, Iceland: Association for Computing Machinery. 27–32.

Qiu, R. *et al.* (2024). "AutoBench: Automatic Testbench Generation and Evaluation Using LLMs for HDL Design". URL: https://arxiv.org/abs/2407.03891.

Rozière, B. *et al.* (2024). "Code Llama: Open Foundation Models for Code". URL: https://arxiv.org/abs/2308.12950.

Thakur, S. *et al.* (2024a). "VeriGen: A Large Language Model for Verilog Code Generation". *ACM Trans. Des. Autom. Electron. Syst.* 29(3).

Thakur, S., J. Blocklove, H. Pearce, B. Tan, S. Garg, and R. Karri. (2024b). "AutoChip: Automating HDL Generation Using LLM Feedback". URL: https://arxiv.org/abs/2311.04887.

Touvron, H. *et al.* (2023). "Llama 2: Open Foundation and Fine-Tuned Chat Models". URL: https://arxiv.org/abs/2307.09288.

Williams, S. (2024). "The ICARUS Verilog Compilation System". URL: https://github.com/steveicarus/iverilog.

Wong, H. (2024). "HDLBits". URL: https://hdlbits.01xz.net/wiki/Problem_sets.

Wu, H. *et al.* (2024). "ChatEDA: A Large Language Model Powered Autonomous Agent for EDA". *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*: 1–1. DOI: 10.1109/TCAD.2024.3383347.

# Editor Biographies

**Imed Ben Dhaou** (S'97-M'02, SM'2011) received his Ph.D. from the Royal Institute of Technology, Sweden, and is currently a Full Professor of Embedded Systems for IoT at the Department of Computer Science, Dar Al-Hekma University, and a Docent at the Department of Computing, University of Turku, Finland. He has authored over 120 publications, including journal articles, conference papers, book chapters, and technical reports, with research interests in embedded systems, IoT, and related technologies. Recognized among the top 2% of scientists worldwide by Stanford and Elsevier, he has received several awards, such as the Best Paper Award at the 1997 Finnish Symposium on Signal Processing, a DAC travel grant (2000), a Publication Award from Qassim University, the Dr. Hussein Mohammed Al-Sayyed Award for Research, and the Community Service Award from Dar Al-Hekma University. A member of Sigma Xi, he has served as an editor for Microelectronics Journal (Elsevier) since 2014 and as a guest editor for ISI journals, including Electronics, Journal of Cloud Computing, Analogue Integrated Circuits and Signal Processing, and Microprocessors and Microsystems, alongside chairing technical committees at various conferences. He is also the editor of the upcoming book IoT-Enabled DC Microgrids: Architecture, Algorithms, Applications, and Technologies, to be published by CRC Press in 2025.

**Hannu Tenhunen**, Prof.Em. Dr.h.c. Prof.h.c., initiated IC and system-on-chip activities at Tampere University in the late 1980s and early 1990s. He then moved to KTH, Sweden, as a chair professor of electronic system design, focusing on electronic system design and Network-on-Chip methodologies. He has successfully launched international master programs in Tampere (with a focus on DSP and DSP-ASIC) and at KTH (with a focus on System-on-Chip). Additionally, he established four double degree programs with elite Chinese universities, approved by the Chinese Ministry of Education. As the director of the Turku Center of Computer Science, he led a doctoral school with around 110 PhD students. He has also served as the European-level education director for the European Institute of Innovation and Technology (EIT) Digital and as an invited part-time professor of smart electronics at the University of Turku. A member of the Finnish Academy of Engineering Sciences, he is also a part-time distinguished professor at Fudan University, focusing on neuromorphic computing and innovative engineering education, particularly with a Global South perspective. He has published over 990 journal and conference papers, with more than 20,000 citations, and has a Google Scholar H-index of 62. His current research interests include AI, neural network accelerators, and neuromorphic computing.

**Ahmed Abdelgawad** received his M.S. and a Ph.D. degree in Computer Engineering from the University of Louisiana at Lafayette in 2007 and 2011, and subsequently joined IBM as a Design Aids & Automation Engineering Professional at the Semiconductor Research and Development Center. In Fall 2012 he joined Central Michigan University as a Computer Engineering Assistant Professor. In Fall 2022, Dr. Abdelgawad was promoted to the rank of Professor. He is a senior member of IEEE. His areas of expertise are Internet of Things (IoT), distributed computing for Wireless Sensor Network (WSN), Structural Health Monitoring (SHM), data fusion techniques for WSN, low power embedded system, digital signal processing, Robotics, RFID, Localization, VLSI, and FPGA design. He has published two books and more than 130 articles in related journals and conferences. Dr. Abdelgawad served as a reviewer for several journals and conferences, including the IEEE IoT

