PPABench: Can Large Language Models Design Custom Silicon?

PPABench contributors*

July 22, 2026

Abstract

Large language models are increasingly used for individual steps of the silicon design pipeline. In order to guide the training of future frontier models, we must study how well agentic systems implement silicon on common chip industry metrics. We present **PPABench**, a benchmark that grades agentic chip design systems on power, performance, and area (PPA). PPABench consists of five real world ASIC tasks and a harness that grades the agentically generated silicon on power, performance, and area. PPABench evaluation works as follows: we provide the agent with a specification of an ASIC. The specification defines the chip I/O, technology node, and functional requirement. PPABench then runs the submitted agentic system and uses the open source toolchain and Skywater 130 PDK to implement the ASIC. In this paper, we report results for one agentic framework (Coresmith), reporting achieved power, performance, and area metrics.

1 Introduction

In the silicon industry, agents use large language models (LLMs) to design digital hardware: generating Verilog from natural-language specifications, debugging designs, and increasingly orchestrating the full architecture-to-GDS flow. As these systems improve, the field needs a way to benchmark the performance of autonomous chip design flows.

The problem is that the dominant evaluation target—RTL—is not a chip. A piece of Verilog can be syntactically valid, pass a unit simulation, and still be unsynthesizable, fail design-rule checks, miss timing, or consume an unacceptable area and power budget. The figures of merit that matter for hardware—power, performance, and area—do not exist until the design has been pushed through a physical-design (PnR) flow against a real process design kit (PDK). You cannot measure the area from source code without synthesis. You cannot know power without a routed netlist and a timing estimation. Existing chip-design benchmarks tend to score one axis—RTL functional correctness, or an RTL-based estimate, or a synthesis-tool result on a unit block—and therefore cannot answer the question a chip designer actually asks: can agents design silicon better than humans can?

A second problem is gameability. The moment a benchmark commits to a fixed expected output, an agent can hardcode it; the moment it trusts a value the design-under-test (DUT) reports about itself (a cycle counter, a "done" flag, an instruction count), the DUT can lie. An evaluation of automated designers must assume the designer will optimize for the metric, including by defeating the measurement.

*Correspondence: the PPABench project. This is a preprint describing the benchmark methodology and the first reference results, measured on designs produced by the Coresmith agentic framework.

PPABench addresses both problems with a single design principle: **grade real silicon, and measure it from the outside.** Concretely, our contributions are:

- A *deliverable contract* that turns a natural-language specification into a manufacturable artifact: the agent is handed a chip's I/O, technology node (sky130 HD), and functional requirement, and must deliver a core hardened into a fixed OpenFrame/chipignite chassis with a locked `user_project_wrapper` pinout (§2), so every submission is comparable and tape-out-shaped.
- A *two-half grading methodology* (§3) that runs a real RTL-to-GDS harden flow (synthesis → OpenROAD place-and-route → Magic DRC → netgen LVS) for PPA and signoff, *and* a *software-oracle* functional check that co-simulates the design's real RTL and compares its emitted output byte-for-byte against an independent software golden.
- A *structural anti-cheat model* (§4): functional correctness is graded *by result* against a golden recomputed for each run's randomized input, over K fresh trials, and nothing the design reports about itself is ever trusted.
- A suite of *five real-world ASIC tasks* (§5)—a GEMM accelerator, an AES-128 cipher, a streaming JPEG compressor, a 1024-point FFT, and a rasterization engine—and *measured reference results* (§7) for one agentic framework (Coresmith) on sky130, including two designs hardened all the way to a signed-off GDS layout.

Everything in PPABench is deterministic and toolchain-grounded: the same submission, graded twice, produces the same scorecard, and every number on that scorecard is traceable to a tool output (a routed `.def`/`.gds`, an STA report, a DRC/LVS log, a byte-exact co-simulation) rather than to a model's self-report.

2 The Chassis and the Contract

PPABench fixes a single *chassis*: the manufacturable harness into which every submitted core is dropped. The chassis is the OpenFrame/chipignite `user_project_wrapper` on the SkyWater **sky130 HD** standard-cell library. Fixing the chassis serves two goals at once: *comparability*—every design is hardened with the same library, the same pinout, and the same top-level contract, so PPA numbers are measured on an equal footing—and *manufacturability*—the wrapper is the actual tape-out interface of a shuttle program, so a passing design is shaped like something that could be fabricated, not an abstract block floating in a vacuum. Figure 1 shows the resulting split: the agent designs the inner core, while the grader owns the chassis and hidden harness and observes the design only across the locked pins.

