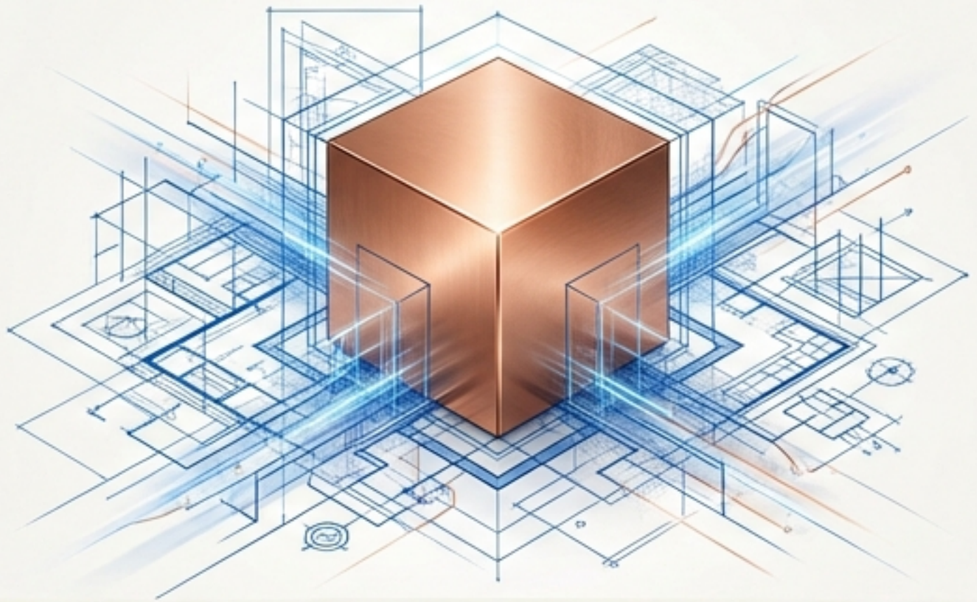
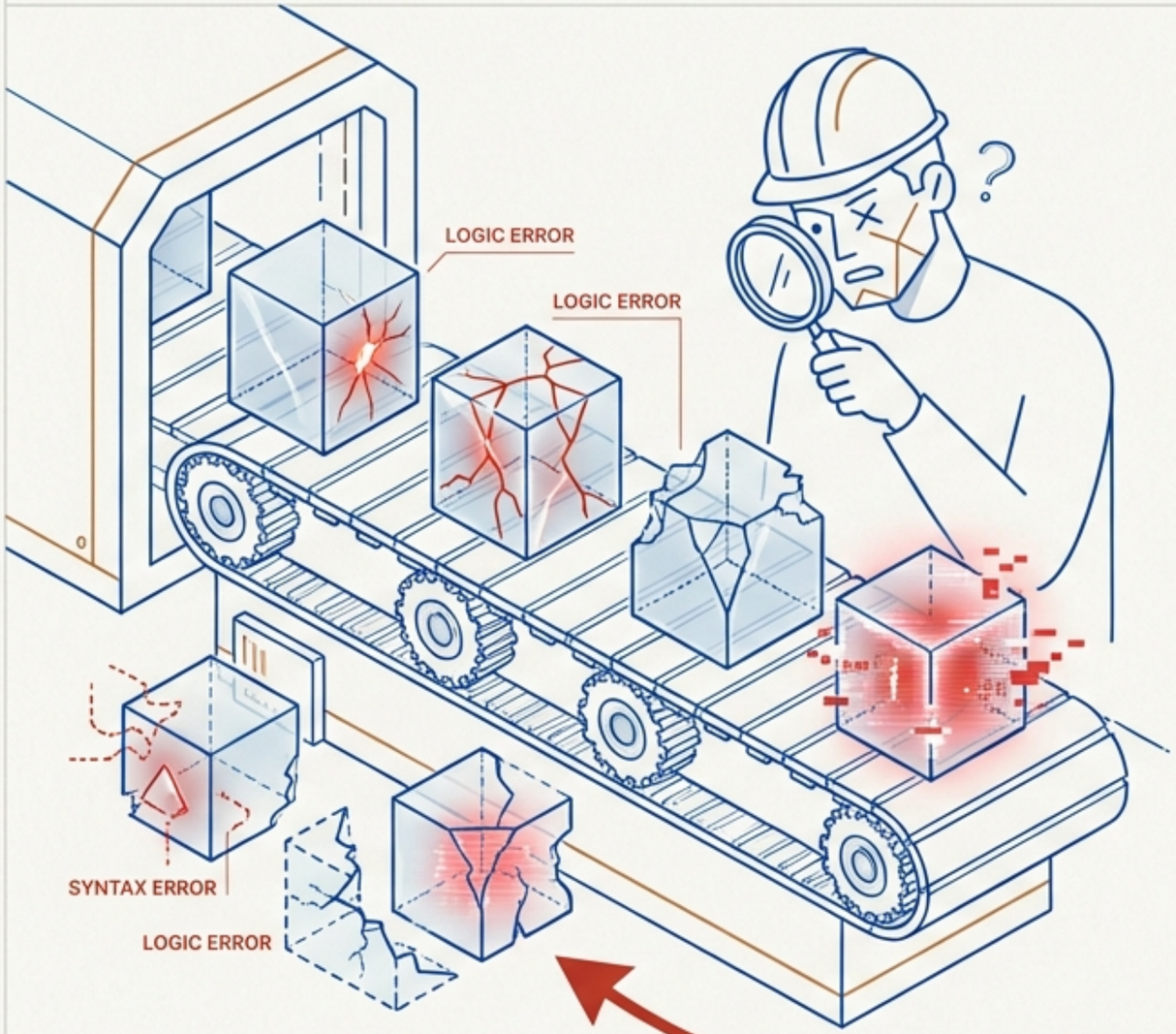
Veritas: A GenAI Framework for Correct-by-Construction Hardware Design