Journal, IEEE Communications Magazine, IEEE Transactions on VLSI, and IEEE Transactions on I&M, Springer, Elsevier, IEEE WF-IoT, IEEE ISCAS, IEEE SAS, and IEEE MWSCAS. Dr. Abdelgawad served as the general chair of the IEEE International Conference on Artificial Intelligence, Blockchain, and Internet of Things, (AIBThings2023), the 3rd IEEE International Conference on Computing and Machine Intelligence (ICMI2024), and the International Conference on Intelligent Systems, Blockchain, and Communication Technologies (ISBCom2024). He served in the organizing committees of IEEE WF-IoT, IEEE ISCAS, IEEE ICIP, IEEE SiPS, IEEE MWSCAS, and GIoTS. In addition, he taught many short IoT courses in different countries. He was the keynote speaker for many international conferences and conducted many webinars. He is currently the IEEE Northeast Michigan section chair and IEEE SPS Internet of Things (IoT) SIG Member. In the last few years, Dr. Abdelgawad was listed in the world's top 2% of scientists by Stanford University, USA. In addition, Dr. Abdelgawad served as a PI and Co-PI for several funded grants from the NSF.

**Sree Ranjani Rajendran** is currently working as an Assistant Professor in the Department of Electrical Engineering and Computer Science at Florida Atlantic University, Boca Raton, Florida, US. Previously she was a postdoctoral associate with the Florida Institute for Cybersecurity Research (FICS Research), in the Department of Electrical and Computer Engineering, University of Florida, Gainesville. She received her Ph.D. in Electronics and Communication Engineering from Amrita Vishwa Vidyapeetham, India. She also worked as a postdoctoral fellow in the Computer Science Department at the Indian Institute of Technology Madras. She is a passionate and curious researcher pursuing knowledge and expertise in the broader domain of Hardware Security, with a specific interest in SoC Verification and developing CAD frameworks for security (design-for-security). She has 10+ years of professional experience in both research and teaching at universities. She authored 40 research articles published in refereed conference proceedings, renowned journal publications, and book chapters. Her research interests include hardware security verification and validation of System-on-Chips. Her research has been published in premier ACM/IEEE journals and confer-

ences, including IEEE Transactions on Emerging Topics in Computing, Transactions on Information Forensics & Security (TIFS), Journal of Cryptographic Engineering, ACM Workshop on Attacks and Solutions in Hardware Security, International Conference on VLSI Design & The International Conference on Embedded Design, and Design Automation and Test in Europe (DATE). She is an IEEE and The Test Technology Technical Community (TTTC) member. Her research has been awarded in the IEEE Young Women Research Grant Award, the 28th IEEE Asian Test Symposium (ATS), 2019, and the Best Technical Research Award in the Hardware.io Poster Competition, 2020. In addition, her work has been nominated for the Best Paper poster award at the GOMATECH 2023 conference.

**Rajat Subhra Chakraborty** is currently Associate Dean of the Faculty of Engineering and Architecture at IIT Kharagpur, and Professor at the Computer Science and Engineering Department of IIT Kharagpur. He received his Ph.D. from Case Western Reserve University (U.S.A.) and B.E. from Jadavpur University. He has professional experience of working at Intel (Bangalore, India), National Semiconductor (Bangalore, India) and Advanced Micro Devices (AMD) (Santa Clara, USA). His research interests include Hardware Security, VLSI Design and Design Automation, Digital Content Protection and Digital Image Forensics. He holds 2 Granted U.S. patents, 3 granted Indian patents, and has co-authored 6 books, 10 book chapters, and over 140 publications in international journals and conferences. His work has received over 7500 citations to date and has won 2 Best Paper awards. He has received several prestigious national and international awards such as IIT Kharagpur Outstanding Faculty Award (2018), IEI Young Engineers Award (2016), IBM Shared University Research (SUR) Award (2015), Royal Academy of Engineering (U.K.) RECI Fellowship (2014) and IBM Faculty Award (2012). He is currently an Associate Editor of IEEE TCAD journal and has previously been an Associate Editor of IEEE TMSCS journal. Prof. Chakraborty is a Senior Member of IEEE and a Senior Member of ACM. He currently holds the position of Vice-chair (2024), and has previously been the Secretary (2023), Treasurer (2022) and Assistant Secretary (2021), of the IEEE Kharagpur Section (R10).