The locked pinout. The contract pins the user-project interface so that the harness can drive and observe any submission without knowing its internals:

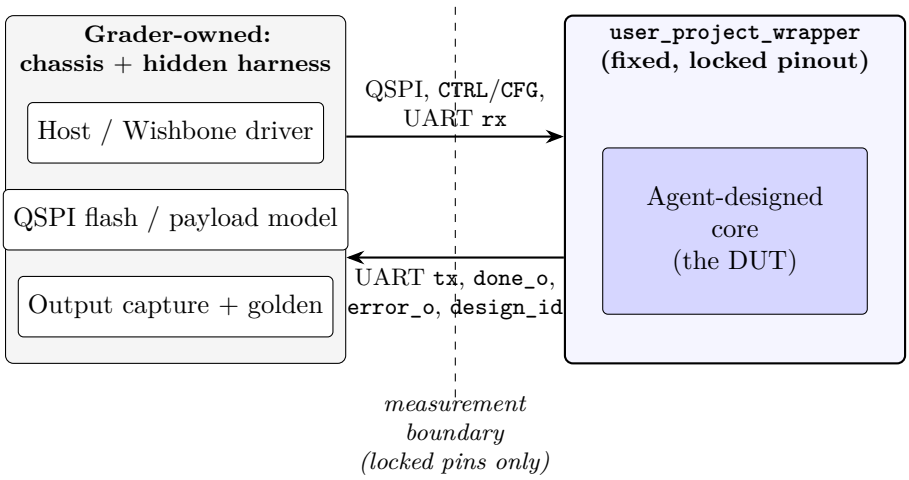

Figure 1: The DUT boundary. The agent designs only the inner core (the DUT), which is dropped into the fixed `user_project_wrapper`. Everything outside the wrapper—the host, the payload model, and the output capture / software-oracle comparison—is grader-owned and never seen by the submitter. The grader interacts with the design exclusively across the locked pins: it drives control and randomized payload in, and observes the emitted output and status out. Nothing internal to the DUT is trusted or measured directly.

Pins	Function	Direction
`io[24]`	QSPI `sck`	out
`io[25]`	QSPI `csb`	out
`io[26:29]`	QSPI `io[3:0]`	bidir
`io[30]`	UART `rx`	in
`io[31]`	UART `tx`	out
`io[32:34]`	`design_id[2:0]`	out
`io[35]`	`done_o`	out
`io[36]`	`error_o`	out
Wishbone host control @ `0x30000000`: `CTRL/STATUS`, `CFG0/CFG1`		

The host controls the core over the Wishbone register block: it releases the core from reset (`CTRL.start`), reads status (`STATUS.busy/done/error`), and passes scalar arguments (`CFG0/CFG1`) that are mirrored to design-visible MMIO. Payloads and operands too large for a scalar (matrices, keys, test images) are delivered through the QSPI-attached memory model; results are emitted on the design's output stream (over UART or the streaming interface) and captured at the wrapper boundary. A design signals completion on `done_o` and faults on `error_o`. The captured output stream—not any status the design reports about itself—is what grading consumes (see §4).

The reference harness. The chassis is validated by a reference harness that exercises the wrapper machinery end to end—host control, payload delivery, output capture, and done/error detection—so that each of the five task cores drops into the *same* wrapper and is graded by the *same* flow, without bespoke per-task infrastructure. The wrapper contract and the co-simulation harness are held constant across tasks; only the inner core changes.

The START/DONE protocol. After `wb_rst_i` deasserts, the inner core is held in local reset until the host writes `CTRL.start`; this gates the core's `resetn` so that execution begins

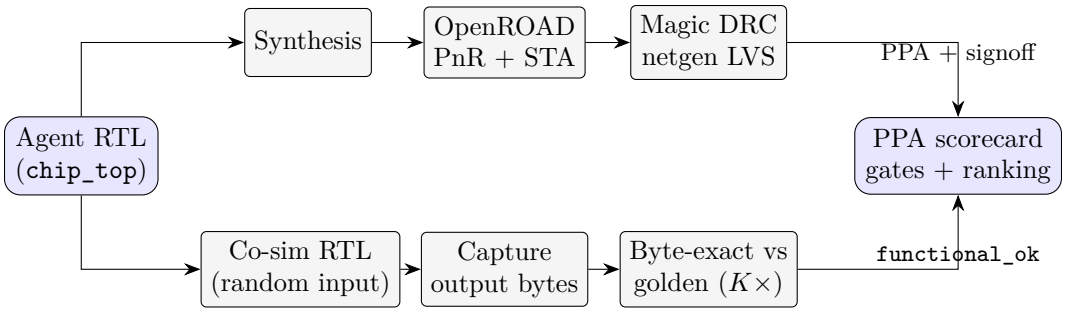

Figure 2: The two-half grading flow. The physical half (top) hardens the RTL through the open-source RTL-to-GDS flow, yielding area, power, $f_{\max}$, and the DRC/LVS signoff booleans. The functional half (bottom) co-simulates the same RTL on randomized input, captures the emitted output bytes, and requires them to be byte-identical to an independent software golden over K trials. The two independent halves fuse into one PPA scorecard: hard gates decide pass/fail, and the passers are ranked by efficiency.

deterministically at the start write. The core runs its workload and signals completion by latching `STATUS.done` and raising `io[35]`; any fault latches a sticky `STATUS.error` on `io[36]`. Grading captures the output stream emitted between the start write and `done` and compares it byte-for-byte against the software golden for this run's input.

3 Grading Methodology

Every submission is graded on *both halves of being a real chip*. The two halves are produced by independent flows and then fused into one scorecard (Figure 2).

3.1 Harden: the physical half

The harden pass runs the open-source RTL-to-GDS flow against sky130 HD:

1. **Synthesis** maps the user project to a sky130 gate-level netlist.
2. **OpenROAD** performs floorplanning, placement, clock-tree synthesis, routing, and static timing analysis (STA), producing area, utilization, gate count, total/dynamic/leakage power, worst negative slack (WNS), and $f_{\max}$.
3. **Magic** runs design-rule checking (DRC) and extracts the GDS layout.
4. **netgen** runs layout-versus-schematic (LVS) between the extracted layout and the synthesized netlist.

