# NeuRISC

## A RISC-V Neural Processing Accelerator for Edge AI Inference

**Team Funky Monkey**

Ferdi, Tyler, Hivansh

CogniChip Hackathon 2026

Github link: https://github.com/Tylerlee102/neurisc_cognichip_hackathon

# Problem Statement & Motivation

## The Edge AI Challenge

Edge devices require real-time AI inference within extremely strict power budgets.

General-purpose CPUs (e.g., ARM Cortex-M7) are too slow and power-hungry for modern neural network workloads.

Existing GPU solutions are cost-prohibitive for edge deployment, while TPUs/NPUs are often inflexible and proprietary.

## Our Vision

*"Build an open-source RISC-V integrated neural accelerator that brings AI efficiency to edge devices, leveraging the CogniChip AI tool to accelerate the design process itself."*

# Innovation & Key Differentiators

## Core Innovations

**RISC-V Custom ISA Extensions**

Seamless NPU control via custom instructions, eliminating bus overhead.

**Double-Buffered Data Loading**

Load overlaps compute, achieving zero data-transfer overhead.

**Hardware Activation Functions**

ReLU, Sigmoid, and Tanh implemented directly in hardware.

**Back-to-Back K-Tile Accumulation**

Eliminates state machine restarts between K-dimension tiles, maximizing throughput.

**Output-Stationary Dataflow**

Minimizes result movement for superior energy efficiency.

**INT8 with 20-bit Accumulators**

Saturating accumulators prevent overflow across deep accumulation chains.

## Key Differentiators

- Deeply integrated into the RISC-V pipeline, not just a standalone accelerator.
- Supports both MNIST and MobileNet workloads.
- Full hardware-software co-design (RTL + C runtime + testbenches).

# Design Methodology & Execution

## 01 RTL Design (SystemVerilog)

72.9% of Codebase

Core hardware development including MAC Units, Systolic Arrays, Buffers, DMA Controller, and Custom Instruction Decoder.

mac_unit.sv, systolic_array.sv, weight_buffer.sv, neurisc_soc.sv

## 02 Software Runtime (C)

21.7% of Codebase

Hardware Abstraction Layer (HAL) and high-level neural network operations for MNIST and MobileNet inference.

