From Natural Language to a 1.55 GHz Verified GDS on ASAP7 Using Open-Source EDA

Kai-Chieh Hsu
kai.chieh.hsu@gmail.com
Independent Researcher
Fremont, CA, USA

Abstract

We carry an RV32I processor end-to-end — RTL generation, verification, and physical design — using an LLM agent on a single workstation, on a consumer subscription with public model access. Four runs are reported. A single-cycle RV32I and a 5-stage forwarding pipeline are taken to DRC-clean GDSII on SkyWater 130 nm at 100 MHz. The 5-stage RTL is then retargeted to ASAP7, closing first-try at 676 ps (1.50 GHz); a tighter SDC sweep with cell sizing alone reaches post-route fmax of 1.55 GHz. All four runs preserve a three-layer verification chain (ISA-grounded golden-trace anchor on the single-cycle reference, pipelined commit-trace alignment for downstream designs, and gate-level simulation on each post-route netlist), passing 40/40 standardized `riscv-tests` at every layer. Total cost: $9.80 in API-equivalent token usage on a Claude Max 20× $200/month subscription, in 2 h 34 min of wall-clock time across the four runs. Concurrent work [9] reaches a comparable closure target on cloud infrastructure with undisclosed frontier models. We show the same is reachable with public Sonnet through the standard Claude Code CLI, lowering the access barrier for community design-space exploration.

Keywords

Large language models, agentic AI, RTL generation, open-source EDA, RISC-V, ASAP7, physical design, reproducibility

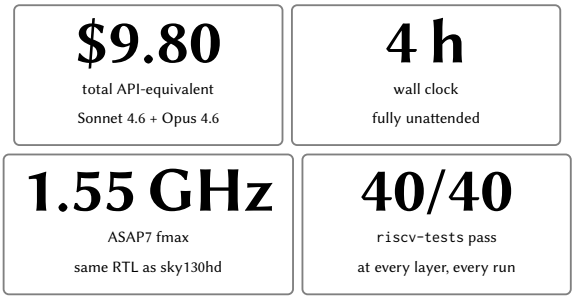

Figure 1: Results at a glance. Four LLM-generated RV32I runs — single-cycle on sky130hd (Run A), 5-stage forwarding on sky130hd (Run B), the same RTL retargeted unmodified to ASAP7 closing first-try at 1.50 GHz (Run C), and a tighter ASAP7 SDC sweep showing 1.55 GHz post-route fmax (Run C-640). All four runs share a single three-layer verification chain: Spike ISS anchor on Run A, pipelined commit-trace alignment for Runs B / C / C-640 against Run A's golden trace, and gate-level simulation on every post-route netlist. Total cost $9.80 in API-equivalent token usage, charged against a $200/month Claude Max 20× consumer subscription via the public Claude Code CLI [2]. No human-written RTL, testbench infrastructure, or EDA scripts.

1 Introduction

LLMs can generate synthesizable HDL from natural-language specifications [3, 12, 18], and open-source EDA tools have reached fabrication-ready maturity [1, 7, 21]. Together they raise a sharper question (Figure 2): *is the access barrier to demonstrating an end-to-end NL-to-GDS flow now low enough that an individual researcher can reproduce it on a consumer subscription?*

Concurrent work has shown autonomous LLM agents can carry RV32I-class processors to DRC-clean GDSII on academic 7 nm PDKs at frequencies above 1 GHz [9], but does so on cloud infrastructure with undisclosed frontier models and tens of billions of tokens. Whether the same flow is reachable for one researcher with a workstation, a public-API model, and a metered subscription has been an open question. We answer it directly.

We report four end-to-end LLM-generated runs sharing a single three-layer verification chain. **Run A** generates a single-cycle RV32I from natural language, anchored to the Spike ISS [16] by a constant instruction-count delta across 40 standardized `riscv-tests` [15]. **Run B** generates a 5-stage forwarding pipeline whose committed PC trace matches Run A's golden reference at every retirement. Both Run A and Run B are taken to DRC-clean GDSII on SkyWater 130 nm at a 100 MHz target. **Run C** re-targets Run B's verified RTL, unmodified, to the ASAP7 PDK [4], closing on the first iteration at 676 ps (1.50 GHz) with WNS +9.17 ps and zero DRC violations. **Run C-640** sweeps the same RTL to a tighter 640 ps SDC target with cell sizing alone and no RTL changes; it does not meet the 640 ps target (setup WNS −6.65 ps) but ORFS post-route analysis reports a minimum achievable period of 646.65 ps, giving 1.55 GHz as the empirical fmax of this RTL on ASAP7 without architectural changes.

