CircuitLM: A Multi-Agent LLM-Aided Design Framework for Generating Circuit Schematics from Natural Language Prompts

Anonymous ACL submission

Abstract

Generating accurate circuit schematics from high-level natural language descriptions remains a persistent challenge in electronics design, as large language models (LLMs) frequently hallucinate in granular details, violate electrical constraints, and produce non-machine-readable outputs. We present CircuitLM, a novel multi-agent LLM-aided circuit design pipeline that translates user prompts into structured, visually interpretable `CircuitJSON` schematics through five sequential stages: (i) LLM-based component identification, (ii) canonical pinout retrieval, (iii) chain-of-thought reasoning by an electronics expert agent, (iv) JSON schematic synthesis, and (v) force-directed SVG visualization. Anchored by a curated, embedding-powered component knowledge base. While LLMs often violate electrical constraints, CircuitLM bridges this gap by grounding generation in a verified and dynamically extensible component database, initially comprising 50 components. To ensure safety, we incorporate a hybrid evaluation framework, namely Dual-Metric Circuit Validation (DMCV), validated against human-expert assessments, which achieves high fidelity in microcontroller-centric designs. We evaluate the system on 100 diverse embedded-systems prompts across six LLMs and introduce DMCV to assess both structural and electrical validity. This work bridges natural language input to deployable hardware designs, enabling reliable circuit prototyping by non-experts. Our code and data will be made public upon acceptance.

1 Introduction

Electronics design traditionally demands deep domain expertise, precise component knowledge, and painstaking schematic drafting—barriers that exclude novices, hobbyists, and rapid prototypers from innovation. Recent advances in large

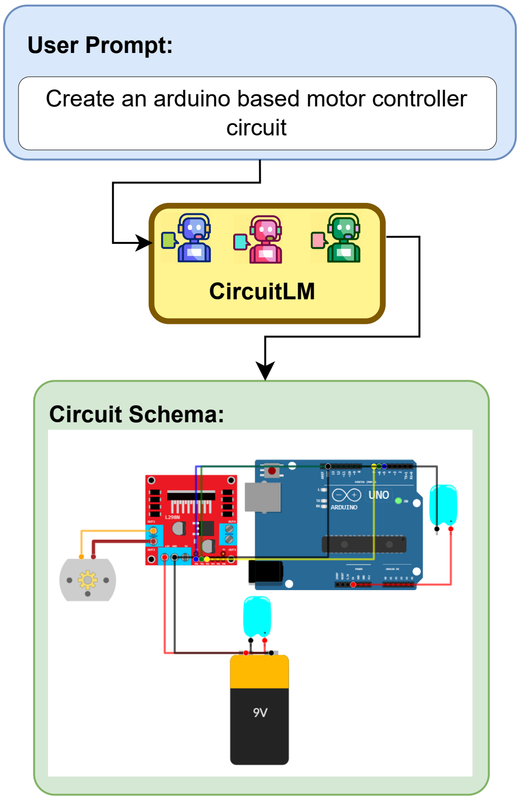

Figure 1: Example of a circuit schema generated by CircuitLM from one of our benchmark prompts.

language models (LLMs) have sparked interest in automating this process via natural language prompts, such as "Build a circuit to blink an LED with a button using Arduino." Yet, LLMs falter here: they hallucinate non-existent pin labels (*e.g.*, inventing "`LEDPIN`" instead of "`D13`"), ignore power rails or safety resistors, and output prose rather than machine-readable formats ill-suited for tools like Fritzing[1] or Proteus Design Suite.[2]

Prior efforts, such as PINS100 and MICRO25 benchmarks (Jansen, 2023), rely on automated pinout matching or manual expert review of schematics and code, but overlook holistic elec-

[1] https://fritzing.org/
[2] https://www.labcenter.com/

trical validity, structured outputs, and visualization. This gap motivates CircuitLM, our multi-agent LLM framework that decomposes prompt-to-circuit generation into modular stages with agent-enforced rules and criteria for automated verification. By integrating LLM reasoning with a local vector database of canonical components, CircuitLM grounds designs in real hardware while enforcing logical wiring (*e.g.*, SDA-to-SDA nets, current limiting).

Our system includes a five-stage pipeline: (i) Identification (NER for components), (ii) Retrieval (Qwen3 embeddings over a ChromaDB index with auxiliary fuzzy string matching), (iii) Reasoning (CoT for pin-level logic), (iv) Generation (`CircuitJSON`), and (v) Visualization (force-directed SVG layouts). A curated knowledge base for electrical components, enabling semantic matching and strict pin enforcement.

We rigorously evaluate six frontier LLMs on 100 prompts using an independent QA agent. Our scoring metric combines Library Compliance (checking exact pins/IDs) and Electrical Logicality (verifying functional soundness) into a weighted total. Furthermore, we provide a suite of open-source artifacts—including our dataset, database, and visualizer—to facilitate reproducibility. Our results demonstrate CircuitLM's efficacy, and overall performance averages across all benchmarks range from 8.503 to 7.865. This work makes four primary contributions: (1) CircuitLM, a modular multi-agent pipeline that transforms natural language into structured circuit connections; (2) `CircuitJSON`, a schematic description format; (3) Dual-Metric Circuit Validation (DMCV), a dual-metric evaluation framework for verifying both library compliance and electrical logic; and (4) a grounded architecture that reduces pin hallucinations through local vector database retrieval.

