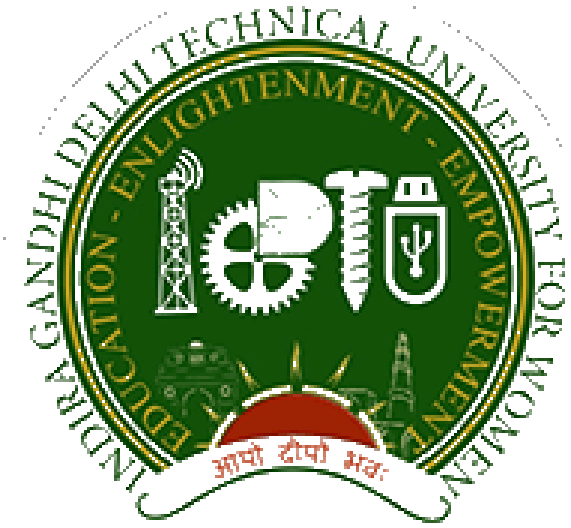
INDIRA GANDHI DELHI TECHNICAL UNIVERSITY FOR WOMEN

-BACHELOR OF TECHNOLOGY-

Electronics and Communication Engineering (ECE)



AI-Driven Circuit Design: Integrating Generative AI with VLSI for Optimized CMOS Architectures

CAPSTONE PROJECT REPORT

Summer Internship Program 2025

Submitted By -

Jiya Yadav (01901022022)

Juhi Sharma (02101022022)

Kajal Goswami (02201022022)

Faculty Advisor: Prof. A.K. Mohapatra

Date of Submission: 15 July 2025

INDEX

ACKNOWLEDGEMENT

We would like to express our heartfelt gratitude to our faculty guide and the department for providing us with the opportunity to work on this project, titled
"AI-Driven Circuit Design: Integrating Generative AI with VLSI for Optimized CMOS Architectures"—a fusion of Generative AI and Prompt Engineering with core electronics.

As electronics students, combining both domains was a unique and enriching experience. This project not only deepened our understanding of circuit design but also introduced us to the innovative potential of AI-driven automation in hardware systems.

We are sincerely thankful to our classmates, friends, and everyone who supported and encouraged us throughout this journey. A special word of thanks goes to the developers of open-source tools and libraries that played a vital role in making this project possible.

CERTIFICATE

This is to certify that the project report entitled:

“AI-Driven Circuit Design: Integrating Generative AI with VLSI for Optimized CMOS Architectures”

Submitted by: Jiya Yadav, Juhi Sharma and Kajal Goswami, in partial fulfillment of the requirements for the Summer Internship Program.

It is a record of original work carried out under our supervision during the 6 weeks summer internship program on Generative AI and Prompt Engineering. This project successfully explores the integration of Generative AI and Prompt Engineering techniques in the design and simulation of CMOS logic circuits, combining AI innovation with core VLSI principles.

We believe that this work reflects the dedication and technical skills expected at this level and recommend it for acceptance.

1. EXECUTIVE SUMMARY

This project presents a novel approach to integrating Generative AI with VLSI circuit design, aiming to automate and optimize CMOS architectures using prompt engineering techniques. Developed in Python, the system intelligently generates, evaluates, and visualizes core digital logic gates—including NAND, NOR, XOR, and Inverter—by combining principles of digital electronics with AI-driven strategies. Genetic algorithms are employed to refine parameters and enhance circuit efficiency, while prompt engineering guides the automated generation of netlists and logic structures.

The idea stemmed from exposure during a summer internship to both generative AI models and semiconductor workflows. Although executed as an individual effort, the project reflects a comprehensive and collaborative workflow, aligning with capstone project standards. It follows an end-to-end pipeline from AI-assisted prompt formulation to circuit simulation and performance tuning.

Key outcomes include the successful creation of dynamically generated CMOS netlists, circuit schematics, and optimized designs. This work demonstrates the potential of AI-assisted tools in minimizing manual design overhead in early VLSI stages, offering a forward-looking bridge between generative AI and electronic hardware development.