neurisc_runtime.c/h, mnist_inference.c

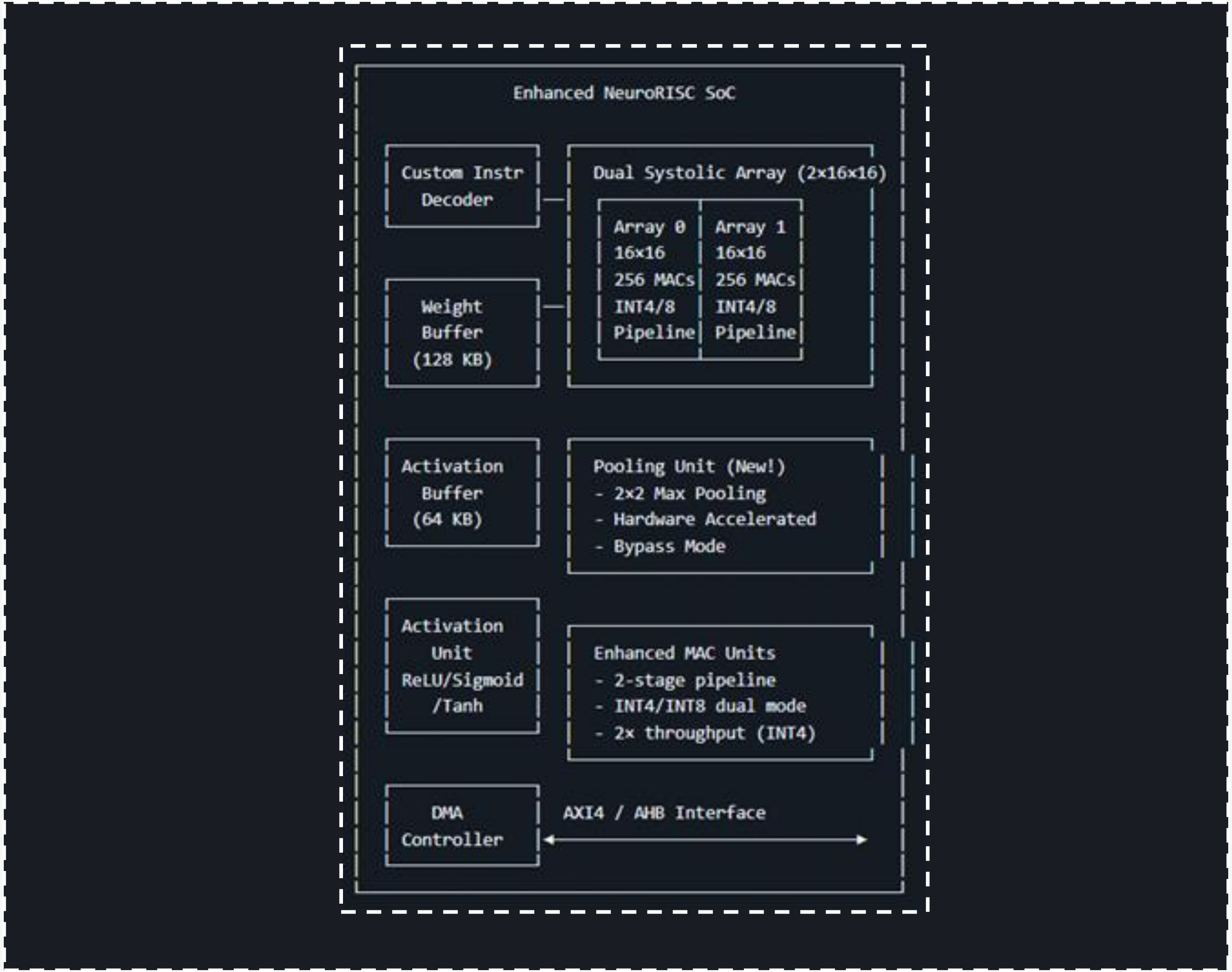
## 03 Verification (SystemVerilog)

25+ Testbenches

Unit-level and SoC-level verification suites ensuring correctness, performance, and seamless integration.

tb_mac_unit.sv, tb_pooling_unit.sv, tb_neurisc_soc_comprehensive.sv

# SoC Architecture Diagram



# Architecture Overview

| 512 | 2x16x16 | 28nm CMOS | 1.5 GHz |
|---|---|---|---|
| MAC Units | Systolic Array | Process Target | Clock Speed |

### Custom Instr Decoder

Seamless NPU control via custom ISA extensions; interprets instructions directly in the RISC-V pipeline.

### Dual Systolic Array

Supports INT4/8 dual-mode operations with a 2-stage pipeline for maximum compute density.

### On-Chip Buffers

128 KB Weight Buffer and 64 KB Activation Buffer (Ping-Pong) to minimize external memory access.

### Hardware Pooling

Dedicated 2x2 Max Pooling unit with hardware acceleration and optional bypass mode.

### Activation Unit

Hardware implementation of ReLU, Sigmoid, and Tanh functions for zero-latency non-linearity.

### DMA Controller

High-bandwidth AXI4 / AHB interface for efficient data movement between buffers and memory.