The harden half yields the physical figures of merit—`area_um2`, `util_pct`, `gate_count`, `total_power_mw`, `dynamic_power_mw`, `leakage_power_mw`, `fmax_mhz`, `wns_ns`—and the signoff booleans DRC-clean and LVS-match. These are real tool outputs on a routed design, not estimates.

3.2 Functional verification: the software oracle

The functional half is graded *by result*. The design's real RTL is co-simulated at the wrapper boundary exactly as a bring-up engineer would drive a packaged part:

1. The harness loads this run's input—randomized per run—into the chassis (a host scalar via `CFG0`/`CFG1`, or a payload baked into the QSPI memory model), then releases the core with `CTRL.start`.
2. It captures the byte stream the design emits at the wrapper boundary.
3. An independent *software golden*, implemented from the task specification, computes the correct output for the *same* input.
4. The two byte streams must be **byte-identical**. Nothing the design reports about its own state is trusted; only the externally observed output is scored.

The functional half yields `functional_ok` (does the emitted output match the software golden for this run's input?), the observed run length in clock cycles, and the workload size (elements, bytes, or pixels processed). We additionally report functional *coverage* on the RTL simulation, both aggregated across the design and at the weakest individual block.

3.3 Fusion: the PPA scorecard

Energy is computed at the *operating clock*—the task's target clock (50 MHz for the memory-bound designs), the rate at which cycles actually elapse and power is actually drawn—*not* at $f_{\max}$; $f_{\max}$is reported separately as timing headroom. Combining the routed power P (mW) and frequency f (MHz) from the harden half with the observed run length from the functional half gives the energy and efficiency metrics of §6. Because the two halves come from independent tools and the functional check is byte-exact against an external golden, neither half can be gamed by manipulating the other.

3.4 Pass/fail versus ranking

PPABench separates *gating* from *ranking*:

- **Pass/fail (hard gates):** PPA caps (area/power/gate/macro limits from the task constraints), `functional_ok` (byte-exact over K trials), `timing_met` (WNS ≥ 0 at the target clock), DRC-clean, LVS-match, and pinout/contract compliance. A submission that fails any gate is not a chip.
- **Ranking (reported, not gated):** the power/area/performance KPIs. Everyone who passes is a real, manufacturable chip; the leaderboard then sorts the passers by efficiency (e.g. lowest energy per unit of work or lowest EDAP).

This split is deliberate. It rewards designs that close the loop to real silicon before it rewards clever efficiency, and it makes "passing" mean something concrete (a routable, DRC-clean, LVS-matched, timing-met, functionally-correct design) rather than a soft score.

4 Anti-Cheat and Verifiability

Because PPABench grades *automated* designers, we assume the designer will optimize for the metric—including by attacking the measurement. The defenses are structural, not heuristic.

Threat 1: hardcoding the answer. An agent could embed the expected output in the design and skip the computation. *Defense:* the correct output is not fixed. Every task perturbs its input at runtime—a host scalar via `CFG0`, a random key/plaintext, a fresh random test image or matrix delivered per run—and functional correctness is verified *by result* against a software golden computed for *this run's* input. A hardcoded constant fails on the next seed.

Threat 2: trusting a self-reported status. A design could assert `done` early or report a flattering status register. *Defense:* grading reads only the *emitted output stream* captured externally at the wrapper boundary, compared byte-for-byte with the software golden. Asserting `done` without the correct output stream is a functional failure; a status register the design writes about itself is never scored.

Threat 3: precomputation or illegal parallelization. A design could precompute results offline, or exploit parallelism the task is not supposed to allow. *Defense:* tasks are chosen so that the work cannot be precomputed or trivially parallelized. The AES cipher is run with sequential (CBC-style) chaining—block i depends on the ciphertext of block $i-1$—so the ciphertext for a random plaintext cannot be produced ahead of time or computed out of order, and the random key makes any precomputed table useless.

Threat 4: overfitting to the public tasks. An agent could be tuned specifically to the released tasks. *Defense:* **held-out challenge variants**. A held-out keyed challenge—kept private—runs the *same* chassis, harden flow, and software-oracle machinery as the public tasks, providing a contamination/overfitting check on a function the submitter has never seen.

Verify-by-result over K trials. Functional correctness is not a single check. The harness runs K fresh random trials and requires *all* of them to match the golden byte-for-byte. A design that happens to be correct on one seed but not others does not pass. This converts "looks correct once" into "is correct as a function," which is the property a chip must have.

Together these make the scorecard *verifiable*: every gate and every ranked KPI is reproducible from tool outputs and external measurements, and the result of grading the same design twice is the same scorecard.

5 The Five ASIC Tasks

The first suite contains five real-world ASIC tasks. Each hands the agent a specification—the chip's I/O, the technology node (sky130 HD), and a functional requirement—and the deliverable is a core in the locked wrapper. All five share the chassis, the harden flow, and the software-oracle machinery; they differ in the datapath they demand and the unit of work they process.

5.1 GEMM — integer matrix-multiply accelerator

An integer general matrix-multiply (GEMM) accelerator: operand and result SRAMs fed by a multiply-accumulate datapath, host-controlled through the register interface. *Oracle:* the emitted result is byte-exact against a Python GEMM golden, including a $32 \times 32 \times 32$ case.