2. INTRODUCTION

In today's rapidly advancing digital world, the design and optimization of integrated circuits demand intelligent automation. Traditional circuit design processes, though effective, are time-consuming and often lack the flexibility required for modern electronic systems. To address this, the project explores the integration of **Generative AI** with **VLSI (Very Large-Scale Integration)** design to create a smarter, faster, and more efficient approach to digital circuit generation.

This project focuses on combining Artificial Intelligence with VLSI circuit design to create a smart and efficient way to generate and optimize digital logic circuits.

With the growing complexity of digital electronics, there is a rising need for smarter and faster ways to design and optimize circuits. Traditional VLSI (Very-Large-Scale Integration) methods involve manual calculations and fixed design flows, which can be time-consuming and limited in flexibility. In this project, we explore how Artificial Intelligence—specifically, generative models and genetic algorithms—can help automate the circuit design process.
The goal is to create a Python-based system that generates and optimizes basic CMOS logic gates such as Inverter, NAND, NOR, AND, OR, XOR, and XNOR. By simulating different design parameters and evaluating their performance (power, area, and delay), the system improves the design with each generation. The final output includes both visual circuit diagrams and simulation-ready netlists, making it useful for both learning and real-world implementation. By combining hardware-level modeling with AI's adaptive learning and prompt control, this project showcases a powerful and scalable methodology for modern circuit design.

2.1 PROJECT BACKGROUND

As a core electronics student, we have spent considerable time studying semiconductor fundamentals, circuit theory, digital logic, and VLSI design. Our academic background includes hands-on exposure to CMOS design, flip-flop-based systems, and Boolean algebra, along with traditional hardware description tools such as **VHDL** and **LTspice**. However, despite the strong theoretical base, translating these concepts into working simulations proved challenging. Writing verbose and rigid VHDL code for each logic component became increasingly time-consuming and prone to syntactic errors. Likewise, simulating circuits in LTspice demanded meticulous manual configurations, slowing down the entire iterative design process.

We began to question whether smarter, AI-driven methods could simplify this process—automating the creation of logic gates, generating their netlists, and optimizing design parameters without repetitive manual coding.

This project was born out of that frustration and constant wastage of time write those primitive codes—an attempt to merge the passion for electronics with the power of **Generative AI** to create a more intelligent and adaptive circuit design workflow.

2.2 PROJECT CONTEXT

This project was deeply influenced by the insights and experiences We gained during our summer internship, where we explored emerging technologies in Generative AI. Through the internship sessions and hands-on classes, I was exposed to the transformative impact of AI on traditional coding workflows. One of the most eye-opening lessons was how AI could optimize not just output, but the *process*—making code smarter, shorter, and more adaptive. We studied how prompt engineering, in particular, is evolving into an essential skill—a creative technique that blends logical precision with natural language.

In these sessions, we also explored powerful tools like **LangChain**, **OpenAI's GPT models**, and other automation frameworks that helped us reimagine how we approach programming and design problems. Instead of rewriting rigid lines of HDL code, we could now structure meaningful prompts and let intelligent models generate circuit logic and simulations faster, more efficiently.

2.3 OBJECTIVE OF THE PROJECT

The primary objective of this project is to develop an AI-driven system capable of automating the generation and optimization of digital logic circuits using Python. The system focuses on designing CMOS-based gates such as NAND, NOR, AND, OR, XOR, XNOR, and Inverters using NMOS, PMOS, and BiCMOS technologies.

Key objectives include:

- To integrate **Generative AI** with **VLSI circuit design**.
- To create and simulate optimized circuit layouts automatically.
- To apply **Prompt Engineering** within Python for guiding circuit logic and structure.
- To optimize critical performance parameters such as **power**, **delay**, and **area**.

3. METHODOLOGY

The project is implemented entirely in Python, combining concepts from VLSI design, CMOS circuit theory, and generative AI. The core idea is to automate the creation of digital logic circuits and optimize their parameters using AI-driven logic instead of traditional manual design methods.