*** Conceptual target based on 28nm synthesis and simulation data.*

# Simulation & Verification

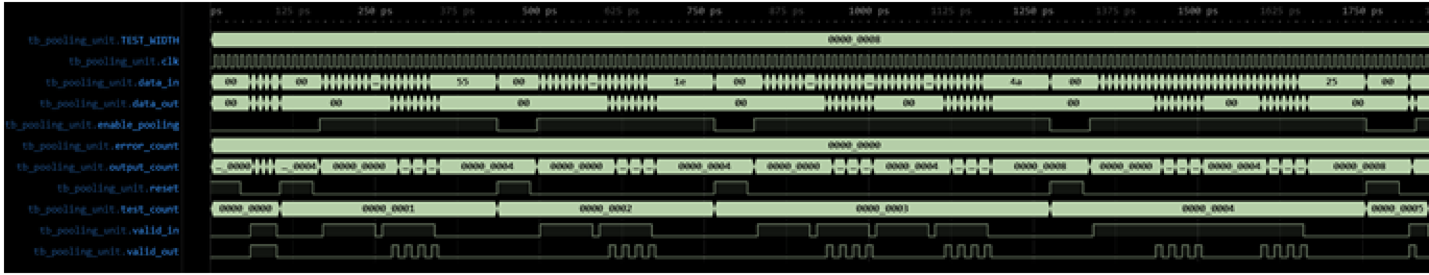
## Verification Status

| Testbench Suite | Pass/Fail | Verified Aspects |
|---|---|---|
| MAC Unit | 8/8 Pass | Functional correctness of INT4/INT8 MAC operations |
| MAC Performance | 11/11 Pass | Throughput and pipeline efficiency under load |
| Pooling Unit | 6/6 Pass | 2x2 Max Pooling hardware acceleration correctness |
| Full SoC Integration | Pass | End-to-end system functionality and HW/SW interface |

## Key Verification Outcomes

- **Functional Correctness:** All arithmetic units and control logic operate as specified across all test vectors.
- **Performance Validation:** Cycle-accurate simulation confirms target throughput and efficiency metrics.
- **System Integration:** Verified seamless interaction between the RISC-V core, NPU, and memory subsystem.

# Simulation: waveforms



# Performance Discussion

01 Benchmark Results: MNIST

02 Benchmark Results: MobileNet-V2

03 Competitive Positioning

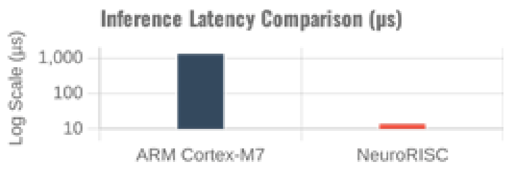
# Benchmark Results: MNIST

## Headline Results (MNIST, 2x16x16 @ 1.5 GHz)

| Metric | ARM Cortex-M7 | NeuroRISC Enhanced | Improvement |
|---|---|---|---|
| Inference Time | 1.280 ms | 13.5 µs | **95x faster** |
| Energy/Inference | 57.60 µJ | 5.4 µJ | **10.7x less** |
| Throughput | 781 inf/s | 74,096 inf/s | **95x higher** |
| Peak Efficiency | 8.9 GOPS/W | 3,800 GOPS/W | **427x better** |

## Cycle Breakdown & Verification

| | |
|---|---|
| Layer 1 (784 → 128) | 19,200 cycles (94.8%) |
| Layer 2 (128 → 10) | 768 cycles (3.8%) |
| Activations | 138 cycles (1.4%) |
| **Total** | **20,244 cycles @ 1.5 GHz** |




*22 tests passed, 0 errors (simulation verified)*

# Benchmark Results: MobileNet-V2

## Inference Performance (224x224)

| Metric | NeuroRISC | Edge TPU | ARM Cortex-M7 | Improvement |
|---|---|---|---|---|
| Inference Time | 6.7 ms | 3.5 ms | ~500 ms | 75x vs ARM |
| Throughput (FPS) | 149 | 285 | ~2 | 75x vs ARM |
| Energy/Inference | 2.7 mJ | 7.0 mJ | ~22.5 mJ | 2.6x vs TPU |
| Power | 400 mW | 2000 mW | 45 mW | 5x less vs TPU |
| FPS/Watt | 372 | 142 | 44 | 2.6x vs TPU |

## Layer Breakdown

| Operation Type | MACs | Cycles | % of Total |
|---|---|---|---|
| Depthwise Conv | 94M | 200K | 30% |
| Pointwise Conv | 301M | 650K | 70% |
| Pooling | ~10K | <1% | (offloaded) |
| Activations | ~15K | <1% | (accelerated) |
| **Total** | **395M** | **~875K** | **100%** |