2 Related Works

Electronic circuit design automation has evolved from deterministic, heuristic-driven algorithms toward stochastic, agentic AI frameworks, reflecting the increasing complexity of embedded systems and the demand for higher-level abstraction in hardware synthesis. Early applications of Machine Learning (ML) in Electronic Design Automation (EDA) focused on localized optimization within the physical design flow, employing Convolutional Neural Networks (CNNs) for routability prediction and Graph Neural Networks (GNNs) for netlist representation and parasitic estimation (Talebzadeh et al., 2021). While effective at specific VLSI stages, these approaches lacked the semantic capacity to interpret high-level user intent or natural language specifications. The success of Large Language Models (LLMs) in software engineering subsequently motivated research into hardware generation, particularly in Hardware Description Languages (HDLs). Systems such as VerilogEval (Thakur et al., 2023) and ChipChat (Blocklove et al., 2023) demonstrated LLM-based RTL generation via conversational interfaces. However, generating complete electronic schematics introduces substantially higher complexity, requiring physical grounding through accurate pin mappings, power management, and integration of heterogeneous peripherals. A key challenge in autonomous schematic generation is the hallucination of connectivity, where models invent invalid pin labels or violate electrical constraints. Recent work has explored structured intermediate representations to mitigate this issue. Schemato (Matsuo et al., 2025) reconstructs schematics from formal netlists, while EESchematic (Liu and Chitnis, 2025) investigates end-to-end generation using standardized symbol libraries. Nonetheless, existing datasets and frameworks—such as LLM4Netlist (Ye et al., 2025)—remain focused on synthesizable digital logic, leaving embedded systems and analog sensor-rich designs largely unaddressed.

To mimic human engineering workflows, research has pivoted toward Multi-Agent Systems (MAS), which have already demonstrated good efficacy in mathematical (Wan et al., 2025) and physics (Siddique et al., 2025) reasoning. By decomposing design into specialized roles—such as design, verification, and routing agents—MAS frameworks employ "debate and critique" mechanisms to reduce error rates (Hong et al., 2024). Cutting-edge frameworks like MenTeR (Chen et al., 2025) utilize diagram-aware RAG (Retrieval-Augmented Generation) and "circuit think tanks" for automated RF/analog design. Furthermore, studies on agent topologies suggest that distributed reasoning consistently outperforms monolithic models by isolating sub-task complexity (Pan et al., 2025).

CircuitLM bridges these gaps by introducing a novel multi-agentic pipeline that enforces

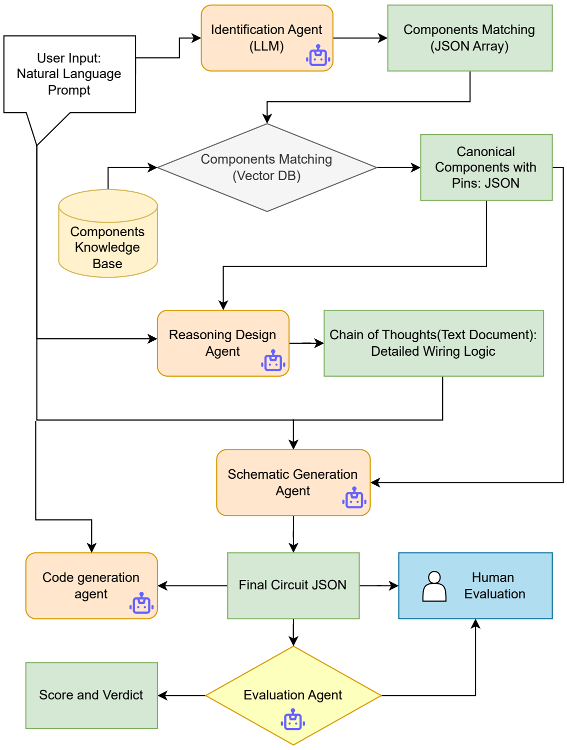

Figure 2: An overview of the CircuitLM framework.

physical grounding through a canonical retrieval database. By moving beyond simple text generation to a structured, reasoning-first approach, this study provides the first comprehensive evidence that agentic frameworks can handle the multi-dimensional constraints of real-world embedded prototyping, producing manufacturing-ready outputs that are both logically sound and visually interpretable.

3 Methodology

We bring forth a multi-step, multi-agent pipeline that transforms a high-level, natural language project idea (the user prompt) into a structured, logically sound circuit definition. The pipeline contains the following stages:

3.1 Stage I: Components Extraction (Identification Agent)

In the initial phase, a specialized **Identification Agent**, powered by an LLM, parses the user's natural language project idea. The agent's primary objective is to perform named entity recognition (NER) and intent analysis to extract a comprehensive list of necessary hardware. We gave the Identification Agents the generic names of components (*i.e.* Pressure Sensor, Arduino Uno, IMU Sensor etc.) so they could use the available components. In order to minimize the number of input tokens, we just sent fixed keywords as component names rather than the complete component data. The output of this stage is a structured JSON array containing the component names (*e.g.*, `[“Arduino”, “Motor Driver”, “DC motor”]`), which serves as the inventory for the subsequent stages. Figure 3 depicts a component identification LLM agent receiving user input and returning an array of required components.

3.2 Stage II: Component Matching (Retrieval Agent)

To ensure the design is grounded in physically realizable hardware, the list from Stage I is passed to the **Retrieval Agent**. First, the agent applies an embedding model to carry out a similarity search over a local vector-based knowledge base (Lewis et al., 2021) containing canonical electronic components. Because semantic search alone cannot identify every component, a secondary fuzzy search is performed using a dictionary of equivalent components. The results from the semantic and fuzzy searches are then amalgamated. By mapping the identified component names to corresponding records in the `component` database, the agent constructs a dictionary that associates each component with its verified canonical name and precise pin-out definitions, as illustrated in Figure 4. This step is crucial as we need the exact pin names and numbers to create a circuit schematic. LLMs often hallucinate such details, so we retrieve this information directly from a curated database. To prevent hallucinations when a user requests hardware not present in the local knowledge base, the system triggers an OOD (out-of-distribution) flag, halts execution, and requests human evaluation.

