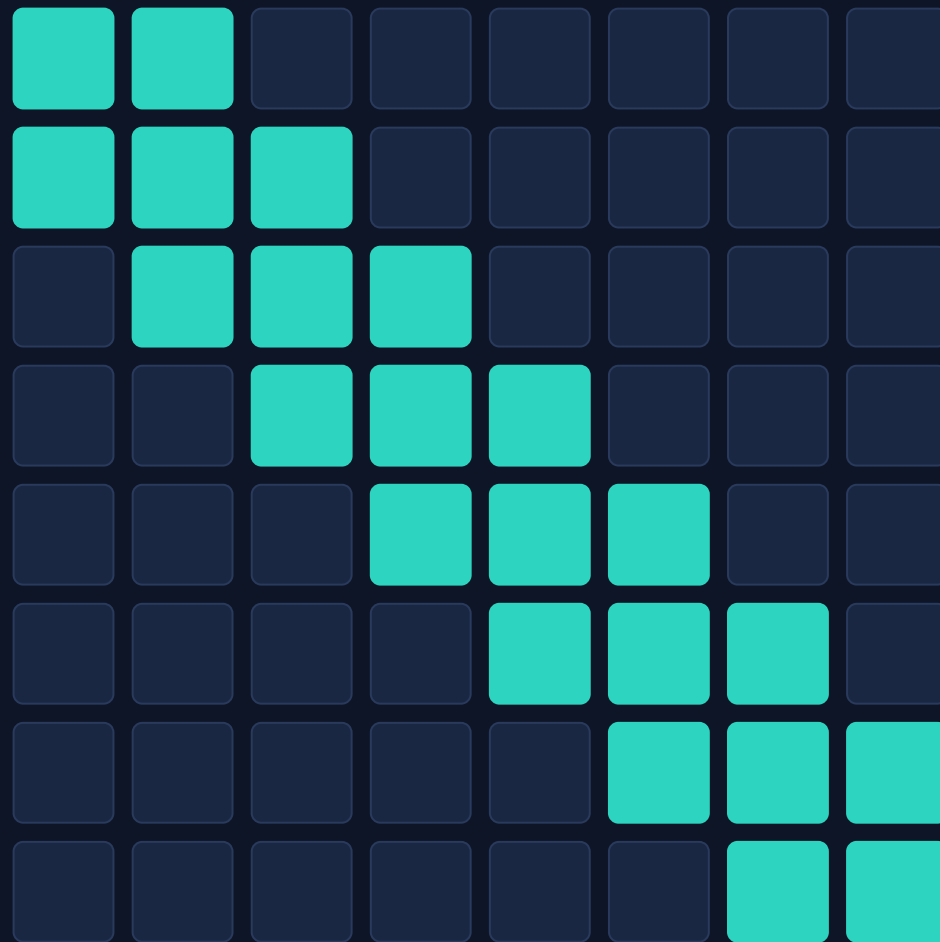


INFERENCE-TIME COMPUTE HACKATHON

Fairchild

Devin For Chip Design





Motivation

- Current LLMs, while powerful, often struggle with the rigorous syntax and logical precision required for Verilog/RTL.
- Chasing subtle timing bugs and race conditions leads to significant "token spend" and developer fatigue in manual debugging loops.
- There is a critical need for an agentic system that automates verification and repair.
- Closing the loop between generation and simulation is the only way to achieve production-grade hardware design.
- **On-premise deployment remains the primary choice for semiconductor teams to maintain IP security and control.**



What Did We Build?

The Vision

A fully integrated IDE


Under the Hood

- Hackathon Scope: **RTL Generation + Verification**
- **Qwen 2.5 7B**
Post Trained RTL Coder
- **Synthetic RTL SFT**
Advanced Data Mix
- **Agentic Harness**
Generation and Verification

Closing the loop between hardware design and automated simulation.



What Did We Build?



Model

Adapter ?

- base
- tuned
- tunedv6
- tuned+repair
- fallback

Server live · base, tuned, tunedv6

http://31.56.109.42:8000/v1 · auth: bearer

● live last → tuned in 3243 ms

Use cached/fixture data ?

[ESTIMATE] = analytical estimate

[MEASURED] = real tool result

Fix my RTL

FairChild IDE

Deploy ⋮


Hi — I'm the FairChild IDE. Describe a chip you want, and I'll compile a spec, propose A/B/C hardware candidates, estimate per-token throughput, write RTL, run Verilator, and explain the trade-offs. Click View results under any of my replies to open a side panel with the artifacts for that turn. Refine by saying things like "lower power to 3W".