Total cost of the four runs is $9.80 in API-equivalent token usage, charged against a single Claude Max 20× $200/month consumer subscription via the public Claude Code CLI [2]. Wall-clock time is 2 h 34 min on a single workstation (Intel i7-12700K, 32 GB RAM, no GPU acceleration required for the EDA flow).

The contributions are:

(1) **Consumer-subscription reproduction.** We carry an end-to-end NL-to-1.55 GHz-GDS flow on public Sonnet via the standard Claude Code CLI, with no API-key access required, no fine-tuned models, no cloud infrastructure, and no proprietary tooling. The cost upper bound is the $200/month subscription fee.

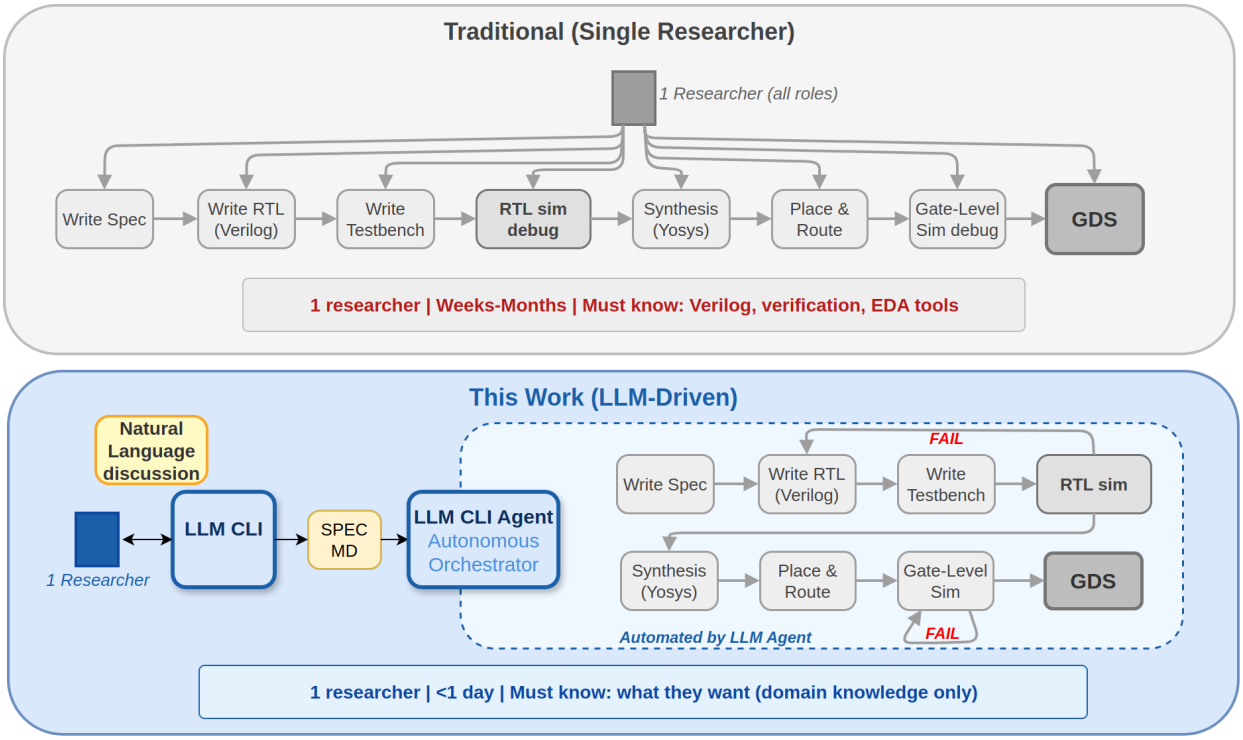

Figure 2: Traditional single-researcher chip design flow (top) vs. the LLM-driven pipeline (bottom). The researcher provides a natural-language specification document; the LLM agent then autonomously executes the entire RTL-to-GDS pipeline through the standard Claude Code CLI in headless mode, with no further human input.

(2) **Per-run cost telemetry.** The launcher captures every LLM call's token usage via `–output-format stream-json`, deduplicating across main and sub-agent JSONL streams. Reported costs are computed from raw token counts, not estimates.
(3) **Cross-PDK methodology validation.** The same verified RTL closes at 100 MHz on sky130hd and 1.50 GHz on ASAP7, with the three-layer verification chain (Spike-anchor, commit-trace alignment, GLS) preserved on both PDKs. Run C-640 also reports 1.55 GHz post-route fmax reachable without RTL changes.
(4) **Open call.** The methodology generalizes naturally: alternative ISA scopes (RV32IM, RV32IMC), microarchitectural variants, and PDK choices are follow-ups whose cost can now be estimated in advance, lowering the barrier to community exploration of the design space.