3.3 Stage III: Design Reasoning (Electronics Expert Agent)

The **Electronics Expert Agent** acts as the cognitive core of the pipeline. Taking the original user prompt, the validated component list, and the associated pin definitions as input, this agent performs high-level engineering reasoning. It produces a structured Chain-of-Thought (CoT) document (Wei et al., 2023). This document explicitly details the circuit's functional goals, power requirements, safety considerations, and the pin-level wiring logic required to achieve the desired behavior. By decomposing the complex electronics requirements into these hierarchical sub-tasks, the agent utilizes a least-to-most prompting strategy (Zhou et al., 2023). This intermediate step reduces complicated significant circuit problems into smaller, multi-step problems and enhances the circuit's electrical logicality (Wang et al., 2023) before any code is generated. Figure 5 illustrates how the retrieved component information and user prompt are fed into the CoT agent, which then returns a detailed reasoning text.

3.4 Stage IV: Schematic Generation (Circuit Generation Agent)

This stage is handled by the **Circuit Generation Agent**, which interprets the detailed CoT document and canonical pin data into a machine-readable format, as shown in Figure 7. The agent outputs a strictly formatted `CircuitJson` object, which is largely inspired by the Wokwi[3] simulation platform by Asparuhova et al. (2024). This object includes precise component placement (coordinates and rotation) and a comprehensive list of pin-to-pin connections. The resulting JSON file is ready for downstream rendering in the schematic viewer. The pseudocode in Listing 1 shows the structure of `CircuitJSON`.

3.5 Stage V: Schematic Visualizer

A visualization engine converts the generated `CircuitJSON` into an interpretable circuit schematic using SVG-based component representations rendered directly in the browser. Each component in the local database is mapped to a corresponding SVG symbol, enabling deterministic, platform-agnostic visualization via TypeScript-based SVG manipulation. For previously unseen or undefined components, the engine dynamically generates a generic rectangular symbol with labeled pins, ensuring graceful handling of unknown component specifications. Layout generation begins with a force-directed placement algorithm (Schönfeld and Pfeffer, 2019), modeling components as nodes and electrical connections as spring forces. Repulsive forces prevent overlap, while attractive forces promote proximity between connected components, producing a compact and readable initial layout. Wire routing is then performed using a multi-strategy Manhattan-style approach (Lee, 1961), evaluating straight, L-, Z-, and U-shaped paths. Routing heuristics prioritize shorter paths, fewer bends, and minimal crossings to reduce visual clutter and improve schematic clarity (Figure 7). The overall prompt-to-circuit pipeline is summarized in Figure 2.

Unlike traditional EDA workflows that focus on netlist generation for fabrication, our system adopts a visual-first synthesis paradigm. By omitting the netlist stage, we prioritize human-readable schematics that emphasize functional structure and spatial relationships, making the system lightweight and well-suited for early prototyping and educational use.

[3] `https://wokwi.com/`

3.6 Firmware Code Generation

While the primary pipeline focuses on circuit generation and evaluation, a firmware code-generation agent is also included to demonstrate the potential for automated firmware deployment, although it is not part of the core design workflow. We include firmware code generation for ESP32 and Arduino boards, if available, in the following circuit. We pass the prompt and the generated `CircuitJSON` to the code-generation agent. LLMs have already demonstrated state-of-the-art performance in standard software engineering benchmarks, such as HumanEval (Chen et al., 2021) and MBPP (Austin et al., 2021). Given that the software-side logic for basic embedded tasks often follows well-documented, repeatable patterns, we consider that generating syntactically correct firmware is a relatively trivial task for frontier models compared to hardware synthesis.

3.7 SPICE Simulation and ERC

Although circuit simulation and electrical rule checking (ERC) are integral components of traditional electronic design automation (EDA) workflows, they were intentionally excluded from the proposed system due to both technical limitations and misalignment with the system's objectives.

SPICE-based simulators are fundamentally designed for continuous-time analog modeling, excelling at transistor-level and passive component analysis. However, the circuits generated by our system are microcontroller-centric, frequently incorporating platforms such as Arduino or ESP32, whose behavior is dominated by firmware execution, digital I/O state machines, and peripheral configuration. Electrical Rule Checking (ERC) typically operates on a formal netlist with pin-level electrical constraints (drive strength, voltage domains, pin direction, and power classes). CircuitLM deliberately bypasses netlist generation in favor of producing human-readable, visually structured schematics intended for early-stage design exploration, education, and prototyping.

4 Experimental Setup

The experiment was designed to ensure fair and robust benchmarking of the candidate LLMs. We run the experiments multiple times, and the final score is calculated as the mean across all runs. This setup contains the implementation of the knowledge base and the configuration of the LLM agents.

4.1 Local Knowledge Base

A key element of the system is the curated component library, stored in a ChromaDB vector database. This database serves two primary functions:

- **Components List**: Providing a finite and authoritative set of components, including their canonical identifiers, mandatory pin labels, and known aliases. This enables strict evaluation while still supporting flexible component retrieval.
- **Semantic Component Resolution**: Translating verbally requested components generated by the LLM in Stage I into definitive canonical component keys and pin sets required for the final `CircuitJson` representation in Stage IV.
- **Database**: ChromaDB (persistent client). Its lightweight architecture and schema flexibility allow rich component metadata (pins, aliases, specifications) to be stored alongside embeddings, simplifying the mapping from semantic queries to canonical component definitions.
- **Embedding Model**: `Qwen3-Embedding-0.6B` (Yang et al., 2025) model is used to compute dense vector representations of component descriptions. Component matching is performed using cosine similarity with an empirically determined similarity threshold. The Qwen3 embedding model provides high-quality semantic representations for short, technical phrases and structured component metadata in electronics domains. The model offers a favorable balance between retrieval accuracy and computational efficiency (Zhang et al., 2025b), making it well-suited for persistent vector search (Lewis et al., 2021) in ChromaDB.
- **Library Structure**: Each component entry includes a canonical name (key), mandatory pin labels, physical dimensions (width and height), aliases, a natural language description, category, typical usage, and technical specifications. Strict adherence to the defined pin labels (e.g., `D21`, `GND`, `VCC`) is enforced during final evaluation (Dimension 1).