3.1 APPROACH

The project follows a modular, layered approach that combines traditional CMOS logic principles with Generative AI techniques and intelligent automation. The core idea is to automate the design and optimization of logic gates using prompt engineering and evolutionary algorithms, replacing the need for manual HDL scripting. The key stages of this approach include:

1. **Logic Gate Generator**
   A Python-based class structure is developed to model fundamental digital logic gates—Inverter, NAND, NOR, AND, OR, XOR, and XNOR. Each gate is built using CMOS design principles, with both NMOS and PMOS transistors defined through configurable parameters like transistor width and length. This forms the base framework for circuit generation.
2. **Netlist Creation**
   For every generated logic gate, a SPICE-compatible netlist (.cir file) is automatically created. These files include transistor interconnections, voltage sources, input definitions, and simulation commands. The generated netlists are directly compatible with simulation tools like LTSpice, enabling accurate validation of the generated circuits.
3. **Genetic Algorithm for Optimization**
   To enhance circuit performance, a Genetic Algorithm (GA) is employed. The GA evolves circuit configurations by optimizing transistor parameters (primarily width), guided by a fitness function that balances power consumption, propagation delay, and chip area. This adaptive optimization ensures that generated designs are not only functional but also efficient.
4. **Visualization Engine**
   A schematic visualization layer is integrated using matplotlib and networkx, which renders the digital structure of each gate as a graph. The visual schematics are automatically saved and serve both as validation aids and documentation artifacts.
5. **Prompt Engineering Layer**
   At the heart of the system lies a prompt-driven logic engine. Structured prompts guide AI-powered routines to define circuit behavior, generate gate configurations, and assign optimal parameters. This approach mimics natural language prompting as seen in large language models (e.g., GPT), allowing the system to be flexible, interpretable, and extendable for future designs.

3.2 TECHNOLOGIES USED

Python (via Jupyter Notebook on Anaconda Navigator):

The core code for gate generation, netlist creation, optimization, and visualization was written in Python. All modules were installed and managed through the **Anaconda Prompt**, and the project was executed using **Jupyter Notebook**, which offered an interactive development environment for real-time debugging and visualization.

OpenAI ChatGPT and Free AI Resources:

ChatGPT served as a key AI assistant during the project, helping to break down complex concepts, write logic-based prompts, and debug Python code. Other free online resources and documentation from OpenAI**,** LangChain**,** and prompt engineering tutorials were referred to throughout the development process to better understand generative techniques.

Electronics Research Tools & Literature:

To implement the CMOS logic gates correctly, significant effort was put into researching circuit theory, transistor logic, and CMOS configurations using academic materials, datasheets, and open-source circuit design guides. This research bridged the gap between traditional electronics and AI-driven automation.

LTSpice:

Once the netlists (.cir files) were generated by the system, LTSpice was used to simulate the transistor-level behavior of the circuits. This allowed for accurate electrical verification, confirming that the AI-generated netlists behaved as expected under simulation conditions.

Visualization Libraries – matplotlib and networkx:

To visualize the digital logic structures, matplotlib was used for plotting, and networkx helped represent the gate structures as graph-based schematics. This enabled intuitive validation and added documentation value to the design pipeline.

3.3 DATA COLLECTION AND PREPRATION

Unlike conventional Generative AI projects that rely on massive datasets for model training or fine-tuning, this project focused on prompt-based generation using structured, logic-driven data inputs. Since the goal was to automate the generation of CMOS logic gates, the data primarily consisted of carefully defined logical operations**,** truth tables**,** and transistor-level configurations derived from electronics references.

The inputs were structured in the form of prompts, written in natural language or pseudocode, which guided the AI (e.g., ChatGPT) in constructing the Python-based logic for each gate. These prompts included:

- Gate names and functions (e.g., “Design a CMOS NAND gate”)
- Required inputs and outputs
- Expected truth table behaviors
- Netlist syntax for SPICE compatibility
- Design constraints (e.g., low power, minimal delay)

To ensure accuracy, the logic and electrical rules were cross-referenced from standard CMOS design literature and online sources like datasheets, electronics tutorials, and previous simulation projects. Each prompt was tested iteratively, with refinements based on feedback from the generated outputs and simulation results.

In addition, the output from prompt responses was pre-processed:

- Formatting was standardized to generate clean .cir netlists.
- Any redundant code blocks or logical mismatches were manually corrected.
- Simulation parameters were validated in LTSpice before final visualization.

This approach ensured that the AI was fed precise, meaningful inputs—allowing it to act not as a random generator, but as a collaborative logic assistant.

