Yongkang Cheng

chengyongkang.me | 437-663-2855 | iwmain@outlook.com | github.com/Ken-2511

September 26, 2025

Dear Tenstorrent Team,

I want to grow where compiler/runtime decisions meet custom RISC-V cores and on-die accelerators—the boundary your rotational role exposes daily. I am Toronto-based and fully available on-site. I bring a blend of: (1) cycle-aware FPGA / RTL prototyping, (2) fixed-point + signal-processing algorithm implementation, (3) disciplined scripting / regression habits, and (4) emerging formal/structured verification practices (now extending into SystemVerilog assertions and coverage thinking).

Hardware–software co-design mindset. In FPGA projects I iterate the same feature from three angles: a Python/C reference (clarify math + control flow), a synthesizable fixed-point datapath (resource + timing trade-offs), and a scripted simulation harness (repeatable stimulus + log diff). A 20-voice fixed-point audio engine (phase accumulators replacing floats) and a cycle-accurate VGA/PS2 subsystem on a Nios-V (RISC-V) soft core trained me to reason about latency hiding, register pressure, and memory layout while keeping deterministic timing contracts.

Algorithmic calibration → hardware control. In an ultra-wideband hybrid (2-PPM + 8-PSK) demod pipeline I implemented symbol timing (2 ns granularity), phase clustering (K-means stabilization), and recovery loops achieving zero-BER under 13 dB SNR with injected phase jitter. That experience strengthened a pattern directly relevant to accelerator / fabric tuning: instrument → model error sources → adapt parameters → re-measure.

Verification discipline in progress. I replaced one-off waveform poking with parameterized ModelSim scripts and am migrating selected Verilog modules to SystemVerilog (‘logic‘, ‘always$_f f/comb$‘, $simpleSV$
$C++harness.$

Runtime + performance intuition. A C++ pathfinding / spatial indexing project (multiple heuristics + data-layout tuning) and a CRNN model implementation (batching + segmentation pre-passes) sharpened reasoning about cache locality, branching behavior, and arithmetic intensity—concepts I can map to instruction mix and queue depth when characterizing custom cores or accelerator DMA strategies. I routinely write microbenchmarks (vector add, convolution kernel slices) to obtain ground-truth baselines before speculating about optimization.

Bring-up tooling mindset. In a wireless power transfer coil project I iterated 14 PCB revisions with structured parameter logs, measurement scripts, and test procedure checklists—habits I intend to extend to lab validation (systematic register sweep scripts, timestamped trace capture, failure triage notes). I am building a small host Python tool (I2C/SPI abstraction + diffable register snapshot) to accelerate future board bring-up and runtime counter harvesting.

Day-one value. I can: (1) implement concise RTL control/datapath stubs for early architectural experiments; (2) add diagnostic counters / tracepoints and surface them through a Python harness; (3) write fixed-point or int8 reference kernels for comparison against accelerator results; (4) script regression bundles (build + sim + metrics) to stabilize iteration; and (5) draft clear boundary documents (inputs, invariants, latency/throughput expectations) that de-risk cross-team handoff.

I am eager to deepen: memory subsystem profiling (LLC / scratchpad interplay), formal verification coverage strategy, and hardware scheduler / DMA orchestration patterns common to modern AI inference fabrics. Tenstorrent's exposure across compiler, runtime, and silicon makes it the ideal environment to accelerate that growth while contributing meaningful engineering rigor from day one.

Thank you for your consideration; I would welcome the opportunity to discuss how I can support your platform's firmware, verification, and performance characterization efforts.

Sincerely,

Yongkang Cheng