4.2 Benchmarked Large Language Models

A total of 6 distinct LLMs are benchmarked as candidate models across the key text-generation stages of the proposed multi-agent pipeline (Stages I, III, and IV). The evaluated models are as follows:

- `openai/gpt-5-mini`
- `google/gemini-2.5-flash`
- `deepseek/deepseek-chat-v3.1`
- `qwen/qwen3-235b-a22b-2507`
- `x-ai/grok-code-fast-1`
- `meta-llama/llama-3.3-70b-instruct`

All models were accessed through a unified API interface to ensure consistency in prompt formatting and execution conditions. The temperature parameter was fixed at 0 for all experimental runs.

4.3 Dataset (User Prompts)

Ye et al. (2025) provides a prompt-based dataset for netlist generation; it focuses exclusively on synthesizable digital logic circuits in Verilog, excluding embedded systems, sensors, IoT, or Arduino-specific components, PWM motor drivers and sensor interfaces. Our experiment utilizes a set of 100 diverse user prompts covering a broad range of embedded systems projects, from basic LED control and motor speed control to complex multi-actuator circuits, communication bus interfacing (I^2C, SPI, UART), and sensor integration (IMU, DHT, PIR, LDR *etc.*). This ensures a rigorous test of the system's ability to handle complexity, ambiguity, and technical constraints. The CircuitLM benchmark prompts primarily focus on physical embedded systems.

4.4 Expert Human Validation

To calibrate the LLM-based Evaluation Agent and validate the proposed DMCV metric, we conducted a rigorous manual audit of 25% of the synthesized circuits. This validation phase utilizes a panel of three electrical engineers serving as domain experts to evaluate the schematics for functional viability, safety, and industrial best practices. Beyond static schematic review, they performed dynamic functional verification. Samples from the generated dataset were reconstructed within high-fidelity virtual prototyping environments, specifically Wokwi and Tinkercad Circuit Simulator. We see that expert judgments correlate strongly with the evaluation agent scores.

4.5 Evaluation and Scoring

While existing benchmarks such as PINS100 and MICRO25 evaluate component pinout knowledge through automated JSON matching and microcontroller-based circuit generation via manual expert review of schematics and code (Jansen, 2023), they lack machine-processable output formats and fine-grained electrical correctness checks, such as net assignment validation, power integrity, or pin-level compatibility. PINS100 focuses narrowly on binary pin accuracy without wiring logic, while MICRO25's PASS@1 relies on human verification of functionality. Consequently, these benchmarks provide limited coverage of practical electrical validity.

To address this gap, we introduce **DMCV (Deterministic and Model-assisted Circuit Validation)**, a hybrid evaluation framework for assessing the correctness of LLM-generated electronic circuit designs. DMCV combines deterministic, rule-based validation with LLM-powered electrical reasoning to produce a unified quality score on a 0–10 scale. This hybrid design enables both precise detection of structural errors and contextual assessment of electrical plausibility.

Electrical rationality is evaluated using an independent QA Agent instantiated with the `anthropic/claude-sonnet-4.5` model, selected for its strong long-context reasoning and instruction-following performance. Prior work has shown that Claude Sonnet models exhibit lower rates of format violations and higher consistency when performing structured, rule-based evaluation tasks (Zhang et al., 2025a; Zheng et al., 2023). Importantly, utilizing a model family distinct from the generation models reduces potential bias arising from self-evaluation.

DMCV Scoring. The score $S_{\text{DMCV}} \in [0, 10]$ is computed as a weighted combination of an electrical logic score and a library compliance score:

$$S_{\text{DMCV}} = 0.6\, S_{\text{logic}} + 0.4\, S_{\text{comp}}$$

The weighting puts more gravitas on electrical correctness over syntactic or library-level compliance, reflecting the higher safety and functionality risks associated with electrical design errors.

Library Compliance Score. The library compliance score captures violations related to component schemas and pin-level connectivity. Let n_s denote the number of component definition errors and n_p the number of pin assignment or net connection errors. The compliance score is computed using a penalty-based formulation:

$$S_{\text{comp}} = \max\left(0, \frac{100 - 10n_s - 5n_p}{10}\right).$$

The penalty weights for n_s and n_p are assigned based on the principle of structural integrity. We weight component definition errors (n_s) twice as heavily as pin assignment errors (n_p) because the former represents a failure in physical grounding—hallucinating a component footprint renders all associated nets logically void. Conversely, pin assignment errors are treated as localized logical faults that, while disruptive to functionality, do not violate the fundamental hardware schema defined in our canonical database.

This formulation begins with a nominal score of 100, applies fixed penalties based on detected errors, and rescales the result to the 0–10 range to match the electrical logic score.

Electrical Logic Score. The electrical logic score evaluates the circuit's functional plausibility, safety, and adherence to fundamental electrical design principles. The QA Agent acts as a *Lead QA Engineer*, identifying and categorizing issues by severity:

1. **Fatal Errors** (-2.0 each): Power rail shorts, direct GPIO-to-supply conflicts, critical bus miswirings etc.
2. **Major Errors** (-1.0 each): Missing current limiting on driven loads, voltage level mismatches, floating grounds, or insufficient power isolation.
3. **Minor Errors** (-0.5 each): Passive component omissions affecting reliability (*e.g.*, LED without a resistor) or ambiguous polarity.

4. **Logical Warnings** (-0.25 each): Non-fatal design concerns such as power budget margin issues or missing decoupling capacitors.

Let n_f, n_m, n_{mi}, and n_w denote the number of fatal errors, major errors, minor errors, and logical warnings, respectively. The electrical logic score is computed as:

$$S_{\text{logic}} = \text{clip}\Big(10 - 2.0n_f - 1.0n_m - 0.5n_{mi} - 0.25n_w,\ 0,\ 10\Big).$$

The score begins at 10 and is reduced according to detected issues, with the final value clamped to the interval $[0, 10]$.

