



Product Brief

Open-source architecture
IP & toolchain



Overview

Ultra-low-power edge devices such as wearables, hearables, AR glasses, and ambient IoT sensors require on-device machine learning within strict thermal and battery budgets. Achieving this performance historically required navigating compromises between rigid proprietary accelerators with high licensing overhead, and custom in-house hardware with burdensome compiler maintenance.

Coral NPU provides a verified, open-source alternative that is intended to be a base reference design for the industry to adopt and extend.

Developed by Google Research and co-designed with Google DeepMind, Coral combines an efficient RISC-V accelerator NPU with an open-source software toolchain. Coral provides a drop-in subsystem optimized for ambient sensing models and compact generative workloads.



Why Coral?

Complete subsystem delivery package

Coral delivers a hardware and software stack ready for evaluation and integration:

- **Synthesizable soft IP:** Parameterized Chisel and Verilog source code for scalar, vector, and matrix engines with standard 128-bit AXI4 interconnect.
- **Verification and test environment:** Comprehensive design verification (DV) test suites and FPGA emulation configurations.
- **Dual simulation targets:** Behavioral instruction simulator (MPACT-CoralNPU) for rapid software bring-up, and cycle-accurate Verilator models for exact timing and hardware validation.
- **Turnkey compiler and runtime:** Open-source MLIR/IREE compiler, runtime HAL, and device drivers supporting PyTorch, JAX, and LiteRT.

Open-source and royalty-free (Apache 2.0)

Released under the permissive Apache 2.0 license, Coral NPU carries zero upfront license fees and zero unit royalties. Adopter engineering teams have full RTL and compiler visibility to inspect, configure, and extend the design for specific application requirements without vendor lock-in.

Co-designed with Google DeepMind

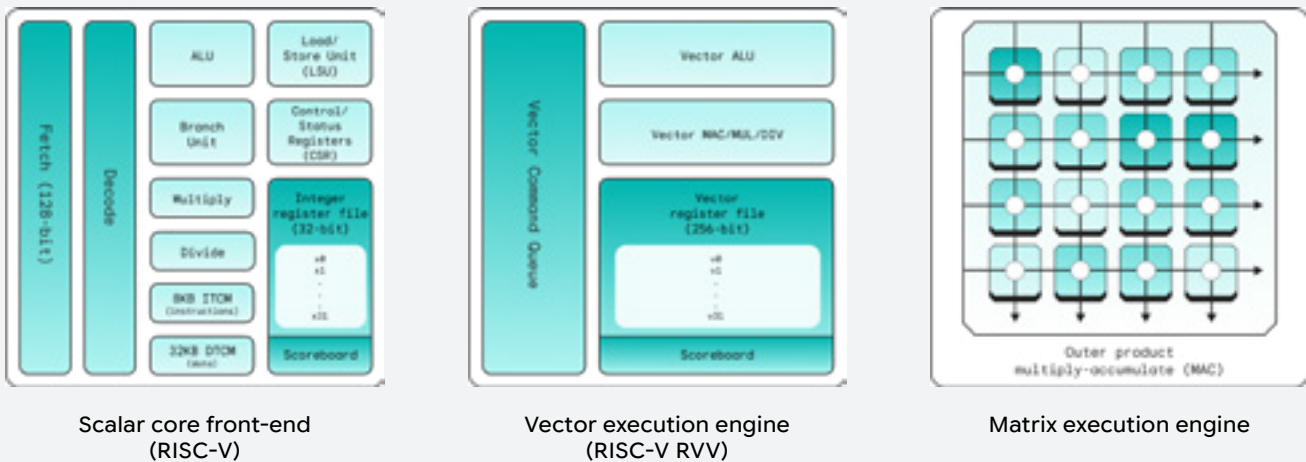
The microarchitecture is co-developed alongside Google DeepMind and Google Research ML teams. The instruction set and data pathing are designed to run production ambient and compact generative AI models from Gemma (270M/500M) and Lyria (audio neural codec) to CNNs and MobileNets.

Standards-based RISC-V foundation

Coral NPU implements the ratified RISC-V instruction set architecture (ISA). By using standard RISC-V scalar, vector (RVV), and proposed matrix extensions (VME) rather than a proprietary VLIW or microcode instruction set, Coral integrates seamlessly with standard embedded tools, debuggers, and runtime environments.

Coral NPU Architecture

Coral NPU includes three distinct processor components that work together: matrix, vector (SIMD), and scalar. This approach keeps data local, eliminates CPU-to-NPU communication bottlenecks, and sustains peak efficiency across both classical CNNs and modern transformer architectures.



Scalar frontend (RV32IMF)

- **Execution and dispatch:** A four-stage, in-order dispatch pipeline (four-way scalar, two-way vector) with out-of-order retirement. This 32-bit core operates as a lightweight run-to-completion executor, managing loop counters, branch control, and exceptions without requiring operating system or interrupt overhead.
- **Command forwarding:** Directly drives instruction queues for the vector and matrix backends, eliminating host CPU coordination during kernel execution.