2 Related Work

2.1 LLMs for End-to-End Hardware Design

Most LLM-for-hardware work focuses on RTL generation in isolation [11–13] or on downstream EDA automation given existing RTL [6, 8]. ChipChat [3] demonstrated a small tapeout but required extensive human guidance. ChipNeMo [10] and similar fine-tuned models [18] pursue domain adaptation rather than agent-driven flow execution.

Closer to this work, two recent efforts close the natural-language-to-layout loop. AiEDA [14] and NL2GDS [5] demonstrate end-to-end flows for application-specific designs and benchmark circuits. Most directly comparable, Verkor's Design Conductor [9] carries a 219-word requirements document to a 1.48 GHz RV32I-Zmmul 5-stage pipeline on ASAP7 in approximately 12 hours, consuming reportedly tens of billions of tokens on cloud infrastructure with undisclosed frontier models. Our work shares the verification anchor (Spike ISS) but differs in two dimensions: (i) it uses public Sonnet 4.6 through the standard Claude Code CLI rather than a frontier proprietary model on managed infrastructure, and (ii) the entire suite consumes $9.80 in API-equivalent tokens, roughly four orders of magnitude less than the figures reported in [9]. The contribution is access, not capability: this paper does not claim a methodology superior to [9] but shows the same closure target is reachable for one researcher on a consumer subscription.

2.2 Open-Source EDA and Verification

OpenROAD [1] and ORFS [19] provide RTL-to-GDS flows on multiple PDKs including SkyWater 130 nm [7] and ASAP7 [4]. Spike [16] serves as the reference ISA simulator; `riscv-tests` [15] provides the standardized RV32I conformance suite used here. Verilator [17] handles RTL simulation and Icarus Verilog [20] the gate-level simulation. We invoke these tools through ORFS; flow times match prior characterizations.

3 Methodology

3.1 Pipeline Overview

Each run begins with a Markdown specification document (50–100 lines) describing the target microarchitecture, verification gates, ISA conformance constraints, and physical design target. The specification is the only human-authored input (Figure 3); everything else (RTL, testbenches, simulation harnesses, ORFS configuration, GLS scripts, and trace comparison logic) is generated autonomously by the LLM agent through the Claude Code CLI [2] in headless mode (`claude -print`).

Two LLM roles are used. An *architect* subagent (Opus 4.6) is invoked at most once per run for spec elaboration on pipelined designs (Run B onward). An *implementer* (Sonnet 4.6) drives the end-to-end flow: tool installation, RTL generation, simulation, debugging, ORFS execution, GLS, and trace verification. Subagents are also dispatched for context isolation on long-running operations (full ORFS, GLS sweeps, regression diffs). Stream-json output is captured to a per-run JSONL file; token usage is extracted post-hoc by deduplicating on `(request_id, message_id)` across main and subagent streams.

3.2 Three-Layer Verification Chain

Layer 1: ISA-grounded golden trace. Run A generates a deterministic single-cycle RV32I, the only design in the suite directly verified against an ISA-spec-grounded oracle. After running 40 `rv32ui-p-*` tests under both Spike [16] and Verilator, instruction counts are compared to confirm a constant delta (4,991 in this run) across all 40 tests. The constant delta corresponds to Spike's HTIF host-interface polling overhead, identical for every test. A constant delta certifies that Run A's committed-PC sequence equals the Spike committed-PC sequence on the architecturally visible instructions; that sequence is preserved as the golden reference for all subsequent runs.

Layer 2: Pipelined commit-trace alignment. Runs B, C, and C-640 each export `(valid_commit, commit_pc)` signals at every retirement and run the same 40-test workload. The committed PC sequence at each retirement is compared (directly, not transitively) against Run A's golden trace. Forwarding, stalling, and pipeline restructuring affect cycle-level timing but cannot change the architectural commit order; if any path through the pipeline produces a divergent committed PC, the diff catches it. All three downstream runs match the golden reference on all 40 tests.