5 Results

We evaluated six state-of-the-art LLMs on a benchmark set of circuit design tasks using the proposed DMCV evaluation framework. For each generated circuit, DMCV assigns scores along two complementary dimensions: **Library Compliance** (S_{comp}, 40%) and **Electrical Logicality** (S_{logic}, 60%). These scores are combined into a single overall performance metric S_{DMCV} Table 1 reports the mean library compliance score, electrical logic score, and aggregated DMCV score for each model, averaged across all benchmark tasks.

Table 1: Benchmark results across LLMs ($\mu \pm \sigma$)

Model	S_{comp} μ	S_{comp} σ	S_{logic} μ	S_{logic} σ	Overall
Gemini 2.5 Flash	9.960	0.06	7.532	0.14	8.503
Qwen-3 235B	9.941	0.07	7.285	0.15	8.347
Deepseek v3.1	9.881	0.08	7.374	0.13	8.377
Grok Fast Code	9.861	0.08	7.381	0.13	8.373
GPT-5 Mini	9.054	0.13	7.658	0.12	8.217
Llama-3.3 70b Instruct	9.479	0.09	6.789	0.16	7.865

The evaluation outputs from each model are processed to compute average per-dimension scores and weighted total scores, which are visualized in Figure 8, which is a grouped bar chart showing library and electrical scores, and Figure 9, a scatter plot chart comparing all six LLMs. These results reflect the behavior of models as served by OpenRouter[4] at the time of evaluation; due to the dynamic nature of unpinned model versions, exact reproducibility across different timespans is not guaranteed.

The aggregated DMCV scores (Table 1) range from 7.865 (Llama-3.3 70B) to 8.503 (Gemini 2.5 Flash). Gemini 2.5 Flash achieves the highest overall score, driven by near-perfect $S_{\text{comp}} = 9.960$ and robust electrical reasoning. Deepseek v3.1 and Grok Fast Code exhibit nearly identical overall performance (8.377 and 8.373 respectively), indicating a stabilization in design capabilities among frontier models. Notably, GPT-5 Mini presents a unique profile: despite having the lowest $S_{\text{comp}} = 9.054$ due to minor pin-labeling hallucinations, it achieves the highest $S_{\text{logic}} = 7.658$ in the cohort. Conversely, Llama-3.3 70B represents the benchmark's lower bound, suggesting that although it can identify components, it faces significant challenges in reasoning. Figures 10 and 11 show the scores of each LLM by prompts.

6 Discussion

The evaluation of LLMs' performance in automated circuit synthesis reveals robust capability in digital protocol mapping, contrasted with pervasive failure in functional analog reasoning. Across the compiled metrics, models consistently achieve a $S_{\text{comp}} \approx 10.0$ for standard component interfacing, yet demonstrate significant variance in S_{logic} when transitioning from logical pin-mapping to complex topological synthesis. High-precision models, specifically DeepSeek v3.1 and Grok Fast, maintained high fidelity in digital communication tasks such as SPI, UART, and I^2C interfacing. In contrast, models like GPT-5 Mini exhibited extreme performance volatility, achieving perfect scores of 10.0 on logic-heavy security system prompts while scoring 0.0 on fundamental sensor integrations such as the Real-Time Clock. A critical "analog reasoning gap" is evident in tasks requiring the design of RC low-pass filters or LC smoothing circuits, in which most models failed to produce electrically valid outputs, yielding scores of 0.0. This suggests that while LLMs excel as digital librarians for ubiquitous platforms like the ESP32 and Arduino, their limited ability to generalize to niche hardware ecosystems—such as the Franzininho—and their lack of systemic electrical intuition render them unreliable for autonomous analog design or safety-critical power applications without expert human oversight.

These results highlight the importance of a strong, high-density component library as a fundamental prerequisite for preventing LLM-driven design errors and guaranteeing schematic validity.

One of the primary goals of our research is to as-

[4] https://openrouter.ai/

sess the design reasoning and one-shot schematic generation capabilities of state-of-the-art LLMs when grounded by a canonical database. Introducing an automated repair loop would obscure the raw performance of the underlying models. Moreover, we adopted a feed-forward multi-agent architecture to minimize computational latency.

7 Ablation Study

To assess the contribution of explicit reasoning to circuit generation quality, we conducted an ablation study by removing the **Design Reasoning (Chain-of-Thought) Agent** from the proposed pipeline. In this ablated configuration, the Circuit Generation Agent receives only the user prompt and the validated component list, without access to the intermediate reasoning document.

7.1 Ablation Setup

All other stages of the pipeline remain unchanged, including component identification, component retrieval, and schematic generation. Instead of passing the CoT trace to schematic generation, we only pass the user prompt, retrieved components to the schema generation agent. The ablated system is evaluated using the same two-dimensional scoring rubric. The experiment is repeated multiple times on the same prompts, and the final score is reported as the mean across all runs.

7.2 Ablation Results

Table 2 details the impact of removing the Design Reasoning Agent from the pipeline. While most models showed performance gains through their removal—most notably DeepSeek v3.1 ($\Delta S_{\text{logic}} = +0.750$) and Gemini 2.5 Flash ($\Delta S_{\text{logic}} = +0.482$)—Llama-3.3 70b demonstrated the highest output stability ($\sigma = 0.11$). Conversely, GPT-5 Mini exhibited a significant degradation when the reasoning agent was integrated, with its electrical accuracy dropping by 0.638 ($\sigma = 0.25$).