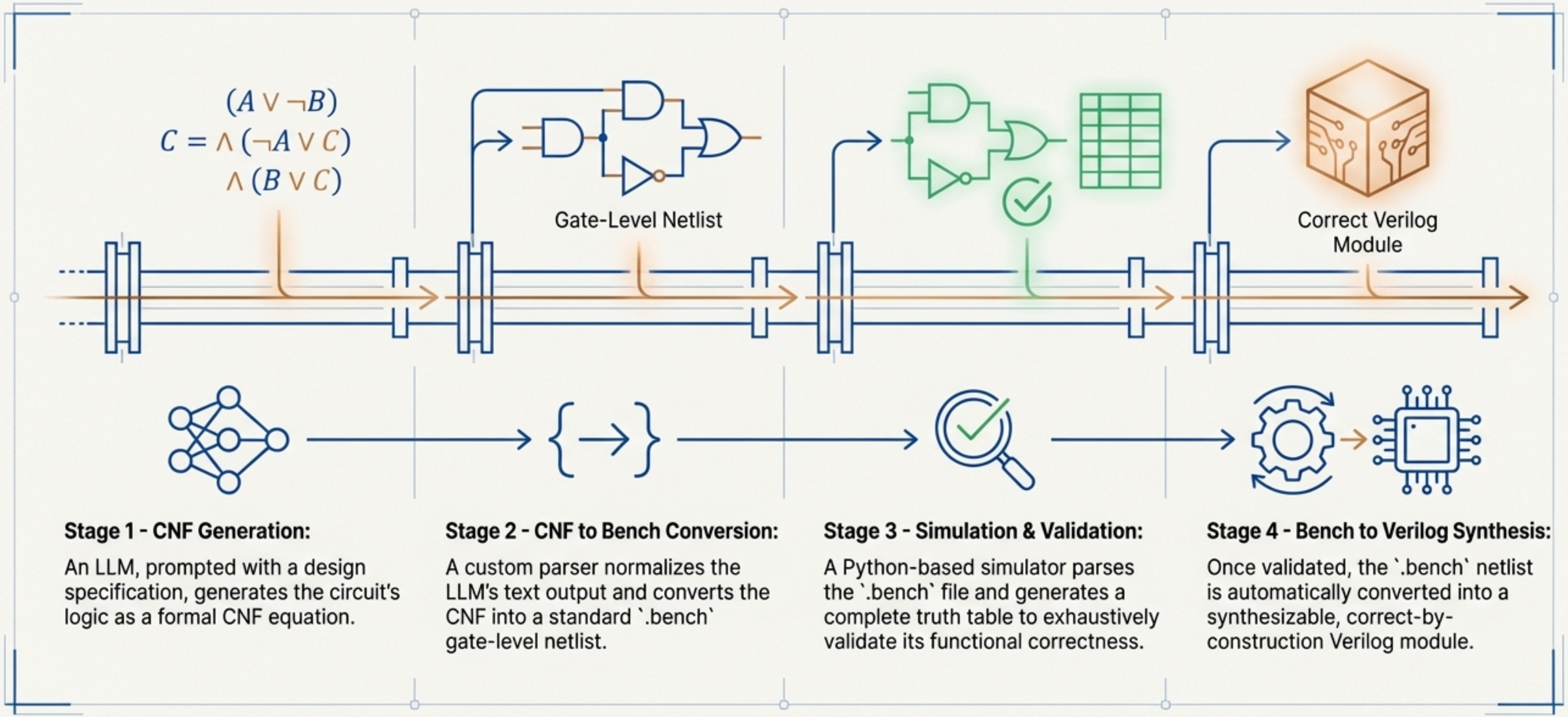
- **Objective:** Create and evaluate a novel workflow that uses a Large Language Model (LLM) to generate provably correct hardware logic.
- **Core Concept:** Shift the hardware design paradigm from "generate-then-test" to "correct-by-construction" by integrating verification into the generation phase.
- **Key Abstraction:** Instead of generating Verilog directly, the LLM produces a formal logic representation called Conjunctive Normal Form (CNF).
- **Core Assumption:** A logically correct CNF can always be converted into a functionally correct and test-free Verilog module.
- **Scope:** Tested on designs from basic logic gates to multi-bit adders and a complex 4-bit ALU.