Layer 3: Gate-level simulation. The post-route gate-level netlist for each run is simulated with the foundry-supplied functional Verilog cell models (the `sky130_fd_sc_hd` standard-cell library for Runs A and B; `asap7sc7p5t_RVT` for Runs C and C-640), under `-DFUNCTIONAL` and `-DUNIT_DELAY` compile flags, on the same 40 tests. The GLS committed-PC trace must match the RTL trace exactly; any mismatch indicates synthesis or place-and-route divergence. All four runs pass GLS at 40/40 with traces identical to RTL.

The trust chain is therefore: Spike-as-spec → Run A at Layer 1 → Runs B, C, C-640 RTL at Layer 2 → Runs B, C, C-640 post-route at Layer 3. Run A is the unique trust anchor; removing it dissolves the chain.

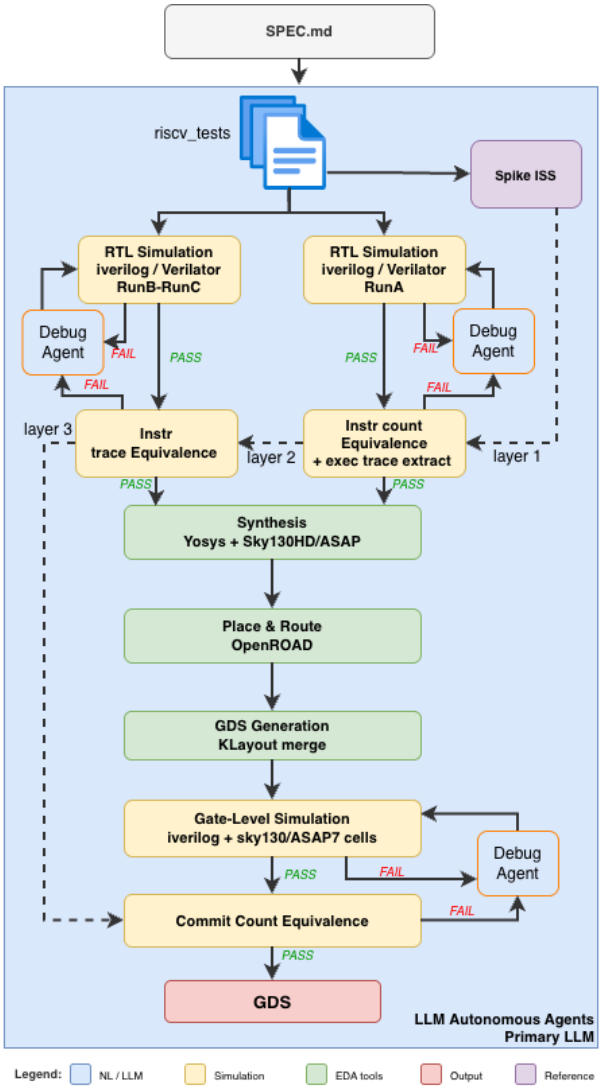

Figure 3: Autonomous execution phase: from a Markdown specification document to DRC-clean GDSII with functionally-verified RTL. All steps after the specification are fully autonomous — RTL generation, simulation, ORFS run, GLS, and trace verification are driven by the LLM agent through the standard Claude Code CLI in headless mode. ISA-grounded tests (`riscv-tests` via Spike) provide independent verification at both the RTL and gate levels.

3.3 Cost Capture

The launcher invokes `claude -print -output-format stream-json -verbose` per run, with `ANTHROPIC_API_KEY` unset to force OAuth authentication against the author's Claude Max subscription. The streaming JSONL is preserved; the per-run `~/.claude/projects/` session directory, which contains additional sub-agent JSONLs, is also kept and not deleted between stages (prior automated experiments deleted this directory between stages and lost the cost

telemetry). A 150-line Python tally script walks every JSONL, deduplicates by request and message ID, applies published Sonnet 4.6 / Opus 4.6 token rates, and reports per-model and per-run cost. Reported costs are exactly what the API would have charged at public rates, regardless of actual subscription billing.

4 Experimental Setup

Hardware: Intel Core i7-12700K, 32 GB RAM, AVX-512 enabled, no GPU. Ubuntu 24.04. Docker 28 with the `openroad/orfs:latest` image (OpenROAD 26Q1, Yosys 0.63+post, KLayout 0.30.3) for both sky130hd and ASAP7 flows.

LLM access: Claude Max 20× subscription (US$200/month) via the standard `claude` CLI. No API key configured. No frontier-model access. Model: Sonnet 4.6 by default; Opus 4.6 invoked once per run for architect-stage spec elaboration.