Describe a chip, refine an existing run, or paste RTL in a ```systemverilog block to fix it · current model: tuned

Closing the loop between hardware design and automated simulation.



What Did We Build?


Deploy ⋮

FairChild IDE

🗄️ Hi — I'm the FairChild IDE. Describe a chip you want, and I'll compile a spec, propose A/B/C hardware candidates, estimate per-token throughput, write RTL, run Verilator, and explain the trade-offs. Click **View results** under any of my replies to open a side panel with the artifacts for that turn. Refine by saying things like "lower power to 3W".

🔄 Design an INT8 transformer decoder accelerator for on-device inference. Batch 1, seq 2048, hidden 1024, 16 heads. Power budget under 1.5W.

🗄️ Candidate A_low_power won with 358 tok/s due to its low power consumption of 1.2W, despite being memory-bound. The Verilator lint passed, and the user should consider using A_low_power for the design.

A_low_power · 358 tok/s [ESTIMATE] · 1.2W [ESTIMATE] · Verilator PASS [MEASURED]

📊 View results (turn #1)

Describe a chip, refine an existing run, or paste RTL in a ````systemverilog` block to fix it · current model: tuned ➤

Model

Adapter ?

- base
- tuned
- tunedv6
- tuned+repair
- fallback

Server live · base, tuned, tunedv6

http://31.56.109.42:8000/v1 · auth: bearer

● live last → tuned in 3243 ms

Use cached/fixture data ?

[ESTIMATE] = analytical estimate


[MEASURED] = real tool result

🔧 Fix my RTL

Closing the loop between hardware design and automated simulation.



What Did We Build?


Deploy ⋮

Model

Adapter ?

- base
- tuned
- tunedv6
- tuned+repair
- fallback

Server live · base, tuned, tunedv6

<http://31.56.109.42:8000/v1> · auth: bearer

● live last → tuned in 3829 ms

Use cached/fixture data ?

[ESTIMATE] = analytical estimate
[MEASURED] = real tool result

🔧 Fix my RTL

FairChild IDE

🗨️ Hi — I'm the FairChild IDE. Describe a chip you want, and I'll compile a spec, propose A/B/C hardware candidates, estimate per-token throughput, write RTL, run Verilator, and explain the trade-offs. Click **View results** under any of my replies to open a side panel with the artifacts for that turn. Refine by saying things like "lower power to 3W".

🚫 Design an INT8 transformer decoder accelerator for on-device inference. Batch 1, seq 2048, hidden 1024, 16 heads. Power budget under 1.5W.

🗨️ Candidate A_low_power won with a token rate of 358 tok/s due to its low power consumption of 1.2W, making it the most power-efficient option within the 2W budget. The Verilator lint passed without issues. For further optimization, consider exploring design modifications or optimizations

Describe a chip, refine an existing run, or paste RTL in a ````systemverilog` block to fix it · current model: tuned

✕ Close

Design RTL Verification Report

Spec

▶ { . . . }

Workload kernels [ESTIMATE]

| | name | M | N | K | macs | bytes |
|---|------|-------|-------|-------|---------------|-----------|
| 0 | gemm | 2,048 | 1,024 | 1,024 | 4,294,967,296 | 5,242,880 |

Hardware candidates


| | id | array_dim | sram_mb | freq_hz | precision |
|---|--------------------|-----------|---------|-------------|-----------|
| 0 | A_low_power | 32 | 2 | 750,000,000 | INT8 |
| 1 | B_balanced | 64 | 4 | 950,000,000 | INT8 |
| 2 | C_high_performance | 128 | 9 | 950,000,000 | INT8 |

Evaluation [ESTIMATE]

Closing the loop between hardware design and automated simulation.



What Did We Build?



Model

Adapter ⓘ

- base
- tuned
- tunedv6
- tuned+repair
- fallback

Server live · base, tuned, tunedv6

http://31.56.109.42:8000/v1 · auth: bearer

● live last → tuned in 3829 ms

Use cached/fixture data ⓘ

[ESTIMATE] = analytical estimate

[MEASURED] = real tool result

🔧 Fix my RTL

FairChild IDE

🗨️ Hi — I'm the FairChild IDE. Describe a chip you want, and I'll compile a spec, propose A/B/C hardware candidates, estimate per-token throughput, write RTL, run Verilator, and explain the trade-offs. Click **View results** under any of my replies to open a side panel with the artifacts for that turn. Refine by saying things like "lower power to 3W".

🗨️ Design an INT8 transformer decoder accelerator for on-device inference. Batch 1, seq 2048, hidden 1024, 16 heads. Power budget under 1.5W.