Table 1: PPABench figures of merit. P = routed total power (mW); f = operating (target) clock (MHz); `cyc` = observed run length in clock cycles; `work` = elements/bytes/pixels processed. Energy is evaluated at f, not $f_{\max}$.

KPI	Formula	Notes
`area_um2` / `area_mm2`	(routed)	silicon area
`total_power_mw`	(routed)	dynamic + leakage
$f_{\max}$ / `wns_ns`	(STA)	timing / headroom
`energy_per_cycle_pj`	$P/f \times 1000$	pJ per cycle
`total_energy_nj`	$P \times \text{cyc}/f$	nJ for the run
`wall_time_us`	cyc/f	µs for the run
`energy_per_byte_pj`	total_energy/bytes	workload-normalized
`energy_per_pixel_pj`	total_energy/pixels	raster / JPEG
`throughput`	work $\times f/\text{cyc}$	work per second
`leakage_fraction`	P_{leak}/P	activity-independent
`area_per_kgate_um2`	area/(gates/1000)	density
`edp_nj_us`	energy × wall_time	energy-delay product
`edap_nj_us_um2`	edp × area	energy-delay-area product

5.2 AES — AES-128 block cipher

An AES-128 block-cipher datapath. *Oracle:* byte-exact against a FIPS-197 software golden over multiple random key/plaintext seeds. Sequential CBC-style chaining makes results non-precomputable and random keys defeat hardcoding.

5.3 JPEG — streaming baseline compressor

A streaming baseline JPEG / BT.656 compressor pipeline: 8×8 block → 2-D DCT → quantize → zig-zag → run-length + Huffman → JFIF byte stream. *Oracle:* the emitted bytes are byte-exact against a software encoder golden over multiple frames.

5.4 FFT — 1024-point fixed-point transform

A 1024-point fixed-point FFT, including multi-cycle butterfly datapaths over SRAM-backed sample banks. *Oracle:* byte-exact transform output against a Python FFT golden.

5.5 Raster — rasterization engine

A raster-graphics / rasterization engine. *Oracle:* byte-exact framebuffer output against a software rasterizer golden, pinned to a reference hash.

6 The KPI Set / Figures of Merit

The scorecard fuses the harden half and the functional half into the figure-of-merit set in Table 1. All energy is computed at the operating clock f (the target clock), where P is the routed total power, `cyc` is the observed run length in clock cycles, and `work` (elements, bytes, or pixels) is the workload normalizer.

Table 2: Functional verification and coverage closure (Coresmith, sky130 HD). Byte-exact = the design's emitted output matched an independent software golden. Coverage = functional coverage on the RTL simulation. Synth $f_{\max}$ = post-synthesis achieved frequency from STA on the synthesized netlist.

Task	Blocks	Byte-exact oracle	Cov. (agg)	Cov. (min block)	Synth $f_{\max}$
GEMM	9/9	✓ (incl. $32 \times 32 \times 32$)	97.95%	96.23%	50.0 MHz
AES	6/6	✓ (FIPS-197, multi-seed)	94.98%	80.50%	202.0 MHz
JPEG	11/11	✓ (multi-frame, 0 mismatch)	89.10%	76.54%	50.0 MHz
FFT	7/7	✓	89.03%	79.18%	52.83 MHz
Raster	8/8	✓ (pinned SHA)	90.60%	83.28%	50.0 MHz

EDP and EDAP are the headline efficiency metrics: EDP rewards designs that are both fast and low-energy (penalizing one bought at the cost of the other), and EDAP further folds in silicon area, so that a design cannot win EDP by spending unlimited area. Leakage power is reported as a fraction because it is activity-independent and therefore exact even under default switching activity (see §9).

7 Reference Results

We report reference results for one agentic framework, Coresmith, which drives the flow from architecture through per-block RTL and design verification, integration, and validation, and then through a physical-design backend to GDS. On the full-stack Amaranth-arm sweep (§7.2), four of the five designs (GEMM, AES, Raster, and FFT) were pushed all the way to a signed-off, manufacturable layout; the remaining one (JPEG) is functionally verified and synthesized, with its physical-design backend still blocked.

7.1 Functional verification and coverage closure

Every design's `chip_top` RTL passed integration and validation, byte-exact against its software golden oracle, over the software-oracle checks of §3. Table 2 reports the per-task block count, the byte-exact result, the functional coverage (aggregate and weakest block), and the post-synthesis achieved frequency reported by STA on the synthesized netlist (the per-block target was met; the AES datapath closes far higher).

All five designs are functionally correct by result: byte-exact against the software golden across the graded trials, with no mismatches, and with functional coverage closing above the floor for every block. A later sweep on Coresmith's *Amaranth* RTL-generation arm (a second, independently-implemented code-generation path inside the same framework, as opposed to the MyHDL arm used for Table 2) re-confirms the same pattern architecture-to-integration: all five designs report `pipeline_done=true` with a byte-exact software-oracle result before backend hardening begins. §7.2 reports that sweep's backend outcome.