Test workload: `rv32ui-p-*` from `riscv-tests` [15], 40 tests covering all RV32I integer and load/store instructions; `fence_i` (Zifencei) and `ma_data` (optional misaligned access) excluded by spec.

PDKs: sky130hd (SkyWater 130 nm) [7] for Run A and Run B at 100 MHz; ASAP7 (predictive 7 nm academic PDK) [4] for Run C at 676 ps and Run C-640 at 640 ps.

5 Results

5.1 Per-Run Outcomes

Table 1 summarizes the four runs. Verification (Layer 1 / Layer 2 / Layer 3) passes 40/40 on every applicable layer for every run. Layer 1 (Spike anchor) is reported only for Run A; downstream runs anchor through Run A's golden trace.

Run A. The agent generates a single-cycle RV32I from a ∼70-line specification, with mtvec/mepc/mcause/mhartid CSRs and `ECALL` mapped to `mcause=11`. After Verilator simulation reaches 40/40 PASS, the agent runs Spike on each test and confirms the instruction-count delta is 4,991 on all 40 tests — the Layer 1 anchor. ORFS targets sky130hd at 10 ns; final post-route WNS is +0.016 ns (a margin of 16 ps), meeting timing at 100.16 MHz with no headroom, as expected for the single-cycle datapath whose entire IF→ID→EX→MEM→WB chain must complete within one clock period. DRC=0. GLS with sky130 cell models passes 40/40 with traces identical to RTL.

Run B. The agent generates a 5-stage forwarding pipeline (stages IF, ID, EX, MEM, WB; full EX-to-EX and MEM-to-EX forwarding; 1-cycle load-use stall; branch resolved in EX with 2-cycle flush) as a monolithic 742-line module. The agent verifies 40/40 RTL via Verilator, then aligns committed-PC traces against Run A's golden reference: 40/40 IDENTICAL. ORFS produces a fresh GDSII at +1.927 ns slack; GLS at 40/40 with traces identical to RTL.

Run C. The agent copies Run B's RTL unchanged into a new working directory and re-targets ORFS to the ASAP7 PDK at a 676 ps SDC. The reference `designs/asap7/riscv32i` configuration is used as a template. ORFS closes on the first iteration: post-route WNS +9.17 ps, hold WNS +14.27 ps (no violations), DRC=0. The closure loop budget of eight iterations is unused; the agent makes no RTL edits (Figure 4). Layer 2 trace alignment against Run A: 40/40 IDENTICAL. Layer 3 ASAP7 GLS: 40/40 IDENTICAL.

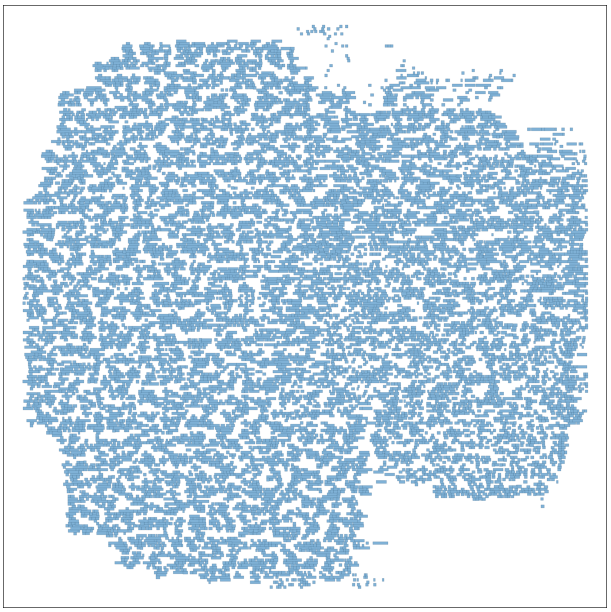

Figure 4: Run C post-route placement on ASAP7, routing layers omitted to expose cell density: 12,621 logic cells (1607 flip-flops) packed at 42% utilization across a 266 μm square die, 0 DRC violations after routing, fmax 1.50 GHz. RTL is byte-identical to Run B (sky130hd, 100 MHz target).

Run C-640. The agent re-runs ORFS on the same RTL with a tighter 640 ps SDC. Setup WNS post-route is −6.65 ps (27 violation paths concentrated on the forwarding-hazard signal: an EX/MEM destination-register comparison feeding a high-fanout buffer driving an OR-tree to the ALU input mux). Cell sizing alone cannot close the gap, and the run's no-RTL-change rule prevents edits. ORFS `report_clock_min_period` reports 646.65 ps; the empirical fmax of this RTL on ASAP7 with cell sizing alone is 1.55 GHz. Hold timing remains met (WNS +22.10 ps). Functional verification passes all three layers.

