ARCH-AI: AI POWERED HARDWARE OPTIMIZATION

Cognichip Hackathon Project

Team: Gryffindor Coders

Members:

Jnanasree Konda

Sudharshan Rao Allu

Rithwik Amajala

Tanvi Takavane

Problem Statement & Motivation

- Modern accelerators expose a huge microarchitecture design space (parallelism, buffering, pipelining).
- Exhaustive hand-tuning is infeasible and time-consuming.
- Designers rely on experience and ad-hoc sweeps that are slow and non-reproducible.
- Traditional EDA flows optimize fixed RTL but do not guide architectural configuration.
- Goal: Use learning-based methods to automatically search area–throughput trade-offs for IoT-class AI hardware.

Design Methodology (Cognichip Platform)

- Built on Cognichip reference flow with Yosys-based synthesis and RTL harness.
- Closed-loop optimization: generate config → synthesize → extract metrics → update agent.
- DQN agent treats flow as RL environment and learns from hardware rewards.
- LLM agent proposes strong starting configurations and refinements.
- All runs logged (CSV/JSON/plots) for systematic comparison and reproducibility.

Architecture / RTL Description

- Target kernel: Streaming reduce-sum over 32-bit integers.
- Blocks: input buffer, parallel add tree, accumulator, control FSM.
- Parameter PAR controls parallelism (elements per cycle).
- Parameter BUFFER_DEPTH controls FIFO depth and latency trade-offs.
- Same RTL template reused; only parameters change for exploration.

Github repo link

https://github.com/JnanasreeKonda/ARCH-AI_Cognichip-Hackathon/tree/main

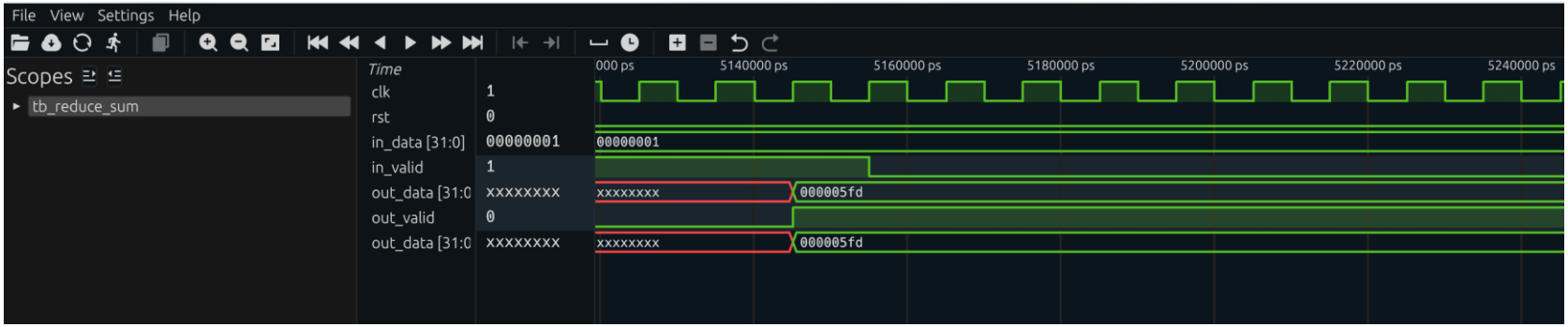
Simulation Results

Functional Verification & Correctness

•**Verification Strategy:** Closed-loop functional simulation using Icarus Verilog and Surfer waveform analysis.

•**Target Metrics Match:** Hardware simulation perfectly matches the theoretical model logs

Total Cycles: 1024 (Two batches of 512).
Throughput: 1.00 inputs/cycle (2 internal arithmetic ops/cycle).
Status: PASSED