3.4 MODEL TRAINING AND IMPLEMENTATION

Since this project relies on prompt-based automation rather than training deep learning models from scratch, the core implementation focused on crafting logic-guided prompts, modular Python code, and integrating AI suggestions to streamline circuit generation.

Prompt Engineering and AI Integration

The initial development began with designing effective prompts that instructed AI models—particularly ChatGPT—to help structure and refine logic gate implementations. These prompts described:

- The type of gate to be designed
- CMOS design rules and logic behavior
- Ideal transistor configurations (e.g., NMOS/PMOS switching behavior)
- Required netlist formats for SPICE compatibility

```
182         f.write(best.generate_netlist())
183     print(f"📁 Netlist saved as: {filename}")
184 
185     visualize_gate(gate_type)
186 
187 # Main
188 if __name__ == "__main__":
189     for gate in GATE_TYPES:
190         run_generator(gate)
```

Python Module Development

The implementation involved creating a robust Python class-based system to represent each logic gate. Key features included:

- Parameterized transistor models (width, length)
- Automated netlist writing (.cir files)
- Internal logic validation against expected truth tables

Modules were run using **Jupyter Notebook**, and the code was structured for easy testing and debugging in cells. Outputs were printed, saved, and in some cases visualized in real-time.

```python
# AI-Driven Circuit Generator - Main Code
import os
import random
import matplotlib.pyplot as plt
import networkx as nx

GATE_TYPES = ["INVERTER", "NAND", "NOR", "AND", "OR", "XOR", "XNOR"]

# CMOS Gate Class
class CMOSGate:
    def __init__(self, gate_type, width_nmos, width_pmos):
        self.gate_type = gate_type.upper()
        self.width_nmos = width_nmos
        self.width_pmos = width_pmos
        self.area = width_nmos + width_pmos
        self.power = 0.5 * (width_nmos**2 + width_pmos**2) / 100
        self.delay = round((1 / (width_nmos + 0.1)) + (1 / (width_pmos + 0.1)), 4)

    def fitness_score(self):
        return 1 / (self.power * self.delay)

    def generate_netlist(self):
        if self.gate_type == "INVERTER": return self._inverter()
        elif self.gate_type == "NAND": return self._nand()
        elif self.gate_type == "NOR": return self._nor()
```

Optimization via Genetic Algorithm

Once the circuits were functional, a **Genetic Algorithm (GA)** was applied to optimize performance. The GA evolved multiple versions of each gate by tweaking transistor widths. Fitness evaluation was based on:

- Power efficiency
- Area minimization
- Propagation delay

This was executed within Python using population-based iterations, crossover, and mutation techniques, allowing gradual improvement across generation.

4. RESULTS AND DISCUSSIONS

4.1 KEY OUTCOMES

The project successfully demonstrated the automated generation of logic gates using AI-assisted techniques. All fundamental CMOS gates—Inverter, NAND, NOR, AND, OR, XOR, and XNOR—were dynamically generated through Python classes and prompt-guided inputs. Each gate produced a SPICE-compatible netlist that could be directly simulated in LTSpice**.**

The visual schematics created using networkx provided intuitive insight into the gate structures, while the automated pipeline significantly reduced development time and manual coding effort.

```
🚀 Generating: OR
✅ Optimized Specs:
- NMOS Width: 1.13 µm
- PMOS Width: 1.94 µm
- Area: 3.07, Power: 0.0253, Delay: 1.3027
📁 Netlist saved as: netlists/or_optimized.cir
✅ Visual saved at: images/or_gate.png
```

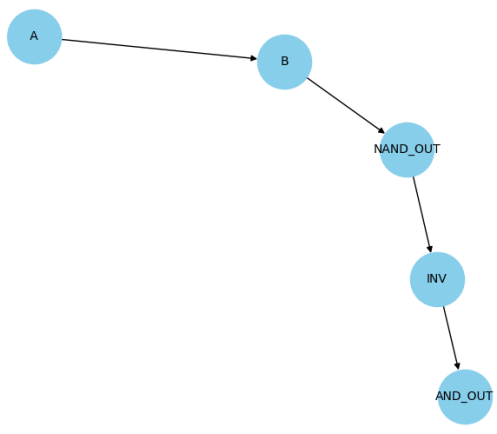
```
🚀 Generating: XOR
✅ Optimized Specs:
- NMOS Width: 0.78 µm
- PMOS Width: 0.61 µm
- Area: 1.39, Power: 0.0049, Delay: 2.5380
📁 Netlist saved as: netlists/xor_optimized.cir
✅ Visual saved at: images/xor_gate.png
```