5.2 Cost Breakdown

Table 2 gives the per-run token breakdown extracted from the stream-json telemetry. Cache-read tokens (Sonnet 4.6 at $0.30 per million) dominate volume from repeated context replay across agent turns, but cost less per token than cache-write or output.

5.3 Wall-Clock Profile

End-to-end wall-clock times: Run A 66 min; Run B 46 min; Run C 30 min; Run C-640 12 min. Total elapsed time across the four runs is 2 h 34 min, with no human intervention between launch and completion. Wall clock is short for two reasons: ASAP7 ORFS turnaround is faster than sky130hd for an equivalent design (smaller cell sizes, fewer routing iterations), and Run C required no closure iterations.

Table 1: End-to-end results across four LLM-generated runs. Run B's RTL is reused unmodified for Run C and Run C-640 (no architectural changes; only PDK and SDC retargeting).

Run	Design	PDK	Target	Final period	Final fmax	WNS	DRC	L1/L2/L3	API $
A	Single-cycle RV32I	sky130hd	10.0 ns	9.984 ns	100.16 MHz	+0.016 ns	0	40/40, 40/40, 40/40	3.60
B	5-stage forward	sky130hd	10.0 ns	8.073 ns	124 MHz	+1.927 ns	0	–, 40/40, 40/40	2.59
C	B's RTL on ASAP7	asap7	676 ps	666.83 ps	1499 MHz	+9.17 ps	0	–, 40/40, 40/40	2.70
C-640	B's RTL on ASAP7	asap7	640 ps	646.65 ps*	1546 MHz*	−6.65 ps	0	–, 40/40, 40/40	0.91
Total									**9.80**

*Run C-640 did not meet its 640 ps SDC target (setup WNS −6.65 ps, 27 violation paths). The reported period (646.65 ps) is the post-route minimum reachable clock period from the ORFS `report_clock_min_period` command—the empirical fmax with cell sizing alone, no RTL changes. Hold timing met (WNS +22.10 ps, 0 violations).

Table 2: Per-run token usage and API-equivalent cost. Sonnet 4.6 rates: $3.00/M input, $3.00/M cache write, $0.30/M cache read, $15.00/M output. Opus 4.6 rates: $15.00/M input, $15.00/M cache write, $1.50/M cache read, $75.00/M output.

Run	Calls	Cache write	Cache read	Output	Cost ($)
A	124	181.7 K	10,007.0 K	3.4 K	3.60
B (Sonnet)	102	217.4 K	4,151.1 K	2.3 K	1.95
B (Opus)	5	36.9 K	42.2 K	0.2 K	0.63
C	121	106.6 K	7,714.9 K	4.5 K	2.70
C-640	54	60.4 K	2,341.3 K	1.7 K	0.91
Total	**406**	**603 K**	**24,256 K**	**12 K**	**9.80**

5.4 Headroom and Critical-Path Diagnosis

The Run C-640 timing report identifies the limiting setup path: EX/MEM destination-register flop, then an XOR/OR-tree forwarding-hazard comparison, into a high-fanout buffer (fanout 36 in Run C, 66 in Run C-640 after sizing), and finally an AO21 gate driving the ALU forwarding mux. Below 647 ps this path cannot be shortened by cell sizing alone; reaching tighter targets would require structural RTL changes (*e.g.*, replicating the EX/MEM destination register, distributing the comparison, or moving the comparison one stage earlier into ID). Such restructuring was out of scope: the paper deliberately preserves the LLM-generated RTL across PDK retargeting, separating the cost of the pipeline from the cost of closure-driven editing.

6 Open Call: Community Design-Space Exploration

The four runs reported here cover one slice of a much wider design space. Possible next experiments, whose cost can now be estimated in advance:

- **ISA scope.** RV32IM (multiplier as a closure constraint), RV32IMC (compressed-instruction decode latency in IF/ID), RV32E (reduced-register variants for area-constrained flows).
- **Microarchitecture variants.** Branch predictor families (gshare, tournament, perceptron); deeper pipelines; dual-issue; speculative load-store.
- **PDK frontier.** Other ORFS-supported platforms (Nangate45, GF180, IHP-SG13G2); side-by-side cost-vs-frequency curves across PDKs for the same RTL.
- **Verification depth.** Formal equivalence post-route; SDF-back-annotated GLS; coverage-driven test selection.
- **Spec-prompt ablations.** Alternative architect-prompt frameworks; structured vs. unstructured specs; multi-model adversarial spec critique.