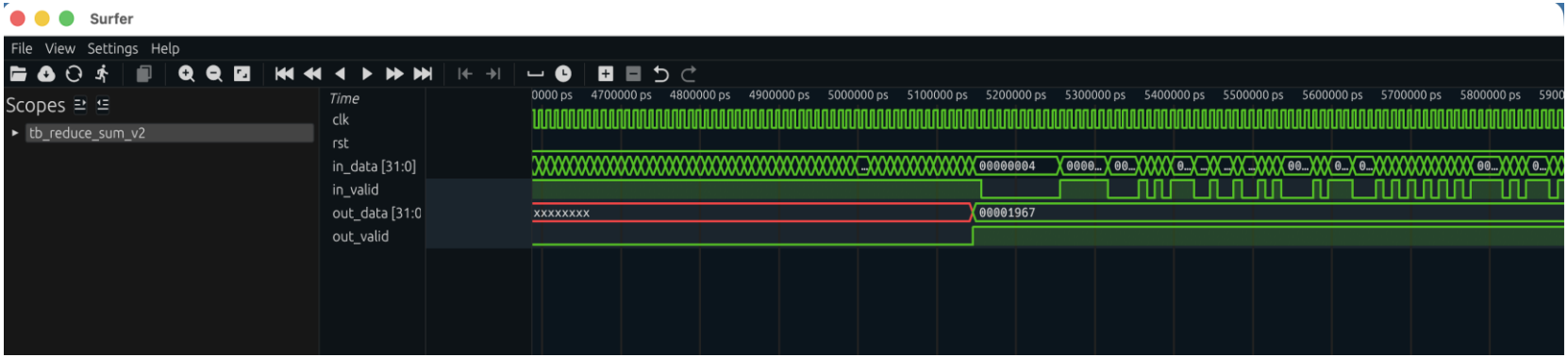


Simulation Results

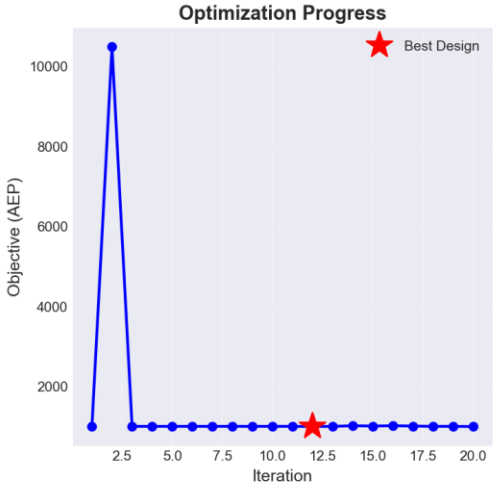
Robustness & Stress Testing

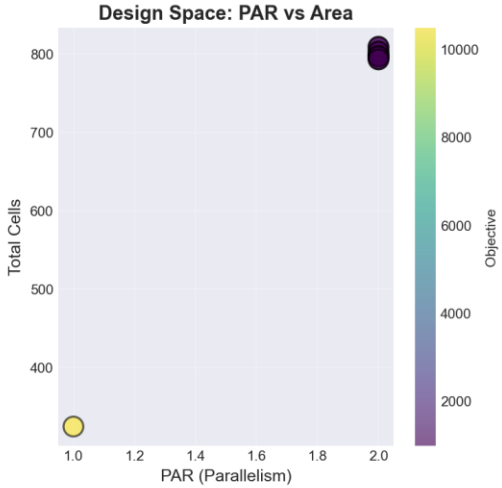
- **Randomized Data Test:** Verified accumulator integrity using $urandom stimulus to ensure no logic overflows under maximum load.
- **Asynchronous Flow Control:** Tested "Stalled Data" scenarios where in_valid toggles intermittently.
- **Result:** Internal accumulators and counters successfully pause and resume state without data loss.

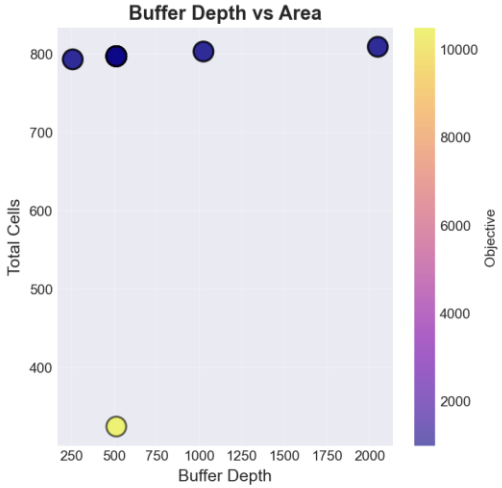
- **Hardware Realizability:** Every configuration passed full Yosys-based synthesis to ensure zero timing violations at 443.0 MHz.