7.2 Backend PPA: routed sky130 signoff (GEMM, AES, Raster, FFT)

A full-stack sweep on the Amaranth arm pushed all five designs from architecture through the backend. Four—GEMM, AES, Raster, and FFT—closed to a signed-off, manufacturable layout in the fixed `user_project_wrapper`: DRC-clean, LVS-matched, and timing-met. JPEG synthesized

Table 3: Backend PPA on routed sky130 HD, hardened in the `user_project_wrapper`, Amaranth model arm. Area, power, timing, DRC, and LVS are tool outputs on the routed design. Timing is reported at the 50 MHz target clock. Area is the placed design area (die, macros, fill, and routing overhead beyond the mapped standard-cell area); †DRC is 0 only after an audit classifies all raw flagged geometry as lying inside a signed-off cell/macro interior (see text); ‡LVS device/net counts match after excluding documented signal-less fill/tap/decap classes and openframe GPIO top-pin constant-tie/replication structure (see text); Raster's power is invalid (see text); FFT's power is a valid PnR estimate (see text).

Metric	**GEMM**	**AES**	**Raster**	**FFT**
Area (mm^2), util.	1.059, 41%	0.306, 26%	2.315, 39%	1.374, 30%
Std. cells / macros	5,886 / 4 SRAM	21,119 / 0	4,722 / 14 SRAM	23,446 / 5 SRAM
Total power (mW)	12.9	16.6	invalid	31.4
Setup slack (ns)	+5.52	+11.32	+6.42	+7.23
Hold slack (ns)	+0.38	+0.40	+0.41	+0.46
DRC violations	0	0†	0†	0
LVS	match	match‡	match‡	match‡
GDS layout (MB)	24.35	43.80	44.72	75.18

and routed but did not close either DRC or LVS (§7.4). Table 3 gives the measured routed PPA results for the four that signed off, captured directly from each run's DRC/LVS/timing reports and JSON signoff artifacts (`backend_signoff_result.json`, `drc_result.json`, `lvs_result.json`; run dirs `exp-{gemm,aes,raster,fft}-sweep-20260722`). FFT closed on a second backend pass after re-routing (§7.2 *FFT* paragraph below); its evidence lives under `pnr/reroute_iter8/signoff/` in that run directory rather than the top-level `pnr/` used by the other three.

On the LVS-match criterion: no openframe-wrapped design in this sweep produces a raw, unique netgen match. Every one carries some combination of physical-only fill/tap/decap cells — present in the placement-derived reference netlist, legitimately dropped by Magic's electrical extraction — and openframe GPIO constant-tie/replication structure that raw netgen reports as top-pin or device-count mismatches. "Match" in Table 3 means these documented, signal-less classes account for the full residual and no real net-fragment (signal) mismatch remains, not that netgen's first pass reported unique equivalence; we disclose the criterion once here rather than re-deriving it per design.

GEMM. The GEMM accelerator's Amaranth-arm sweep hardens to a **1,059,057 µm²** (1.059 mm²) placed design at 41% utilization on a 2.789 mm² die (5,886 mapped standard cells, 46,485.8336 µm² mapped cell area, four placed SRAM macros), drawing **12.9 mW** (PnR estimate; 12.5 mW from OpenRCX-extracted parasitics). It is DRC-clean (Magic reports "DRC count: 0") and LVS-matches uniquely (Δ_{dev}=0, Δ_{net}=0; netgen: "Circuits match uniquely," 6,879 = 6,879 devices, 7,276 = 7,276 nets). Timing *meets* the 50 MHz target with OpenRCX-extracted setup slack +5.52 ns and hold slack +0.38 ns—an improvement over an earlier baseline run of the same design that missed timing by −0.20 ns (§7.3). GDS: 24,352,164 bytes, SHA-256 `cfbbaf6a...`.

AES. The AES cipher's Amaranth-arm sweep hardens to a **306,106 µm²** (0.306 mm²) placed design at 26% utilization on a 1.192 mm² die (21,119 mapped standard cells, 236,585.6544 µm² mapped cell area, no hard macros—the design's buffers are register-tier), drawing **16.6 mW** total from OpenRCX-extracted parasitics (10.5 mW sequential, 0.155 mW combinational, 5.89 mW clock). DRC is classified clean after a coordinate audit (see Table 3 footnote). LVS uses hierarchical

extraction with signed-off standard cells kept as LEF black boxes: device and net counts match exactly (23,383 = 23,383 devices, 23,349 = 23,349 nets), and the raw top-pin mismatch is fully accounted for by 161 constant-tied/replicated output records, which the run's deterministic classifier accepts as benign. Timing meets the 50 MHz target with worst setup slack +11.32 ns and worst hold slack +0.40 ns. GDS: 43,802,406 bytes, SHA-256 `822cecba...`. *Caveat:* this run's `backend_results.json` orchestration record was not regenerated after the final signoff (a disclosed orchestration-state gap, not an open silicon gate); the numbers above are taken directly from the run's DRC/LVS/timing/power report files, not from that missing summary file.

Raster. The rasterization engine's Amaranth-arm sweep hardens to a **2,315,166 µm²** (2.315 mm²) placed design at 39% utilization, with 14 placed SRAM macros and 4,722 final routed cells. DRC is classified clean after a coordinate audit (Table 3 footnote). LVS matches after excluding two signal-less physical decap/filler classes: 5,468 = 5,468 devices, 5,799 = 5,799 nets, Δ_{dev}=0, Δ_{net}=0, with all 14 SRAM macros and both tie-high cells verified equivalent. Timing meets the 50 MHz target with setup slack +6.42 ns and hold slack +0.41 ns. GDS: 44,720,436 bytes, SHA-256 `f9a8b119...`. *Power is not reported*: the OpenRAM macro internal-power characterization returned a nonphysical value (5.09×10^7 W); the run's own harness flags `power_valid=false`, and we do not substitute the finite non-macro-plus-leakage subtotal (≈2.264 mW) as a full-chip figure, since macro dynamic power is missing from it.