The Problem: GenAI-Generated Verilog is Unreliable



- **Current State-of-the-Art:** LLMs can generate Verilog code, but it is frequently flawed and untrustworthy.
- **Common Failure Modes:** Generated code often contains both simple syntax errors and, more critically, incorrect logic.
- **The Human Bottleneck:** Verifying and correcting these AI-generated flaws requires significant time and supervision from expert hardware engineers.
- **An Inefficient Cycle:** The standard “generate-then-test” workflow relies on exhaustive, post-generation bug fixing, which is a slow and inefficient process.

The Veritas Workflow: A Four-Stage Correctness Pipeline



Stage 1 - CNF Generation:

An LLM, prompted with a design specification, generates the circuit's logic as a formal CNF equation.

Stage 2 - CNF to Bench Conversion:

A custom parser normalizes the LLM's text output and converts the CNF into a standard `.bench` gate-level netlist.

Stage 3 - Simulation & Validation:

A Python-based simulator parses the `.bench` file and generates a complete truth table to exhaustively validate its functional correctness.

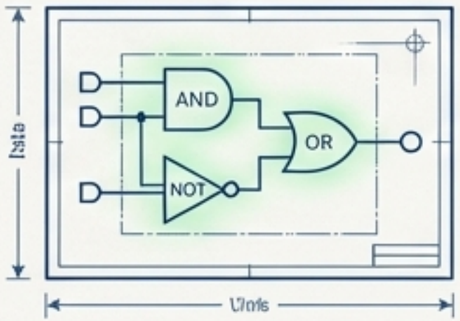
Stage 4 - Bench to Verilog Synthesis:

Once validated, the `.bench` netlist is automatically converted into a synthesizable, correct-by-construction Verilog module.

Methodology: Scaling Complexity with Incremental Prompting

Building Block Approach: The framework operates on the principle of building complex circuits from simpler, validated components.