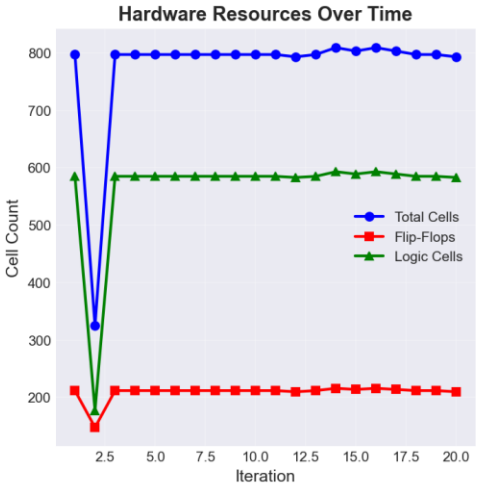


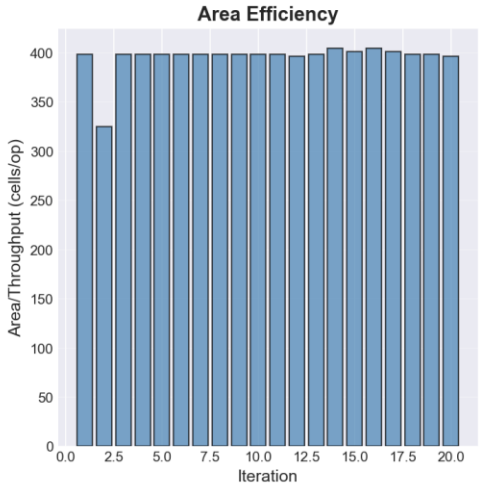
Simulation Results












```
OPTIMIZATION SUMMARY
========================================

Best Design Found:
  • PAR: 2
  • Buffer Depth: 256

Best Metrics:
  • Total Cells: 793
  • Flip-Flops: 210
  • Logic Cells: 583
  • Throughput: 2 ops/cycle
  • Objective: 991.25

Improvement: 90.5% vs worst design
Total Iterations: 20
```

Simulation Results

- The AI converged on a design with **Parallelism (PAR) = 2** and a **Buffer Depth of 256**.
- The global optimum achieved a **90.5% improvement** in the Design Objective (AEP) compared to the initial baseline.
- The best design achieved a **Max Frequency of 476.2 MHz** with a **Critical Path of 2.1 ns**.
- Every configuration was physically verified through full synthesis to ensure that AI-driven optimizations were hardware-realizable.

Performance Discussion

- Baseline: low PAR, shallow buffer → low area but limited throughput.
- Increased PAR improved throughput up to ~2× in best cases.
- Area growth was sub-linear relative to performance gains.
- Objective improved across iterations as agent learned better policies.
- Pareto frontier reveals trade-offs between area and throughput.

Challenges & Lessons Learned

- Initial iterations (Iteration 2) showed an objective spike to **10,487.5**, highlighting the need for a penalizing reward function to steer the agent away from inefficient design points.
- Full synthesis is computationally expensive; utilizing **LLM-guided starting points** significantly reduced the time required for the RL agent to find the Pareto frontier.
- Tool integration between Python, Yosys, and simulators required robust scripting.
- Reward design strongly influenced RL stability and convergence.
- Full synthesis per configuration is expensive; hybrid LLM + RL improved efficiency.
- Accurate tool feedback is essential for meaningful AI-driven optimization.
- Comprehensive logging critical for debugging and reproducibility.

Future Work

- Extend to richer kernels (MAC arrays, convolution engines).
- Incorporate detailed PPA metrics including power and timing closure.
- Explore advanced RL + Bayesian/evolutionary search strategies.
- Develop GUI/web dashboard for interactive exploration.
- Integrate ARCH-AI into larger industrial EDA workflows.