🗨️ Candidate A_low_power won with a token rate of 358 tok/s due to its low power consumption of 1.2W, making it the most power-efficient option within the 2W budget. The Verilator lint passed without issues. For further optimization, consider exploring design modifications or optimizations

Deploy ⓘ

Results for turn #1 · 16:01:24 ✕ Close

Design RTL Verification Report

Source: live tuned in 31797 ms

generated/rtl/run/mac_cell.sv

```

module mac_cell (
    input clk,
    input rst_n,
    input en,
    input clear,
    input signed [DATA_W-1:0] a,
    input signed [DATA_W-1:0] b,
    output reg signed [ACC_W-1:0] acc
);

parameter DATA_W = 8;
parameter ACC_W = 32;

always_ff @(posedge clk or negedge rst_n) begin
    if (!rst_n || clear) begin


```

Describe a chip, refine an existing run, or paste RTL in a `` systemverilog block to fix it · current model: tuned ➤

Closing the loop between hardware design and automated simulation.



What Did We Build?



Model

Adapter ?

- base
- tuned
- tunedv6
- tuned+repair
- fallback

Server live · base, tuned, tunedv6

`http://31.56.109.42:8000/v1` · auth: bearer


• live last → tuned in 3524 ms


Use cached/fixture data ?

[ESTIMATE] = analytical estimate
[MEASURED] = real tool result

[Fix my RTL](#)


Deploy ⋮


 Design an INT8 transformer decoder accelerator for on-device inference. Batch 1, seq 2048, hidden 1024, 16 heads. Power budget under 1.5W.

 Candidate A_low_power won with a token rate of 358 tok/s due to its low power consumption of 1.2W, making it the most power-efficient option within the 2W budget. The Verilator lint passed without issues. For further optimization, consider exploring design modifications or optimizations specific to the transformer_decoder architecture.

A_low_power · 358 tok/s [ESTIMATE] · 1.2W [ESTIMATE] · Verilator PASS [MEASURED]

[View results \(turn #1\)](#)

 increase the power to 5W

 Candidate A_low_power won due to its highest token throughput of 1,526 tok/s while staying within the 5W power cap. The Verilator lint passed with no issues. To improve further, consider exploring optimizations like loop unrolling or revisiting the design for potential bottlenecks.

A_low_power · 1,526 tok/s [ESTIMATE] · 2.2W [ESTIMATE] · Verilator PASS [MEASURED]

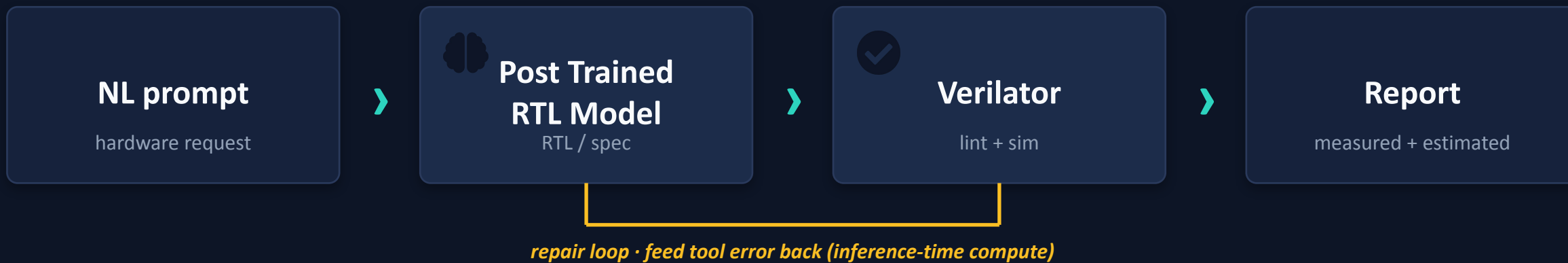
[View results \(turn #2\)](#)

Describe a chip, refine an existing run, or paste RTL in a `` systemverilog block to fix it · current model: tuned ➤

Closing the loop between hardware design and automated simulation.



The model proposes. The tools evaluate. The IDE explains.



LAYER 1 — the contribution

Post-trained 7B chip model + a reproducible benchmark harness over a held-out set.

LAYER 2 — the demo

Agentic IDE that runs the model inside the loop — and keeps measured vs estimated strictly separate.

Same pattern as an agentic SWE tool (Cognition) — act, read tool output, repair — **applied to RTL.**



The model

Post-training Qwen2.5-Coder-7B for RTL

Base coding LLMs write plausible-looking but often non-synthesizable Verilog. We domain-adapt with LoRA SFT (bf16) on a three-task RTL mixture.