1-Bit Adder CNF

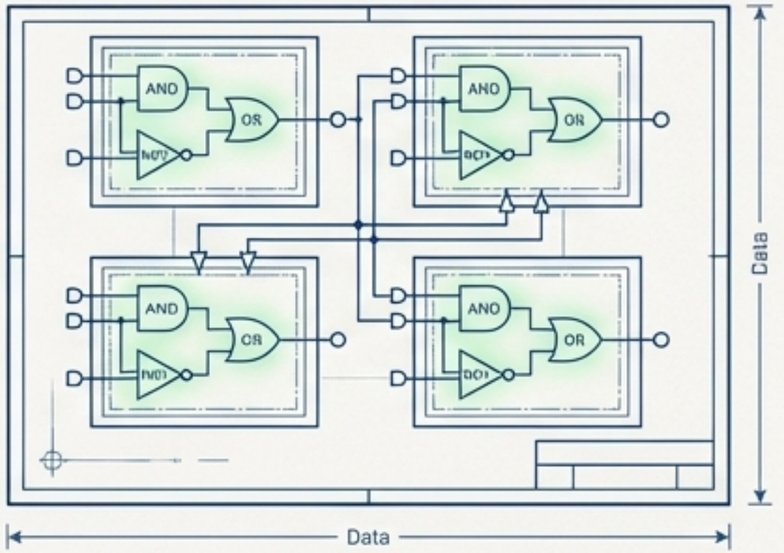


Base Case Generation: The process starts by prompting the LLM for the CNF of the simplest version of a design (e.g., a 1-bit adder).

Context

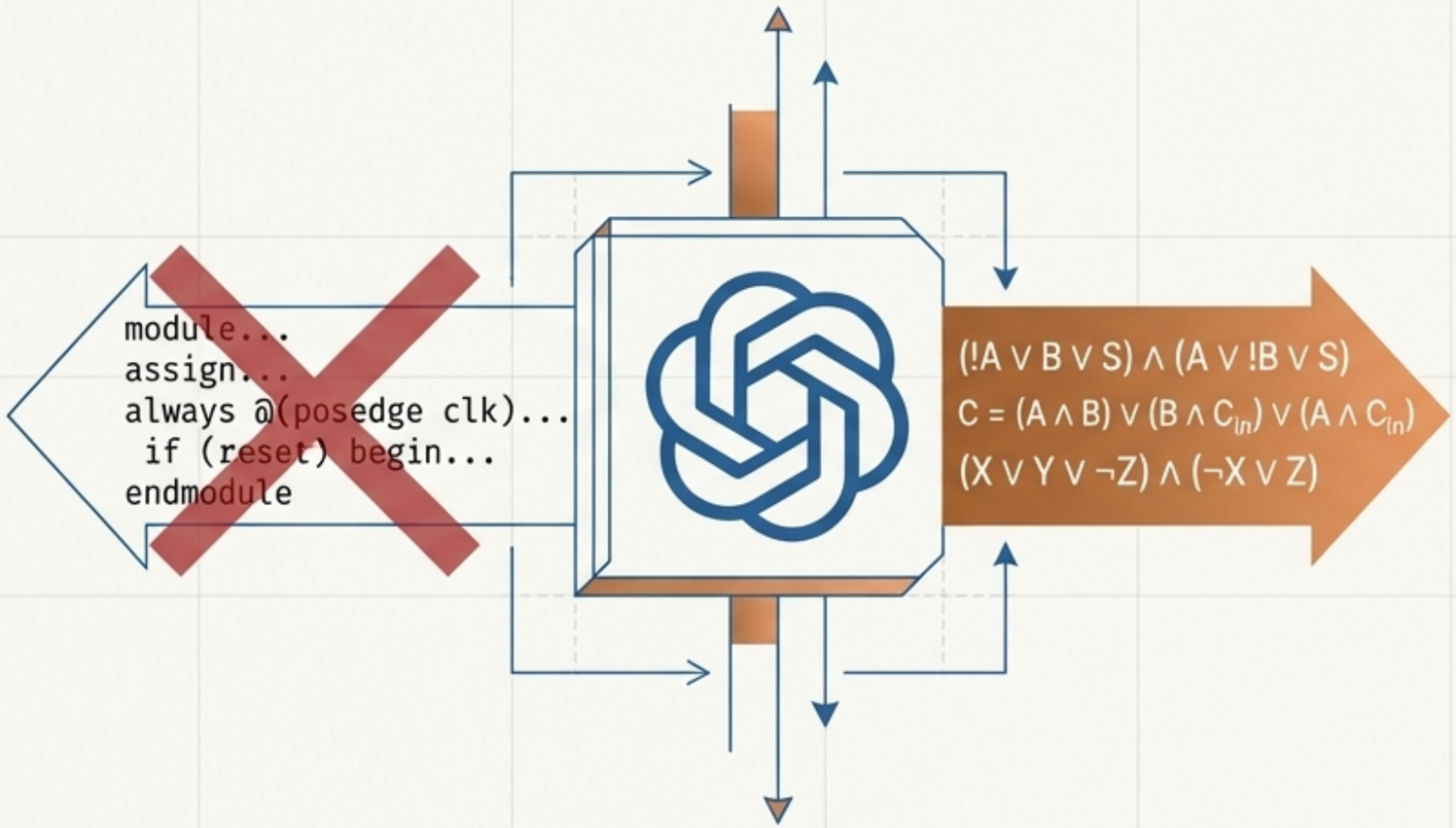
In-Context Learning: The correctly generated CNF for the simple case is then included as a reference in the context of the next prompt.