Table 2: Impact of removing the CoT agent

Model	Full Overall	No CoT Overall	S_{logic} Δ	S_{logic} σ
Gemini 2.5 Flash	8.503	8.314	↑ +0.482	0.18
Qwen-3 235B	8.347	8.173	↑ +0.290	0.15
DeepSeek v3.1	8.315	7.880	↑ +0.750	0.24
Grok Code Fast 1	8.373	8.176	↑ +0.328	0.16
GPT-5 Mini	8.217	8.367	↓ −0.638	0.25
Llama-3.3 70b Instruct	7.865	7.840	↑ +0.092	0.11

These findings reinforce recent evidence that Chain-of-Thought prompting is not universally optimal (Meincke et al., 2025). While CoT can improve average performance for some models, it may also introduce variability or reduce accuracy on tasks the model would otherwise solve correctly, particularly for models with built-in reasoning capabilities. In such cases, explicit linguistic reasoning may interfere with internal representations—a phenomenon consistent with verbal overshadowing (Liu et al., 2025). Moreover, the added token cost and latency of CoT prompting often outrank its marginal accuracy gains for reasoning-oriented models (Meincke et al., 2025). We therefore advocate a model-aware reasoning strategy that selectively applies Chain-of-Thought prompting only when it yields measurable gains, rather than adopting it as a default.

8 Conclusion and Future Work

We present a multi-agent framework that bridges the gap between high-level natural-language intent and physically realizable electronic schematics. By decomposing the design process into modular agents for retrieval, reasoning, and synthesis, we mitigate the pervasive issues of pin hallucination and structural inconsistency inherent in general-purpose LLMs. Our results demonstrate that grounding design logic in a curated, canonical component knowledge base allows frontier models to achieve high levels of library compliance.

Future work aims to evolve the current framework into an industrial-grade EDA system. A key objective is the adoption of a heterogeneous multi-agent architecture, replacing a monolithic model with specialized agents assigned to distinct stages—using high-level reasoning models for logic design and syntactically constrained models for structured JSON generation. To improve reliability, we plan to incorporate an **Evaluation-Feedback-in-the-Loop (EFIL)** mechanism, enabling iterative correction based on evaluation errors and warnings. Furthermore, we will also explore a **Consensus-Based Evaluation Ensemble**, utilizing multiple frontier models to conduct a "cross-check" of electrical logic, thereby further reducing the probability of undetected design failures. We encourage future researchers to adopt the proposed `CircuitJSON` format as a standard for dataset creation, helping bridge natural language specifications and hardware design.

9 Limitations

Our framework presents specific limitations, primarily regarding computational overhead and latency. Due to the multi-agent architecture requiring iterative queries for each prompt, inference times can be significant. We observed variable latency contingent upon the specific model tier and external API network conditions. To mitigate this in future iterations, we intend to explore local model quantization to ensure deterministic response times. Additionally, our experimental scope was constrained to a knowledge base of 50 components; while this set was selected to represent a broad taxonomy of electrical categories and easily scalable, it remains a limited subset. Furthermore, regarding validation, the extensive volume of circuits generated by the six LLMs precluded exhaustive manual verification. Instead, we used a random sampling strategy to validate the automated assessment. Moreover, relying on a single LLM model as the primary evaluator introduces a limitation due to potential stochastic bias. Finally, the reliance on the fixed `CircuitJSON` structure constitutes a limitation on interoperability.

References

Katya Asparuhova, Shehova Daniela, Stanislav Asenov, Hristo Kanevski, and Anatoliy Parushev. 2024. Using wokwi simulator to support engineering student learning in microcontrollers and sensors. pages 1–4.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. 2021. Program synthesis with large language models. *Preprint*, arXiv:2108.07732.

Jason Blocklove and 1 others. 2023. Chip-chat: Challenges and opportunities in conversational hardware design. In *Proceedings of the 2023 ACM/IEEE Workshop on Machine Learning for CAD*.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, and 39 others. 2021. Evaluating large language models trained on code. *Preprint*, arXiv:2107.03374.

Pin-Han Chen, Yu-Sheng Lin, Wei-Cheng Lee, Tin-Yu Leu, Po-Hsiang Hsu, Anjana Dissanayake, Sungjin Oh, and Chinq-Shiun Chiu. 2025. Menter: A fully-automated multi-agent workflow for end-to-end rf/analog circuits netlist design. *Preprint*, arXiv:2505.22990.

Sirui Hong and 1 others. 2024. Metagpt: Meta programming for multi-agent collaborative framework. *arXiv preprint arXiv:2308.00352*.

Peter Jansen. 2023. From words to wires: Generating functioning electronic devices from natural language descriptions. *Preprint*, arXiv:2305.14874.

C. Y. Lee. 1961. An algorithm for path connections and its applications. *IRE Trans. Electron. Comput.*, 10:346–365.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2021. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Preprint*, arXiv:2005.11401.

Chang Liu and Danial Chitnis. 2025. Eeschematic: Multimodal-llm based ai agent for schematic generation of analog circuit. *Preprint*, arXiv:2510.17002.

Ryan Liu, Jiayi Geng, Addison J. Wu, Ilia Sucholutsky, Tania Lombrozo, and Thomas L. Griffiths. 2025. Mind your step (by step): Chain-of-thought can reduce performance on tasks where thinking makes humans worse.

Ryoga Matsuo, Stefan Uhlich, Arun Venkitaraman, Andrea Bonetti, Chia-Yu Hsieh, Ali Momeni, Lukas Mauch, Augusto Capone, Eisaku Ohbuchi, and Lorenzo Servadei. 2025. Schemato – an llm for netlist-to-schematic conversion. *Preprint*, arXiv:2411.13899.

Lennart Meincke, Ethan Mollick, Lilach Mollick, and Dan Shapiro. 2025. Prompting science report 2: The decreasing value of chain of thought in prompting. *Preprint*, arXiv:2506.07142.

Jingyu Pan, Guanglei Zhou, Chen-Chia Chang, Isaac Jacobson, Jiang Hu, and Yiran Chen. 2025. A survey of research in large language models for electronic design automation. *Preprint*, arXiv:2501.09655.