Vector execution engine (SIMD)

- **Math and activations:** Implements the RISC-V vector specification (RVV 1.0 zve32f/zve32x) with a 128-bit SIMD, 256-bit pipeline. This executes General Matrix-to-Vector Multiplications (GEMV) and natively evaluates non-linear layers (Softmax, SwiGLU, LayerNorm, GELU) where matrix hardware struggles.
- **Data precision:** Supports integer (int8, int16) and floating-point formats (FP32, FP16, BF16), ensuring full operator coverage for PyTorch, JAX, and LiteRT models.
- **Bandwidth scalability:** Features a parameterizable vector length (VLEN = 128, 256, 512, up to 1024). Wider vector configurations scale memory bandwidth to support weights for compact LLMs like TinyGemma 270M directly on-device.

Matrix execution engine (outer-product GEMM)

- **Compute density:** A high-throughput General Matrix-to-Matrix Multiplication (GEMM) engine performing 256 MACs per-clock-cycle via four 8-bit multiplies reduced into 32-bit accumulators.
- **Two-dimensional broadcast:** Features a parallel 2D broadcast topology that routes parallel weights along one axis and transpose-shifted batch inputs along the other. This maximizes compute delivery per-memory-access, accelerating 2D convolutions and LLM pre-fill operations while minimizing SRAM read energy.

Technical Specifications

Spec	Parameter	Details, Metrics
<i>Instruction set</i>	Base & extensions	RV32IMF 32-bit RISC-V microcontroller base + Zve32f for RVV vector extensions + Zvt for Vector Matrix Extensions (VME)
<i>Pipeline & dispatch</i>	Core execution	4-stage pipeline; 4-way scalar dispatch, 2-way vector dispatch
<i>Supported numerics</i>	Data types	int8, int16, IEEE 754 FP32, FP16, and BF16
<i>Tightly-coupled memory</i>	Internal SRAM	8 KB ITCM and 32 KB DTCM (single-cycle latency; zero cache-miss penalty)
<i>System interconnect</i>	Bus interfaces	Dual AXI4 128-bit interfaces (master for DMA/memory, slave for host control)
<i>System control</i>	Power management	Interrupt signaling, Wait-For-Interrupt (WFI) clock gating, fault handling
<i>Debug infrastructure</i>	Hardware debug	JTAG TAP controller with standard GNU Debugger (GDB) support
<i>Software license</i>	Distribution	Permissive Apache 2.0 (royalty-free)

Software Toolchain and Compiler Architecture

Coral offers standards-based, easy-to-use developer tools that support multiple ML frameworks. The tools leverage existing compiler and library support from the broader RISC-V ecosystem, simplifying development. Coral NPU is the ideal hardware accelerator for running ML code developed with PyTorch, JAX, or LiteRT (formerly TensorFlow Lite).

The Coral platform also provides an open-source software stack built on MLIR and IREE, removing the need to develop proprietary toolchains. Key highlights of the Google MLIR compiler include:

- **Multi-framework ingestion:** Accepts models directly from PyTorch, JAX (via PJRT plugin and StableHLO), and LiteRT (via TOSA).
- **Automated graph optimization:** Applies constant folding, dead-code elimination, buffer reuse, loop tiling, and operator fusion matched to ITCM/DTCM boundaries.
- **Heterogeneous partitioning and host fallback:** Tensor operations mapped to Coral NPU run on the accelerator; any unsupported operators automatically fall back to execute on the host CPU, ensuring out-of-the-box model compatibility.
- **Standard LLVM codegen:** Lowers through MLIR Linalg dialects into standard LLVM with RISC-V target backends, emitting standard ELF binaries compatible with embedded toolchains.
- **Pre-silicon software enablement:** Embedded developers can compile models and profile execution cycle counts using the MPACT-CoralNPU and Verilator simulators before physical silicon is available.

Production Silicon and Ecosystem Support

Coral NPU is validated in commercial silicon and supported by established industry partners:

- **Production silicon:** Synaptics integrates the Coral NPU core in its Astra™ SL2610 AI-native IoT processor family (as the Torq™ NPU).
- **Prototyping hardware:** The Coralboard™ program offers low-cost evaluation boards for hardware prototyping and rapid software development. These boards are available via distributors like DigiKey.
- **Design and integration services:** Partners including VeriSilicon and Antmicro provide physical design services, IP hardening, custom bus bridging, and compiler adaptations.
- **Foundry compatibility:** Synthesizable design with reference implementation flows validated on GlobalFoundries 22FDX and TSMC advanced nodes.

Resources and Getting Started

- **RTL repository:** <https://github.com/google-coral/coralNpu>
- **Gemini code wiki:** <https://codewiki.google/github.com/google-coral/coralNpu>
- **Technical guides:**
 - Coral NPU datasheet (<https://developers.google.com/coral/guides/hardware/datasheet>)
 - IP evaluation guide (<https://developers.google.com/coral/guides/hardware/evaluation>)
 - IP integration guide (<https://developers.google.com/coral/guides/hardware/integration>)
- **Partnership inquiries:** Coral collaboration request form (<https://docs.google.com/forms/d/e/1FAIpQLScRig-ccAWjitNg7YZhnhHzUVsmUr4D9nw1fNwzIMnqTKGI1g/viewform?usp=header>)