4-Bit Adder CNF



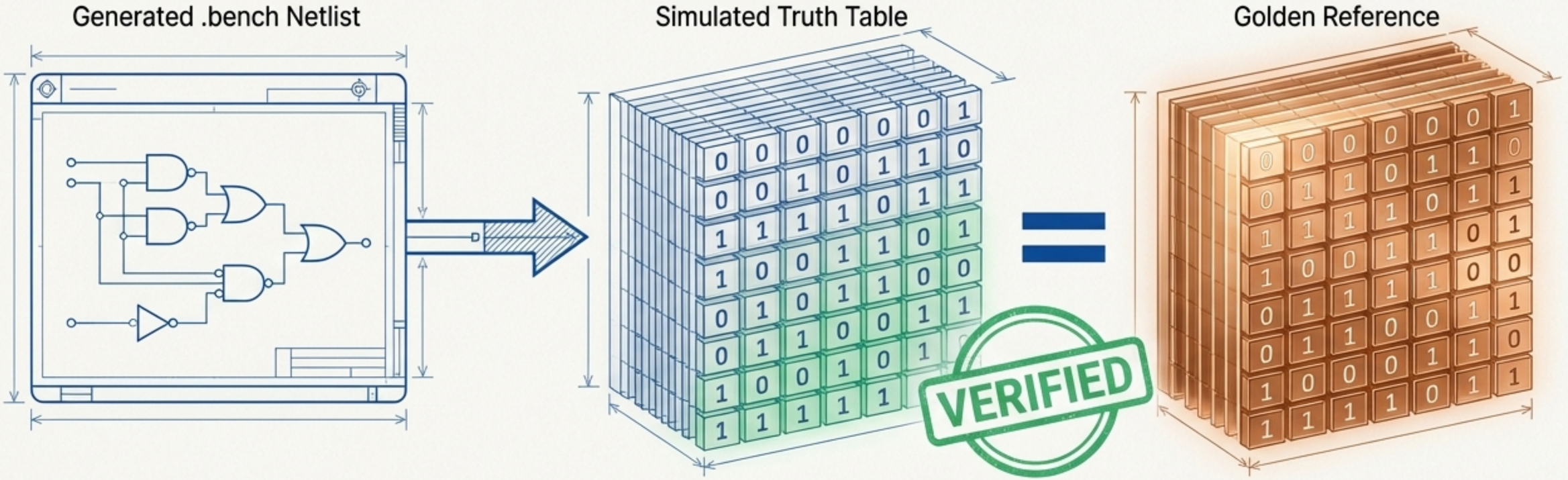
Iterative Scaling: This technique allows the LLM to learn from the validated base case and use it as a pattern to generate larger and more complex versions.