The launcher is a ~40-line Bash script and a ~150-line Python tally; the spec MDs are 50–100 lines each. Adapting the workflow to a new design point is a half-hour engineering task, not a research program. A related submission is under peer review; specification documents and reproduction scripts will be released alongside it.

7 Limitations

Scope. The four runs cover only RV32I (no Zmmul, no M, no C). The choice was deliberate, to keep the verification chain simple and the cost minimal, but it limits direct comparison to systems carrying a multiplier or compressed-instruction support.

Single sample. Each run is N=1. Earlier ten-run reproducibility data (under separate review) shows mean 2.8 RTL bugs per run with all runs converging to DRC-clean GDSII, this paper does not repeat that characterization. Run-to-run variance in cost and wall time is not quantified.

ASAP7 is predictive. The 1.50 GHz / 1.55 GHz ASAP7 results are research-quality, not foundry tape-out targets. ASAP7 is a predictive 7 nm academic PDK; absolute frequency numbers would change on a real foundry node. What this paper claims is reproducibility of the methodology, not absolute fmax.

No formal equivalence. Layer 3 (GLS) verifies functional equivalence between RTL and post-route netlist on the 40-test workload, but does not constitute formal equivalence checking. Adding LEC is a natural follow-up.

Subscription dependence. The cost claim ($9.80 API-equivalent) reflects what the public Sonnet 4.6 API would have charged at published rates for the captured token usage. The runs were executed on a $200/month Claude Max 20× subscription for convenience; the same cumulative usage (around 24 M tokens and 400 LLM calls) sits well within the budget of a $20/month Claude Pro account, though Pro's tighter 5-hour rate limits may extend wall-clock time. The launcher script accepts an optional inter-run sleep (`WAIT_BETWEEN_RUNS`) to space runs across rolling-window boundaries; this is defensive and was not exercised here. The methodology itself does not depend on the specific provider tier.

8 Conclusion

Four LLM-generated RV32I runs (a single-cycle reference, a 5-stage forwarding pipeline on sky130hd, the same RTL retargeted to ASAP7 at 1.50 GHz, and a tighter SDC sweep showing 1.55 GHz

post-route fmax) are carried end-to-end from natural language to DRC-clean GDSII, preserving a three-layer verification chain anchored to the Spike ISS. Total cost is $9.80 in API-equivalent token usage, charged against a $200/month consumer subscription, in 2 h 34 min of wall-clock time on a single workstation. The access barrier to an end-to-end NL-to-GDS flow is no longer cloud infrastructure or frontier-model access: a public-API model through the standard CLI suffices. Reporting the cost from raw token telemetry rather than estimates lets others price similar experiments before committing to them.

Generative AI Disclosure

All RTL, testbench infrastructure, simulation harnesses, ORFS configurations, and trace-comparison logic in this work were generated by Claude Sonnet 4.6 (with one architect-stage invocation of Claude Opus 4.6 in Run B) through the Claude Code CLI in headless mode. The author wrote the four specification Markdown documents (50–100 lines each, the only human-authored input to the generation pipeline), the launcher Bash script, the token tally Python script, and this paper. The paper text was written by the author with light editorial assistance from Claude (revision suggestions, phrasing). All numerical results (fmax, slack, cost, wall time, verification pass rates) come from ORFS reports and stream-json telemetry; none were generated by an LLM.

References

[1] Tutu Ajayi, David Blaauw, Timothy Chan, Chien-Yi Cheng, Vidya A. Chhabria, Dae Hyun Choo, Matteo Coltella, Stefan Dobre, Ronald Dreslinski, et al. 2019. OpenROAD: An Open-Source, Autonomous Digital Layout Generation Flow. In *Government Microcircuit Applications & Critical Technology Conference*. https://doi.org/10.1145/3316781.3326334

[2] Anthropic. 2026. Claude Code Command-Line Interface. Online; https://docs.anthropic.com/en/docs/claude-code.

[3] Jason Blocklove, Siddharth Garg, Ramesh Karri, and Hammond Pearce. 2023. Chip-Chat: Challenges and Opportunities in Conversational Hardware Design. In *Proceedings of the 5th Workshop on Machine Learning for CAD (MLCAD)*. https://arxiv.org/abs/2305.13243

[4] Lawrence T. Clark, Vinay Vashishtha, Lucian Shifren, Aditya Gujja, Saurabh Sinha, Brian Cline, Chandarasekaran Ramamurthy, and Greg Yeric. 2016. ASAP7: A 7-nm Predictive Process Design Kit. Online; https://asap.asu.edu/asap/. Predictive 7-nm academic Process Design Kit, version r1p7.