FFT. The FFT's Amaranth-arm sweep hardens to a **1,374,175 µm²** (1.374 mm²) placed design at 30% utilization (23,446 placed standard-cell instances, 5 placed SRAM macros), drawing **31.4 mW** total from the routed PnR power report (26.8 mW internal, 4.48 mW switching, 0.095 mW leakage; a valid figure, not the nonphysical OpenRAM value seen on Raster). It is DRC-clean (hierarchical Magic DRC report is empty; the signoff log records "No errors found. Total DRC errors found: 0"). Timing meets the 50 MHz target with setup slack +7.23 ns and hold slack +0.46 ns. GDS (macro-referenced): 75,184,094 bytes, SHA-256 `9cbd48e5...`. LVS-matches under the same benign classification as GEMM/AES/Raster: at the top-pin level, the only unmatched pins are 88 openframe `io_out`/`io_oeb` GPIO constant-tie/replication entries plus the SRAM `VPWR`/`VGND` black-box annotations (raw verdict line "Top level cell failed pin matching," reconciled benign, identical in kind to Raster and GEMM). The underlying top-level device/net *counts* initially read as mismatched too — netgen's grand total shows 26,156 devices / 26,565 nets (layout, flagged `Mismatch`) against 26,159 devices / 26,487 nets (reference, also flagged `Mismatch`), Δ_{dev}=−3, Δ_{net}=+78 — but this traces to a fully identified, benign cause: the only 3 `(no matching element)` class entries in the entire 6,288-line LVS report are `sky130_fd_sc_hd__tapvpwrvgnd_1` (1 tap cell) and `sky130_fd_sc_hd__fill_1`/ `fill_2` (2 fill-cell classes) — zero-transistor, signal-less physical cells that the reference `_pwr.v` netlist enumerates but that Magic's `ext2spice` legitimately drops from electrical extraction; every other device class matches exactly, and the net delta traces to the same dropped cells fragmenting `VPWR`/`VGND`/write-mask net classes, with no signal shorts in the net-fragment listing. This is the same signal-less-physical-filler exclusion already applied to Raster's LVS (§7.2 *Raster* paragraph above). *How it closed*: FFT initially blocked on DRC with thousands of route-level markers (§7.4 reported the earlier state); the cause was a hand-added `-droute_end_iter 2` cap in the run directory's detailed-routing script, which truncated OpenROAD's router before convergence. Re-running detailed route uncapped (`-droute_end_iter 16`), on the identical die, utilization, floorplan, and macro placement, converged route-DRC markers 13,857 → 4,621 → 3,634 → 179 → 12 → 7 → **0** over six router optimization iterations, after which Magic DRC signoff came back clean. The original blocker was a run-directory routing-iteration cap, not a congestion or

engine limit (§7.4).

7.3 Prior DRC-clean baseline runs

Independent of the Amaranth-arm sweep above, earlier hardening runs of the same three designs also reached clean or near-clean physical signoff, corroborating that these designs route on sky130 HD. An earlier full MyHDL-arm run of GEMM (`exp-gemm-full4-20260721`) closed *full* signoff: 0 DRC violations under hierarchical Magic DRC, LVS match (Δ_{dev}=0, Δ_{net}=0; 5,041 = 5,041 devices, 5,147 = 5,147 nets, with 233 pin-table entries classified as benign tied/replicated aliases), setup slack +0.25 ns met (tight but positive) and hold slack +0.42 ns met, at 14.2 mW total power, with a 25,166,526-byte routed GDS. Two earlier baseline runs of GEMM and AES on a prior milestone (`exp-gemm-m13-20260718`, `exp-aes-m13-20260718`) reached LVS match under the same benign-tie classification but did *not* close full signoff: `exp-gemm-m13-20260718` meets DRC/LVS but genuinely misses timing at 50 MHz (setup WNS −0.20 ns, TNS −0.27 ns); `exp-aes-m13-20260718` meets timing (WNS/TNS 0.00 ns) but its LVS genuinely fails to match (device delta 3, net delta 77,662). A third baseline run (`exp-raster-m13-20260718`) reached LVS match with the same benign-tie exception pattern later seen in the Raster sweep, but its own persisted DRC report recorded 72 unclassified `met4.4a` violations at the time it was captured, despite that run's own orchestration summary recording zero; we do not count it as independent DRC corroboration for Raster and rely on the classified-clean result in §7.2 instead. We report all of this plainly, including the partial and contradictory results, rather than selecting only the clean runs.

7.4 Backend closure (FFT), and one that did not close (JPEG)

Both designs that were backend-blocked in an earlier pass of this sweep were revisited. One (FFT) closed fully: DRC, LVS, and timing all now meet the same signed-off bar as GEMM/AES/Raster, and it is reported in Table 3 and §7.2. The other (JPEG) did not close, and a second attempt at closing it produced a report that was itself wrong on both DRC and LVS; we describe both below rather than reporting either as generic "in progress."