Mirco Schönfeld and Jürgen Pfeffer. 2019. *Fruchterman/Reingold (1991): Graph Drawing by Force-Directed Placement*, pages 217–220. Springer Fachmedien Wiesbaden, Wiesbaden.

Oshayer Siddique, JM Alam, Md Jobayer Rahman Rafy, Syed Rifat Raiyan, Hasan Mahmud, and Md Kamrul Hasan. 2025. Physicseval: Inference-time techniques to improve the reasoning proficiency of large language models on physics problems. *arXiv preprint arXiv:2508.00079*.

Samaneh Talebzadeh and 1 others. 2021. Machine learning in eda: A survey. *ACM Transactions on Design Automation of Electronic Systems*.

Shailja Thakur and 1 others. 2023. Verilogeval: Evaluating large language models for verilog code generation. *arXiv preprint arXiv:2303.05461*.

Ziyu Wan, Yunxiang LI, Xiaoyu Wen, Yan Song, Hanjing Wang, Linyi Yang, Mark Schmidt, Jun Wang, Weinan Zhang, Shuyue Hu, and Ying Wen. 2025. ReMA: Learning to meta-think for LLMs with multi-agent reinforcement learning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models. *Preprint*, arXiv:2203.11171.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. Chain-of-thought prompting elicits reasoning in large language models. *Preprint*, arXiv:2201.11903.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Kailiang Ye, Qingyu Yang, Zheng Lu, Heng Yu, Tianxiang Cui, Ruibin Bai, and Linlin Shen. 2025. Llm4netlist: Llm-enabled step-based netlist generation from natural language description. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*.

Tao Zhang, Kehui Yao, Luyi Ma, Jiao Chen, Reza Yousefi Maragheh, Kai Zhao, Jianpeng Xu, Evren Korpeoglu, Sushant Kumar, and Kannan Achan. 2025a. No-human in the loop: Agentic evaluation at scale for recommendation. *Preprint*, arXiv:2511.03051.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. 2025b. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *Preprint*, arXiv:2506.05176.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *Preprint*, arXiv:2306.05685.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, and Ed Chi. 2023. Least-to-most prompting enables complex reasoning in large language models. *Preprint*, arXiv:2205.10625.

A Appendix

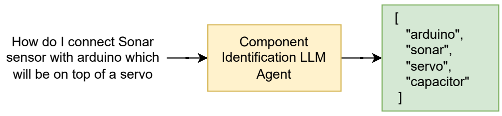

Figure 3: Input output data of component identification agent

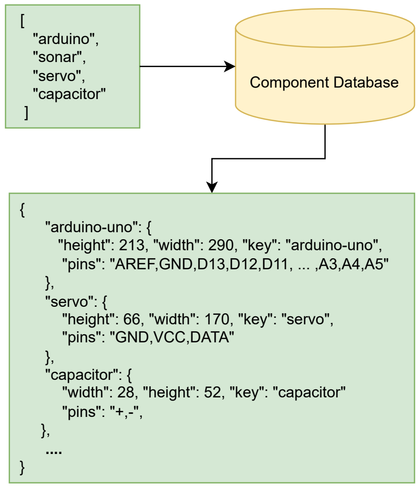

Figure 4: Components pin out information is retrieved by using the component's name from the database

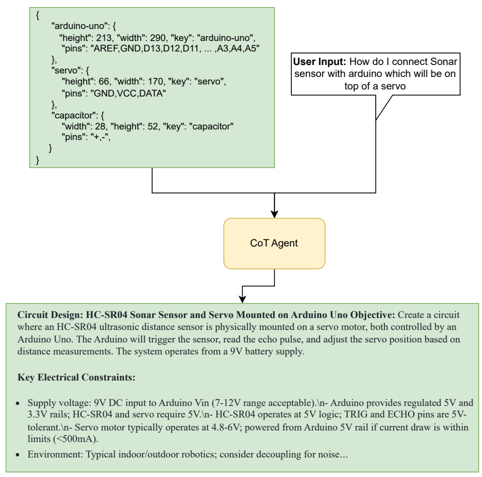

Figure 5: Workflow of the chain of thought agent, it takes the user's prompt and the retrieved components list and generates a reasoning for building the circuit

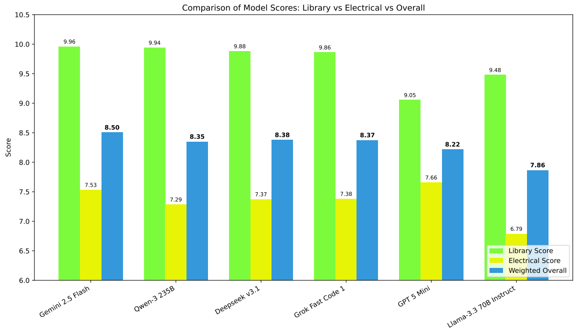

Figure 8: Stacked bar chart showing the average scores for Library Compliance, Electrical Logicality, and Overall score across all LLMs.

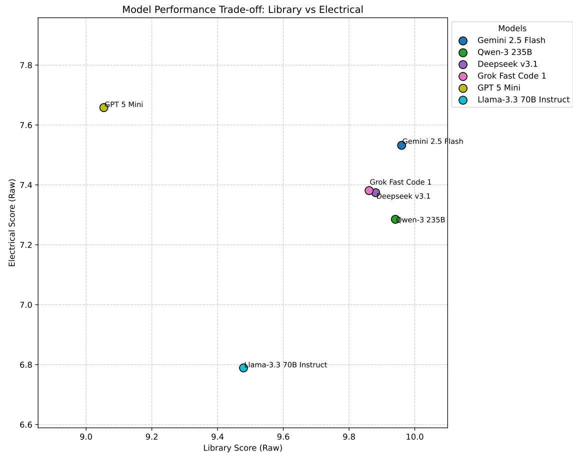

Figure 9: Scatter plot chart illustrating normalized average scores per dimension for all LLMs. This visualization highlights the relative strengths and weaknesses of each model in Library Compliance and Electrical Logic.