Base `Qwen/Qwen2.5-Coder-7B-Instruct` Adapter `LoRA r=16, bf16`

SFT mixture — 39,151 examples



spec2rtl

32.2k 82%

Public corpora — description → code.
MG-Verilog (MIT) 16.2k + RTL-Coder Resyn27k
16.1k.



repair

5.8k 15%

inject bug → run tool → capture real error → diff
fix.
Lint (Verilator) + semantic (iverilog) buckets.

SYNTHETIC · tool-verified



explain

1.2k 3%

Code → natural-language description.
Inverted from MG-Verilog modules.

Decontaminated against the 362-task held-out set (VerilogEval + RTLLM) — 0 source-hash overlaps. Resyn27k is model-generated / non-commercial — flagged in the report.



Training results

Held-out SFT validation: base vs tuned

| model | eval loss | perplexity | token acc |
|--------------------------|--------------|-------------|--------------|
| base — Qwen2.5-Coder-7B | 0.490 | 1.63 | 88.6% |
| tuned — v5 (LoRA) | 0.245 | 1.28 | 92.7% |

-50%

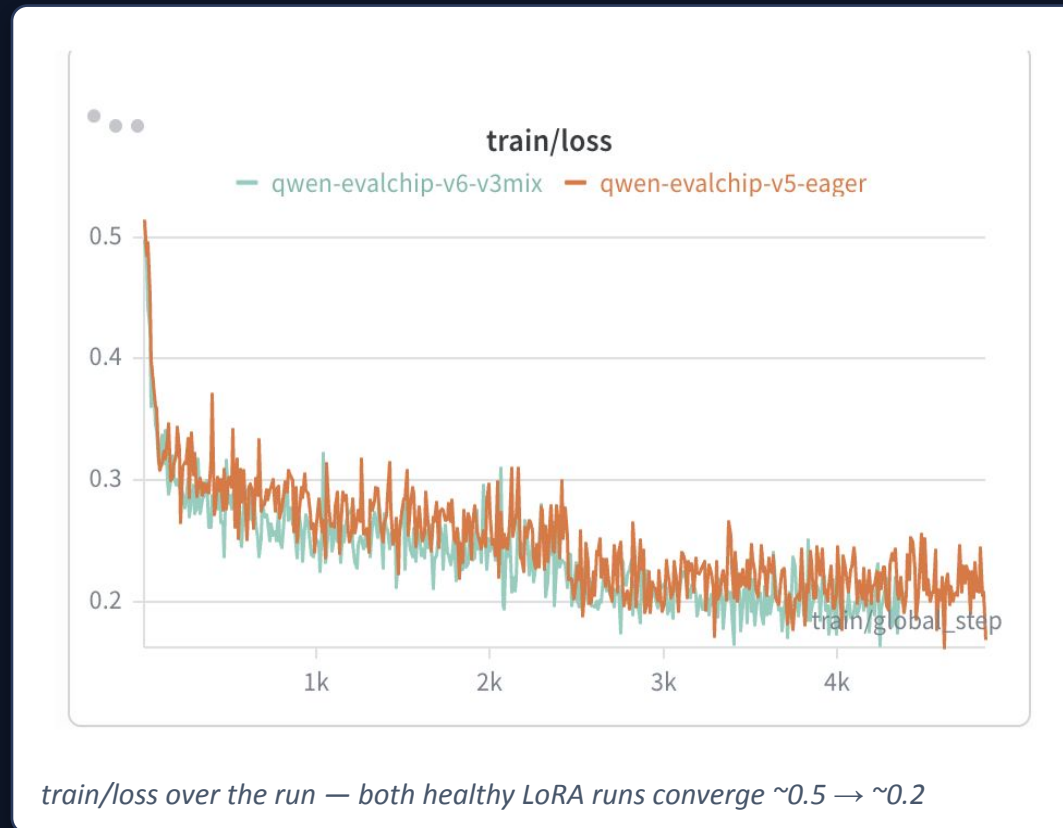
held-out eval loss

+4.1 pp

next-token accuracy



1× H100 80GB · bf16 LoRA · ~2 epochs (~4.8k steps) · ≈ 4 h
 ~4 runs to land a clean one — H100 bf16 attention-backward NaN,
 fixed with eager attention.