[5] Max Eland, Jeyan Thiyagalingam, Dinesh Pamunuwa, and Roshan Weerasekera. 2026. NL2GDS: LLM-aided Interface for Open Source Chip Design. *arXiv preprint arXiv:2603.05489* (2026). https://arxiv.org/abs/2603.05489

[6] Amur Ghose, Andrew B. Kahng, Sayak Kundu, and Zhiang Wang. 2025. ORFS-agent: Tool-Using Agents for Chip Design Optimization. In *Proceedings of the ACM/IEEE Workshop on Machine Learning for CAD (MLCAD)*. https://arxiv.org/abs/2506.08332

[7] Google and SkyWater Technology Foundry. 2020. SkyWater SKY130 Open Source PDK. https://github.com/google/skywater-pdk.

[8] Zhuolun He, Haoyuan Lu, Yucheng Chen, et al. 2024. ChatEDA: A Large Language Model Powered Autonomous Agent for EDA. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)* (2024). https://arxiv.org/abs/2308.10204

[9] Ravi Krishna, Suresh Krishna, and David Chin. 2026. Design Conductor: An Agent Autonomously Builds a 1.5 GHz Linux-Capable RISC-V CPU. *arXiv preprint arXiv:2603.08716* (2026). https://arxiv.org/abs/2603.08716

[10] Mingjie Liu, Teodor-Dumitru Ene, Robert Kirby, Chris Cheng, Nathaniel Pinckney, Rongjian Liang, Jonah Alben, Himyanshu Anand, Sanmitra Banerjee, et al. 2023. ChipNeMo: Domain-Adapted LLMs for Chip Design. *arXiv preprint arXiv:2311.00176* (2023). https://arxiv.org/abs/2311.00176

[11] Mingjie Liu, Nathaniel Pinckney, Brucek Khailany, and Haoxing Ren. 2023. VerilogEval: Evaluating Large Language Models for Verilog Code Generation. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. https://arxiv.org/abs/2309.07544

[12] Shang Liu, Wenji Fang, Yao Lu, et al. 2024. RTLCoder: Fully Open-Source and Efficient LLM-Assisted RTL Code Generation Technique. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)* (2024). https://arxiv.org/abs/2312.08617

[13] Yao Lu, Shang Liu, Qijun Zhang, and Zhiyao Xie. 2024. RTLLM: An Open-Source Benchmark for Design RTL Generation with Large Language Model. In *Proceedings of the 29th Asia and South Pacific Design Automation Conference (ASP-DAC)*. https://arxiv.org/abs/2308.05345

[14] Aditya Patra, Saroj Rout, and Arun Ravindran. 2024. AiEDA: Agentic AI Design Framework for Digital ASIC System Design. *arXiv preprint arXiv:2412.09745* (2024). https://arxiv.org/abs/2412.09745

[15] RISC-V Software Ecosystem. 2024. riscv-tests: Unit Tests for RISC-V Processors. https://github.com/riscv-software-src/riscv-tests.

[16] RISC-V Software Ecosystem. 2024. Spike RISC-V ISA Simulator. https://github.com/riscv-software-src/riscv-isa-sim.

[17] Wilson Snyder. 2024. Verilator: Open-source SystemVerilog simulator. Online; https://www.veripool.org/verilator/. Version 5.x.

[18] Shailja Thakur, Baleegh Ahmad, Benjamin Tan, Brendan Dolan-Gavitt, Ramesh Karri, Hammond Pearce, and Siddharth Garg. 2024. VeriGen: A Large Language Model for Verilog Code Generation. *ACM Transactions on Design Automation of Electronic Systems* (2024). https://arxiv.org/abs/2308.00708

[19] The OpenROAD Project. 2024. OpenROAD-flow-scripts: A Complete Open-Source RTL-to-GDS Flow. https://github.com/The-OpenROAD-Project/OpenROAD-flow-scripts.

[20] Stephen Williams. 2024. Icarus Verilog: Verilog compilation system. Online; https://github.com/steveicarus/iverilog. Version 12.0.

[21] Clifford Wolf. 2016. Yosys – A Free Verilog Synthesis Suite. In *Proceedings of the 21st Austrian Workshop on Microelectronics (Austrochip)*. https://yosyshq.net/yosys/