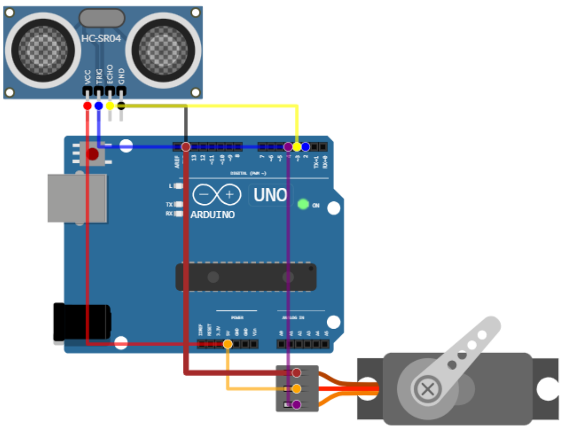

Figure 6: Circuit Schematic generated by the visualization engine

```
CircuitJson
  version : Number
  author  : String
  parts : List of Part
  connections : List of Connection

Part
  type   : String    // e.g. "arduino-uno"
  id     : String    // unique instance ID
  top    : Number    // Y-coordinate
  left   : Number    // X-coordinate
  attrs : Map<String, Any>
  rotate: Number (optional)

Connection
  startPin: String   // e.g. "arduino:5V"
  endPin  : String   // e.g. "l298n:5V"
  color   : String   // wire color
  route   : List<String>  // routing
    instructions
```

Listing 1: CircuitJSON Schema Definition
The top, left, rotate are for placement of components, attrs is a flexible map for additional information

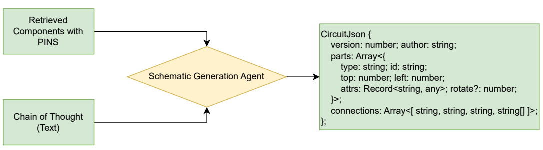

Figure 7: The circuit schematic generator agent takes the chain of thoughts along with a components list to create a high-level JSON representation of the circuit.

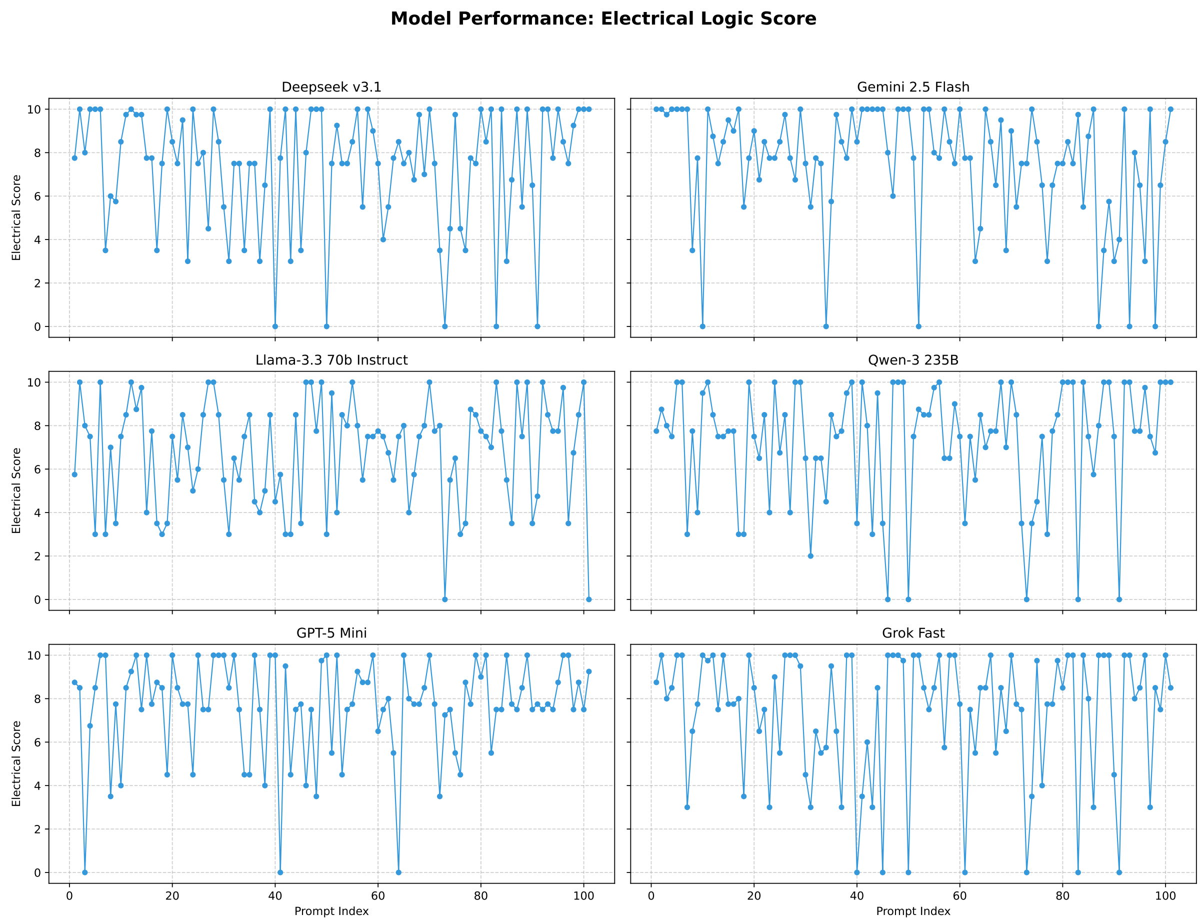

Figure 10: The matrix plot chart illustrates the performance of six LLMs with regard to the electrical logic score (S_{logic}) for each prompt.

Figure 11: The matrix plot chart illustrates the performance of six LLMs with regard to the library compliance logic score (S_{comp}) for each prompt.