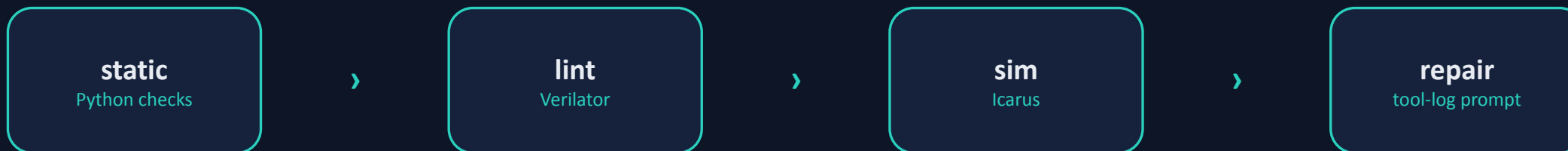
This is the SFT fit metric (teacher-forced, assistant-masked LM loss on the same val split) — necessary but not sufficient for pass@k, which the Verilator harness scores separately. v5 measured at ~0.7 epoch.



An Agentic Harness For Generation and Verification

Implemented harness: static checks, Verilator lint, Icarus simulation, and repair re-score.

IMPLEMENTED HARNESS fail-fast, cached, each stage timed



🔄 **Repair loop:** lint/sim failure → tool log + RTL → repair prompt → regenerated RTL → re-score

AROUND THE PIPELINE

Tool scoring API

Verilator/Icarus returns compile/test pass plus raw logs.

Repair benchmark

Failed greedy outputs are re-prompted with tool logs, then re-scored.

Typed result contract

Each run saves pass/fail, tool, errors, metrics, raw RTL, and logs.

Benchmark outputs

Harness writes table.md/csv, per-sample RTL, eval.json, and raw completions.

*Honesty contract: a result is marked **measured** only when a real tool actually ran.*



An Agentic Harness For Generation and Verification - Eval Results

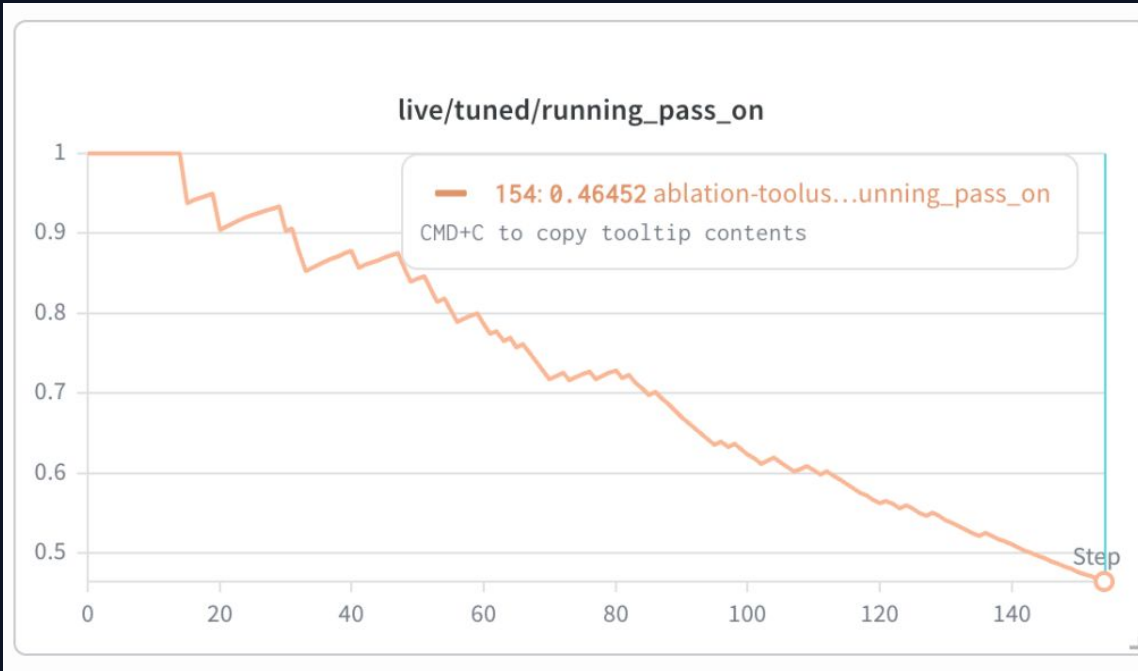
Validation set: 155 held-out VerilogEval spec-to-RTL tasks (data/eval_tasks.jsonl), scored with Icarus/Verilator testbenches.



| model | compile | test | pass@1 | pass@3 |
|-----------|-------------|-------------|-------------|-------------|
| base | 0.81 | 0.35 | 0.35 | 0.46 |
| v5 | 0.89 | 0.38 | 0.38 | 0.50 |



Ablation - tool use(w/o the harness)



The running_pass_off is one shot generation

A