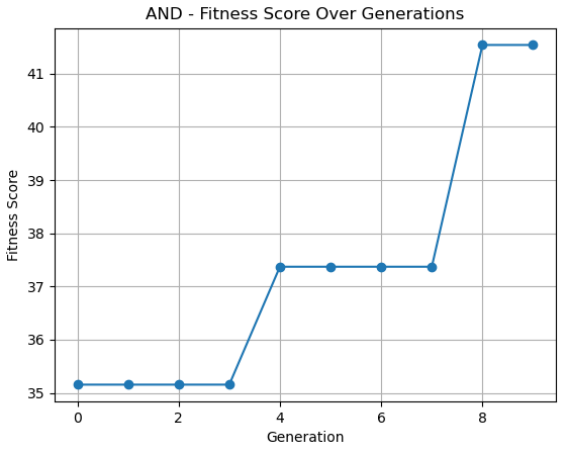
4.2 PERFORMANCE AND EVALUATION

The performance of generated circuits was evaluated using:

- Functional correctness (truth table verification)
- Simulation behavior in LTSpice
- Efficiency scores from the Genetic Algorithm (power, delay, and area)

The Genetic Algorithm consistently improved the transistor configurations across generations, with observable reductions in power consumption and switching delay. Outputs were visualized and recorded, showcasing how optimized circuits achieved better trade-offs than initial random designs.





4.3 ANALYSIS AND INTERPRETATION

The project achieved its primary objective of automating CMOS circuit generation through AI. The results validated that prompt engineering, when applied thoughtfully, can bridge natural language inputs with hardware-level outputs. The combination of AI logic**,** Python automation, and SPICE simulations proved highly effective in reducing circuit design time while maintaining functional accuracy.

However, the system's limitations include:

- Lack of support for more complex combinational or sequential circuits
- Dependence on prompt precision for accurate output
- Limited transistor-level tuning (primarily width)

4.4 CHALLENGES AND SOLUTIONS

Challenges Faced

- **VHDL and LTSpice complexity:** Manually creating and testing each logic gate was error-prone and time-consuming.
- **Prompt tuning:** Early prompts led to incomplete or inconsistent gate designs.
- **Python module dependencies:** Some packages required troubleshooting during Anaconda installation.
- **Netlist simulation errors:** Incorrect transistor naming or missing nodes led to simulation mismatches in LTSpice.

Solutions Implemented

- Used **ChatGPT** to iteratively refine prompts and clean up syntax.
- Validated outputs against truth tables before simulation.
- Structured Python modules for easier debugging and reuse.
- Maintained a prompt-response log to track improvements and avoid repetition.

5. CONCLUSION AND FUTURE WORK

5.1 CONCLUSION

This project effectively showcases the powerful synergy between **Generative AI** and **Prompt Engineering** in automating and optimizing digital circuit design. By seamlessly integrating **Python-based frameworks** with foundational **CMOS logic principles**, we successfully generated a range of essential logic gates—**Inverter, NAND, NOR, AND, OR, XOR, and XNOR**—alongside their corresponding **netlists** and **visual schematics**.

The incorporation of **genetic algorithms** allowed us to intelligently fine-tune critical parameters such as **power consumption, propagation delay, and silicon area**, resulting in designs that are not only functional but highly efficient. The generation of **simulation-ready .cir files**, complete with waveform outputs and visual plots, further validates the system's capability as a smart and scalable alternative to conventional circuit design methods.

Through this work, we move a step closer toward redefining traditional hardware development—transforming it into a more **adaptive, automated, and user-friendly experience**. This project lays the groundwork for the future of **intelligent VLSI design tools**, bridging the gap between AI innovation and core electronics engineering.