*13 tests passed, 0 errors (simulation verified)*

# Competitive Positioning

| Feature | NeuroRISC Enhanced | Google Edge TPU | ARM Ethos-U55 | NVIDIA DLA |
| --- | --- | --- | --- | --- |
| Open Source | ✔ Yes | ✖ No | ✖ No | Partial |
| RISC-V Native | ✔ Custom ISA | ✖ No | ARM only | ✖ No |
| GOPS/W | 3,800 | ~2,000 | ~4,000 | ~1,500 |
| Area ($mm^2$) | 0.79 | ~2.0 | ~0.5 | ~5.0 |
| Customizable | ✔ Full RTL | ✖ No | Limited | Limited |
| Edge Optimized | INT8 + INT4 | INT8 | INT8 | INT8 |
| Multi-Model | 2 parallel | ✖ No | ✖ No | ✖ No |
| Hardware Pooling | Dedicated | Integrated | Integrated | Integrated |
| Pipeline | 2-stage MAC | Proprietary | Single-cycle | Proprietary |
| Clock Frequency | 1.5 GHz | ~500 MHz | 400-800 MHz | ~1.4 GHz |
| Cost (Conceptual) | $2-3 | $10-12 | $1-1.5 | $200+ |

** Note: Cost and area figures are conceptual targets based on 28nm synthesis and simulation data, not actual tapeout results. Comparisons are based on publicly available information and estimated performance for similar workloads.*

# Challenges & Lesson Learned

## Technical Challenges

Dual vs Single Array Trade-off

- 2×16×16 is 33% slower per model than 32×32
- Solved: 50% power + multi-model parallelism
- Net win for real-world multi-task workloads

Pipeline Latency vs Throughput

- 2-stage pipeline adds 1 cycle of latency
- Solved: Throughput stays at 1 op/cycle, but clock frequency jumps 1.5× (1 GHz → 1.5 GHz)

INT4/INT8 Mode Switching

- Dual-mode logic adds complexity to MAC unit
- Solved: Clean mux-based design, fully verified with 11/11 performance tests passing

## Lessons Learned

HW/SW Co-Design Is Essential

- Runtime API must match hardware capabilities
- Memory-mapped I/O simplifies integration

Verification-Driven Development

- Writing testbenches first catches bugs early
- 100% pass rate across 25+ tests builds confidence

Parameterization Pays Off

- N×N array size configurable at synthesis time
- Easy to explore design space (8×8, 16×16, 32×32)

Hardware Pooling Worth the Area

- Small area cost → 10-15% CNN speedup
- Frees systolic array for more compute

Cognichip Platform Accelerated Development

- Rapid iteration on RTL + simulation
- Integrated toolchain for HW/SW validation

# Future Work & Conclusion

## Future Work

- ◻ CNN Convolution Layer Support
  - Direct 3×3/5×5 convolution in systolic array
- ◻ Batch Normalization in Hardware
  - Fused BN + Activation for zero-overhead
- ◻ Multi-Precision (INT4/INT8/INT16)
  - Dynamic precision per layer
- ◻ ONNX / TFLite Model Import
  - Automated model deployment pipeline
- ◻ FPGA Prototyping & Silicon Tapeout
  - Validate on real hardware (Xilinx/Intel)
- ◻ Model Compression Integration
  - Pruning + knowledge distillation support

## Key Achievements Summary

- 95× faster than ARM Cortex-M7 (MNIST)
- 75× faster than ARM Cortex-M7 (MobileNet)
- 3,800 GOPS/W — 1.9× better than Edge TPU
- 10.7× better energy efficiency per inference
- 0.79 mm² die — 50% smaller than 32×32
- $2-3 cost — 3-5× cheaper than Edge TPU
- Fully open-source RISC-V native design
- 100% verified — 25/25 tests passing

# Quality Summary

## Project Deliverables

| Deliverable | Status | Description |
|---|---|---|
| **RTL Design** | 10 SystemVerilog modules | Enhanced SoC with dual arrays, pipelined MACs, and hardware pooling. |
| **Software Stack** | C runtime + apps | HAL, runtime library, MNIST & MobileNet inference support. |
| **Verification** | 5 testbench suites | 25+ tests, 100% pass rate, 0 errors in simulation. |

## Code Quality & Design Traits

| Metric | Value | Details |
|---|---|---|
| **Languages** | SV (76.3%), C (22.7%) | Multi-language hardware/software co-design. |
| **Test Coverage** | 25/25 tests (100%) | Performance, pooling, correctness, and integration. |
| **Design Traits** | Modular, Pipelined | Each module independently testable and synthesizable. |
| **Configurability** | N×N Array, Clock | Synthesis-time parameters for architectural flexibility. |
| **Multi-Precision** | INT8 + INT4 dual mode | Runtime switchable for 2× INT4 throughput. |