The Role of GenAI: A Formal Logic Synthesizer



- **Model Utilized:** The lab employs the `gpt-5-mini-2025-08-07` model accessed via the OpenAI API.
- **System Prompt:** The LLM is primed with a specific role: "You are an expert in logic synthesis and digital circuits."
- **Task Abstraction:** The LLM's primary task is abstracted away from language syntax (Verilog) and focused on generating pure, formal logic in CNF.
- **Core Function:** The framework leverages the LLM's capacity to understand a simple, validated context and extrapolate that pattern to produce more complex logic.

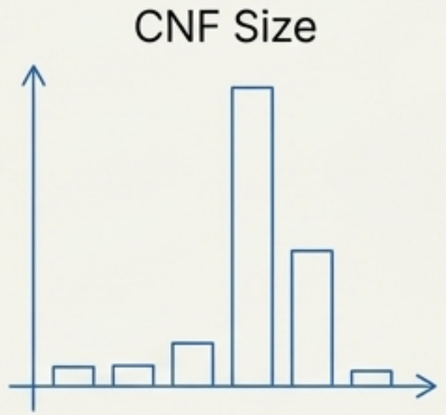
Verification: Exhaustive Truth Table Simulation

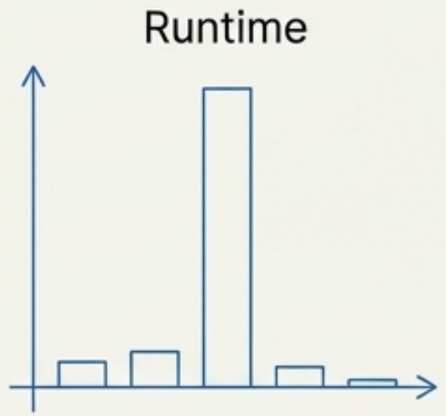


- **Automated Circuit Simulation:** A dedicated Python simulator parses the generated `.bench` netlist file to model the circuit's behavior.
- **Complete Functional Testing:** The simulator iterates through every possible combination of input vectors to generate a complete and exhaustive truth table.
- **Golden Reference Comparison:** The simulated truth table is programmatically compared against a pre-defined or dynamically generated 'golden' truth table.
- **Success Criterion:** A design is only considered 'verified' if its simulated output matches the expected output for 100% of all possible input combinations.