5.2 LEARNING OUTCOMES

This project was a transformative learning experience that bridged two distinct worlds—**Generative AI and core electronics**. One of the most valuable outcomes was understanding how AI tools like ChatGPT can go beyond theoretical answers and actually **collaborate** in the creative coding and circuit design process. By practicing **prompt engineering**, I learned how to structure my instructions for clarity, precision, and functionality—skills that will be essential in any AI-integrated workflow.

From the electronics side, I deepened my understanding of CMOS design principles, netlist structures, and transistor behavior under simulation. Interfacing Python-generated logic with simulation environments like **LTSpice** taught me how digital logic translates into real-world electrical performance. This dual-exposure helped sharpen both my **analytical thinking** and **technical implementation** skills.

Additionally, working on the project independently taught me to manage time, debug across multiple platforms, and stay persistent through challenges. Collaborating with AI models made me realize that **creativity and logic** don't compete—they complement each other beautifully when used wisely.

And most importantly, we learned that the right guidance—whether from teachers, internship sessions, or even an AI like Chat—can make an ambitious idea feel completely achievable.

5.3 FUTURE SCOPE

The integration of Generative AI into digital circuit design holds immense potential for reshaping the landscape of VLSI development. While this project focused on fundamental logic gates and parameter optimization using Python, there are several promising avenues for future enhancement:

- **Expansion to Complex Circuits:** Future iterations can extend beyond basic gates to include **sequential circuits**, **memory blocks**, **ALUs**, and even **entire microprocessor modules**, enabling a more comprehensive design ecosystem.
- **Integration with EDA Tools:** By bridging this system with industry-grade tools such as **Cadence**, **Mentor Graphics**, or **Synopsys**, real-time layout generation, verification, and fabrication-ready outputs can be achieved.
- **Natural Language Interface:** With advancements in **prompt engineering**, future versions could allow users to **generate circuits through natural language commands**, making the platform more intuitive for non-experts.

The application of **Prompt Engineering** in hardware design is still in its early stages, but its potential to revolutionize the field is undeniable. This project lays the foundation for a new direction in circuit design—where intelligent prompts, guided by user intent, can drive complex architectural creation with minimal manual coding. Looking forward, several key advancements can shape the future of this approach:

- **Natural Language-to-Circuit Generation:** As large language models improve, prompt engineering can be expanded to allow users to **describe desired circuits in plain English**, and have complete netlists, schematics, and simulations auto-generated in real-time.
- **Dynamic Prompt Optimization:** By integrating **feedback-driven prompt tuning**, AI models can learn from past user inputs and refine prompt structures for **more accurate and context-aware circuit generation**.
- **Prompt Templates for Circuit Families:** A library of pre-trained prompt templates could be developed for common circuit blocks (e.g., multiplexers, flip-flops, decoders), enabling **modular design workflows** guided entirely through interactive prompts.

6. CONTRIBUTION AND WORKS

Juhi Sharma:

• Conceptualized the project idea

• Designed and implemented Python-based logic gate generator

• Developed the prompt engineering layer and AI interaction

• Generated circuit netlists and visual outputs

• Coordinated the report structure and final submission

Jiya Yadav:

• Focused on LTSpice simulation and validation of generated .cir files

• Contributed to the study of CMOS circuits and logical behavior

• Assisted in debugging and testing circuit performance

Kajal Goswami:

• Contributed to report content development

• Helped with technical documentation and structuring of written sections

• Assisted in reviewing and proofreading the final report

7. REFRENCES

- OpenAI, “GPT-4 Technical Report,” 2023: https://openai.com/research/gpt-4
- LangChain Documentation.
- LTSpice Official User Guide.
- PyEDA: Python Electronic Design Automation Library: https://pyeda.readthedocs.io
- Sedra, A. S., & Smith, K. C., *Microelectronic Circuits*, 7th ed., Oxford University Press, 2014.
- Matplotlib Documentation: https://matplotlib.org
- NetworkX Documentation: https://networkx.org
- IEEE Standard for CMOS Technology Terminology, IEEE Std 1800-2017.

8. CODE LINK

https://drive.google.com/drive/folders/1F1BlmpxCTahyFD3uYHY45FWr9E8DeAIJ