FFT (`exp-fft-sweep-20260722`) originally blocked on **1,985** hierarchical Magic DRC violations dominated by metal spacing, with OpenROAD's detailed router left holding 3,634 unresolved markers after three exhausted attempts. The root cause was a hand-added `-droute_end_iter 2` cap in the run directory's detailed-routing script, truncating the router well before convergence, not datapath congestion or a routing-capacity limit: re-running detailed route uncapped on the identical die/utilization/floorplan converged to 0 route-DRC markers and 0 Magic DRC violations (§7.2 *FFT* paragraph). Timing and GDS generation followed cleanly. LVS initially looked open too — the same re-verification pass that confirmed DRC surfaced a netgen-flagged top-level device/net-*count* mismatch (Δ_{dev}=−3, Δ_{net}=+78) — but tracing it against the full class list resolved it completely: the only 3 `(no matching element)` entries in the entire report are 1 tap cell and 2 fill-cell classes, zero-transistor physical-only cells the reference netlist emits but Magic's extraction correctly omits, with the net delta attributable to the same dropped cells and no signal shorts (§7.2 *FFT* paragraph). We flag the DRC fix as a lesson for the benchmark's own harness as much as for the agent under test — a blocked run should be checked for run-directory tool-configuration artifacts before being counted as a genuine engine or design limitation — and we flag the LVS trace as a reminder that a raw netgen "Mismatch" tag is a prompt to investigate the specific class list, not a verdict in itself.

JPEG (`exp-jpeg-sweep-20260722`) originally blocked on a 3-rectangle `li.3` Magic DRC result plus an LVS net-count mismatch (`Circuit 1 contains 22993 nets, Circuit 2 contains`

`29156 nets.  *** MISMATCH ***`, devices matching exactly at 29,071 = 29,071). A second, engine-standard backend pass over the same routed placement was later reported upstream as closed under the same benign-tie criterion used for GEMM/AES/Raster/FFT. On re-verification for this paper, that report did not hold up on either axis, and unlike FFT's residual, neither traces to a benign physical-cell-accounting explanation. LVS: netgen's grand total reads 29,076 devices / 22,690 nets (layout) against 29,079 devices / 29,156 nets (reference) — the same $\Delta_{\text{dev}}=-3$ magnitude seen on FFT, but a $\Delta_{\text{net}}\approx+6{,}466$ net delta roughly 80× larger and, on inspection, architectural rather than extraction noise: JPEG's DCT row/column caches are implemented as flop arrays rather than SRAM macros, and the resulting read-mux symmetry defeats a unique LVS match. No evidence of a real electrical short, but not a clean certificate either. DRC: the automated signoff step's own console summary read `Total DRC errors found:  0`, but the same tool invocation also wrote a 785,085-byte DRC report enumerating thousands of real, coordinate-tagged `li.1/li.3/li.c2` local-interconnect violations across the die (roughly 26,695 constituent rectangles, 4,166 by Magic's own error count) — the "0" was an artifact of the harness's stdout parser treating an empty Tcl list result as a zero count, not a genuine clean layout (a fix is in progress). We confirmed the real count independently by re-running Magic from scratch against the same routed DEF with the same non-flattened DRC methodology used for FFT's (independently verified) clean result, which returned **4,166** violations. JPEG is therefore still reported **blocked** on both DRC and LVS: a routed, 119,067,964-byte flattened candidate GDS exists (plus a 48,257,516-byte macro-referenced compact GDS generated from the same routed DEF for archive consistency with the other four designs), but neither is a signoff deliverable. True closure needs an li-spacing repair plus re-architecting the DCT caches as SRAM macros — out of scope for this reference run. This episode is reported for the same reason §7.3 reports the raster-m13 DRC/orchestration-summary contradiction: an automated "clean" verdict is not, by itself, evidence of a clean layout, and this benchmark's standard is to check and disclose rather than to accept a green summary field at face value.

8 Related Work

PPABench sits at the intersection of agentic software-engineering benchmarks, systems performance benchmarks, and ML-for-EDA datasets. *SWE-bench* [1] established the template of grading autonomous agents by whether their patches pass held-out tests on real repositories—the spirit of our verify-by-result gating, transplanted from software to silicon. *MLPerf* [2] showed the value of a tightly-specified, externally-measured systems benchmark with rigorous run rules; PPABench's insistence on byte-exact, externally-verified functional checks and a fixed chassis is in the same tradition. On the hardware side, RTL-generation benchmarks such as *RTLLM* [3] and *VerilogEval* [4] score LLM-produced Verilog for functional correctness in simulation, and domain-adapted models such as *ChipNeMo* [5] target EDA tasks—but these grade RTL, not hardened silicon. ML-for-EDA datasets and flows such as *CircuitNet* [6] and the *OpenROAD*/ORFS flow [7] provide the physical-design substrate (and, in our case, the actual harden tools) but are infrastructure rather than agent benchmarks. PPABench's contribution is to combine these: grade an *agent*'s deliverable on *real PPA from a real harden flow* plus *functional correctness from a byte-exact software oracle*, over a fixed manufacturable chassis.