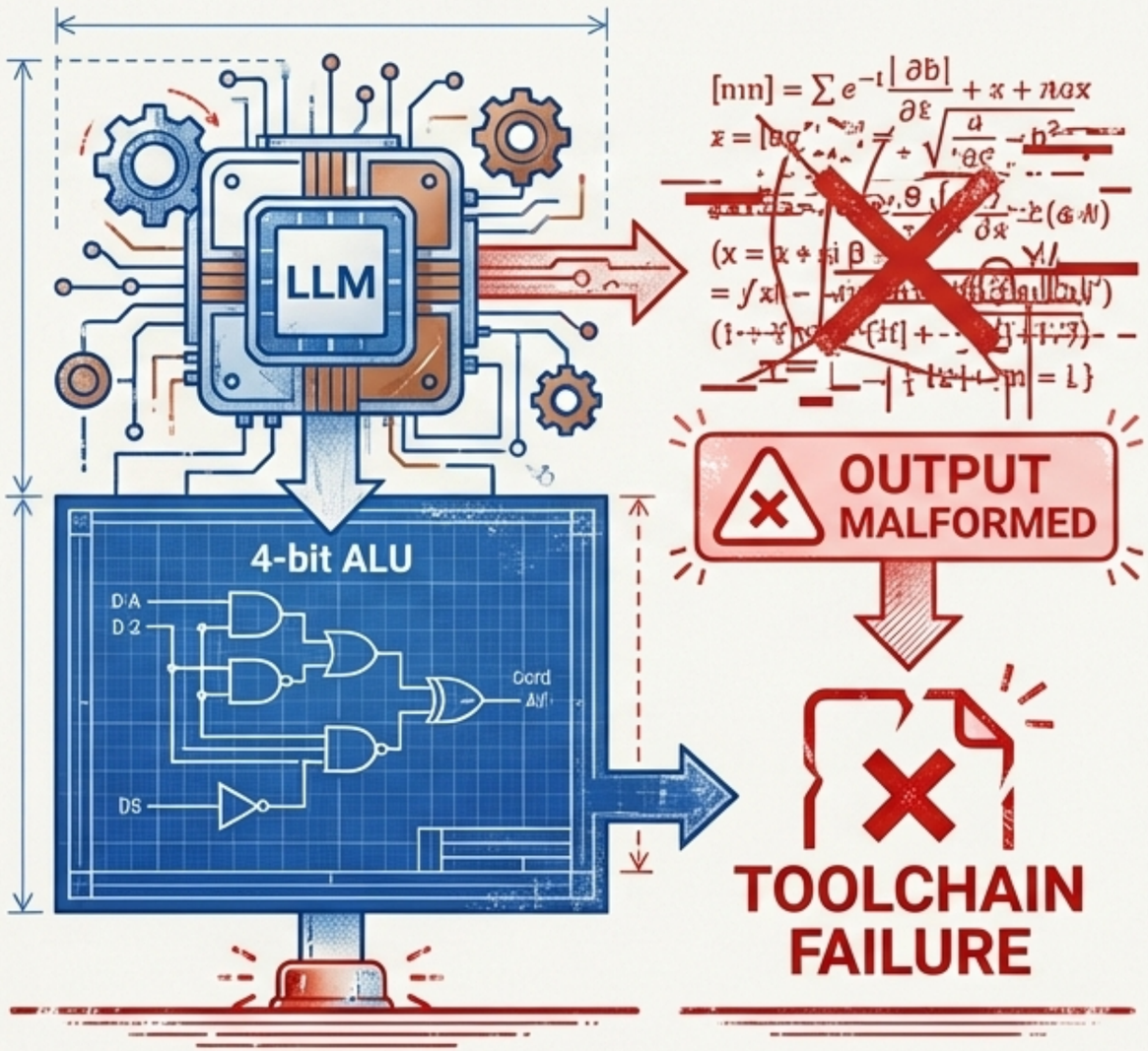
Key Results: Successes and a Critical Failure

Design Name	Verified	CNF Size (chars)	Runtime (ms)
AND Gate	✅	39	49
OR Gate	✅	38	18
XOR Gate	✅	63	19
NOT Gate	✅	23	14
1-bit Full Adder	✅	302	18
4-bit Ripple-Carry Adder	✅	616	2,589
2:1 Multiplexer	❌	63	5



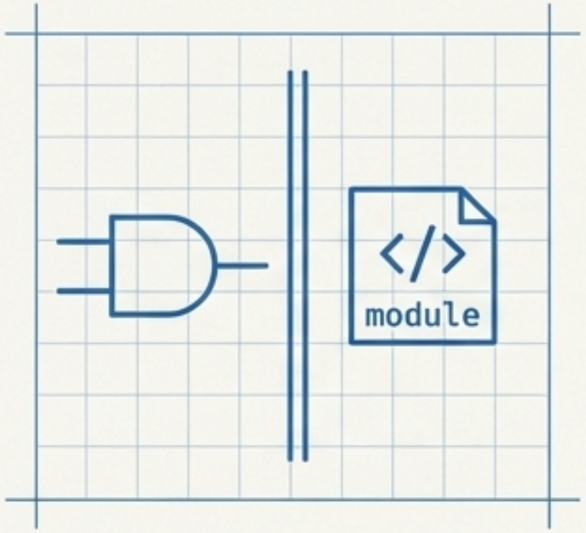


Limitations Exposed: The 4-Bit ALU Experiment



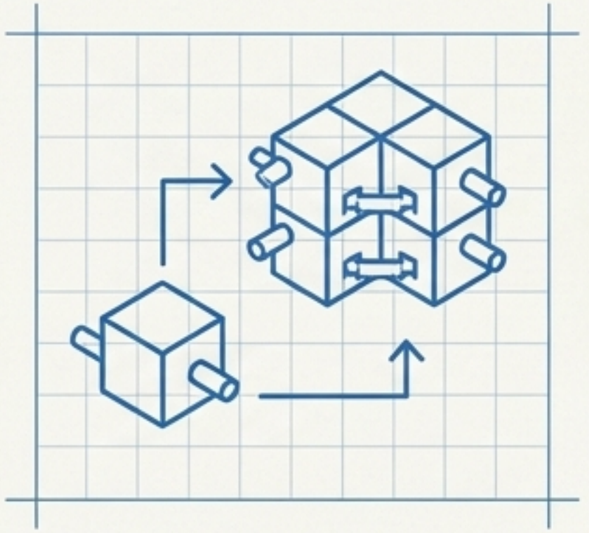
- **LLM Output Instability:** The LLM struggled to generate a clean, parsable CNF for the 4-bit ALU, producing malformed and unusable output.
- **Required Manual Intervention:** Due to the LLM's failure, a `.bench` file for the ALU had to be created manually to continue the workflow.
- **End-to-End Toolchain Failure:** The complete end-to-end process, even with the manual `.bench` file, ultimately failed for the full ALU and its black-box variants.
- **Black-Boxing Challenges:** The attempt to integrate black-box modules revealed significant challenges that the tooling could not handle.

Lessons Learned from the Veritas Experiment



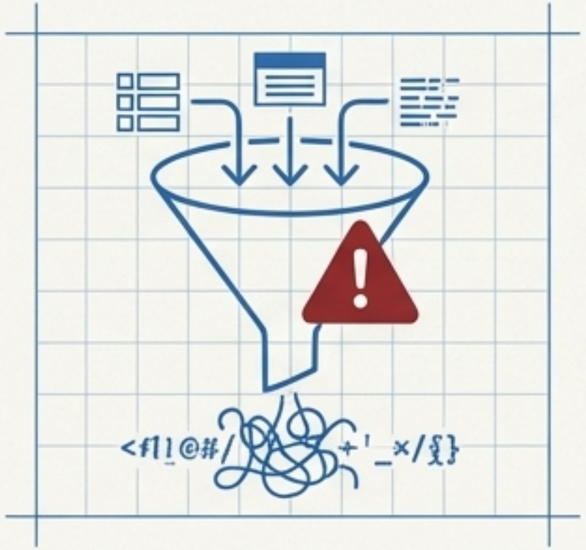
CNF is a Powerful Intermediate

Using a formal representation like CNF effectively separates the complex task of logic creation from the nuances of HDL syntax generation.



Incremental Prompting Works

Building logic iteratively from smaller, validated blocks is a highly effective strategy for scaling LLM-based generation.



Parsing is a Critical Point of Failure

The toolchain for parsing unstructured LLM text into structured formats like `.bench` is the most fragile part of the system.

'Correct-by-Construction' Requires Proof

The verification failure of the 2:1 multiplexer proves that the goal is not automatically achieved and that rigorous, exhaustive validation remains essential.

Conclusion: A Promising but Immature Framework

- **Proof-of-Concept Success:** The Veritas lab successfully demonstrated a viable proof-of-concept for a GenAI-driven, correct-by-construction hardware design flow.
- **Key Achievement:** The framework fully automated the creation of verified logic for several common digital circuits, validating the potential of the approach.
- **Primary Hurdles:** The main challenges are the reliability of LLM output and the scalability of the toolchain, as shown by the failures with the MUX and ALU.
- **Immediate Next Step:** Investigate the 2:1 multiplexer failure to determine the root cause: an error in the LLM's generated CNF, the toolchain, or the expected truth table.
- **Future Work:** Explore formal equivalence checking as a more efficient alternative to truth table simulation, enabling verification of more complex designs.