9 Limitations and Future Work

Dynamic power uses default switching activity. The routed power we report is estimated from default switching activity rather than from the actual simulation. Leakage and all physical/timing metrics (area, $f_{\max}$, WNS, DRC, LVS) are exact today; dynamic power is approximate. Feeding the co-simulation activity trace into power analysis for activity-accurate dynamic power is a planned refinement. We report leakage as a fraction precisely because it is activity-independent and therefore already exact.

Four of five backends are closed; one is not yet closed. All five designs are functionally verified byte-exact and synthesized. On the Amaranth model arm's full-stack sweep (§7.2), GEMM, AES, Raster, and FFT route to full signoff (DRC-clean, LVS-matched, timing-met at 50 MHz). JPEG synthesizes and routes with positive timing slack but does not close DRC *or* LVS, and we report the specific blocking defects (§7.4) rather than a generic "in progress." A second attempt at closing JPEG's backend was reported upstream as successful; on independent re-verification for this paper that report did not hold up on either axis, and we disclose the mechanism (an automated DRC-count parser misreading an empty tool-list result as a zero violation count, plus an architectural LVS cause — flop-array caches where an SRAM macro would give a clean match) rather than the closure it initially appeared to be. That same re-verification pass initially surfaced an apparent LVS device/net-count mismatch on FFT as well; tracing it against the full netgen class list resolved it to 3 fully identified signal-less physical cells (1 tap, 2 fill) that the reference netlist counts but Magic's extraction correctly drops, so FFT is reported as fully signed off rather than at a separate tier. Raster's power figure is invalid due to a nonphysical value from the OpenRAM macro-power characterization and is reported as unavailable rather than substituted or rounded. We additionally disclose, rather than omit, that some earlier baseline hardening runs of these same designs produced partial or internally-contradictory DRC/LVS/timing results (§7.3).

Scope. The first suite is five tasks on one PDK (sky130 HD). Broadening to more tasks, more PDKs, and more design styles (DSP, networking, larger accelerators) is future work, as is a fully cloud-hosted held-out evaluation so that holdout challenges can be graded without exposing them. None of these change the core methodology—real harden plus byte-exact, anti-gameable functional measurement over a fixed chassis.

10 Conclusion

Agentic chip design needs a benchmark that grades chips, not source code. PPABench does this by fixing a manufacturable chassis, hardening every submission through a real RTL-to-GDS flow for true PPA and signoff, and verifying function through a software oracle that captures the design's emitted output and requires it to be byte-identical to an independent golden over K randomized trials. Across five real-world ASIC tasks—GEMM, AES, JPEG, FFT, and rasterization—the Coresmith framework produced designs that are all functionally correct by result, with functional coverage closing on every block. Four of the five were hardened to a signed-off, manufacturable GDS on the Amaranth model arm: a GEMM accelerator at 1.059 mm^2 and 12.9 mW, an AES cipher at 0.306 mm^2 and 16.6 mW, a rasterization engine at 2.315 mm^2 (power not reported, invalidated by a nonphysical OpenRAM macro-power characterization), and an FFT at 1.374 mm^2 and 31.4 mW (closed on a second backend pass once a run-directory routing-iteration cap was removed, with an initially-flagged

LVS device/net-count mismatch traced to 3 signal-less physical cells and resolved benign)—all four DRC-clean, LVS-matched, and timing-met at 50 MHz. The remaining one, JPEG, is functionally verified, synthesized, and routed with positive timing slack, but is blocked on both DRC and LVS (a much larger, architecturally-rooted LVS net-count mismatch than FFT's transient one, and a DRC result that a second closure attempt mis-reported as clean due to an automated-parser bug we identify and describe) rather than reported as generically "in progress." Earlier baseline hardening runs of GEMM, AES, and Raster are reported alongside these, including the runs whose own DRC/LVS/timing evidence was partial or self-contradictory, because the benchmark's standard is disclosure, not curation. These are reference results, not ceilings: the benchmark's purpose is to map out, honestly and from the outside, how well autonomous systems can turn a specification into real silicon.

References

[1] C. E. Jimenez, J. Yang, A. Wettig, S. Yao, K. Pei, O. Press, and K. Narasimhan. SWE-bench: Can Language Models Resolve Real-World GitHub Issues? *ICLR*, 2024.

[2] P. Mattson et al. MLPerf Training Benchmark. *Proc. Machine Learning and Systems (MLSys)*, 2020.

[3] Y. Lu, S. Liu, Q. Zhang, and Z. Xie. RTLLM: An Open-Source Benchmark for Design RTL Generation with Large Language Models. *Asia and South Pacific Design Automation Conf. (ASP-DAC)*, 2024.

[4] M. Liu, N. Pinckney, B. Khailany, and H. Ren. VerilogEval: Evaluating Large Language Models for Verilog Code Generation. *IEEE/ACM Int. Conf. on Computer-Aided Design (ICCAD)*, 2023.

[5] M. Liu et al. ChipNeMo: Domain-Adapted LLMs for Chip Design. *arXiv:2311.00176*, 2023.

[6] Z. Chai et al. CircuitNet: An Open-Source Dataset for Machine Learning in VLSI CAD Applications. *IEEE Trans. on Computer-Aided Design (TCAD)*, 2023.

[7] T. Ajayi et al. OpenROAD: Toward a Self-Driving, Open-Source Digital Layout Implementation Tool Chain. *Government Microcircuit Applications and Critical Technology Conf. (GOMACTech)*, 2019.