AI-Guided Memory Hierarchy Design For Edge LLM Inference

Hao Ding & Sichen Li

https://github.com/peylix/lm_memory_controller.git

Problem Statement & Motivation

Efficient memory hierarchy design is critical for enabling real-time, privacy-preserving LLM inference on edge devices under strict resource constraints.

- Edge LLM inference is memory-bound
- Limited on-chip SRAM, expensive off-chip DRAM
- Frequent movement of weights / activations / KV-cache
- Memory bandwidth and latency dominate performance & energy

Design Methodology

How We Used the Cognichip Platform:

AI - Guided Design Workflow

- Started from natural-language descriptions of:
  - Edge LLM inference constraints
  - Memory-bound workload characteristics
- Used Cognichip AI to propose multiple design configurations, including:
  - Buffer sizes and memory partitioning
  - Tile granularity and access patterns
  - Prefetch and data-movement strategies
- Selected representative configurations for implementation

Design Methodology

How We Used the Cognichip Platform:

RTL & Simulation Development

- Used AI assistance to bootstrap the RTL framework
- Implemented a parameterized memory controller design
- Built a cycle-level simulation testbench to:
  - Drive tiled workloads
  - Model realistic memory behavior
  - Validate functional correctness

Analytical Performance Model

Purpose

Reduce the RTL design search space under limited project time.

Optimization Method

Evaluated design dimensions:

- Buffer size allocation
- Tile granularity
- Prefetch depth and overlap strategy

Multi-objective evaluation using Pareto frontier:

- Minimize DRAM accesses
- Minimize compute idle cycles
- Constrain SRAM footprint

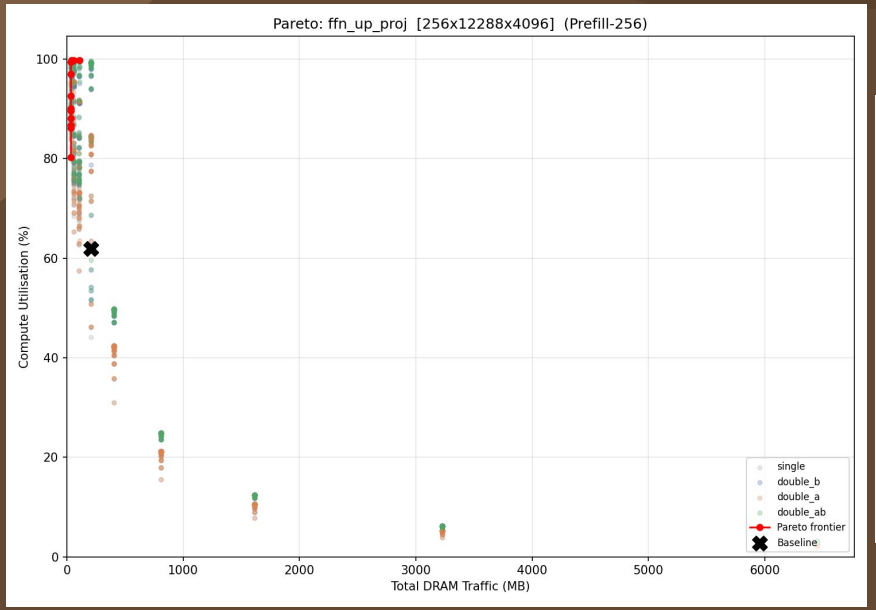
Model Principle

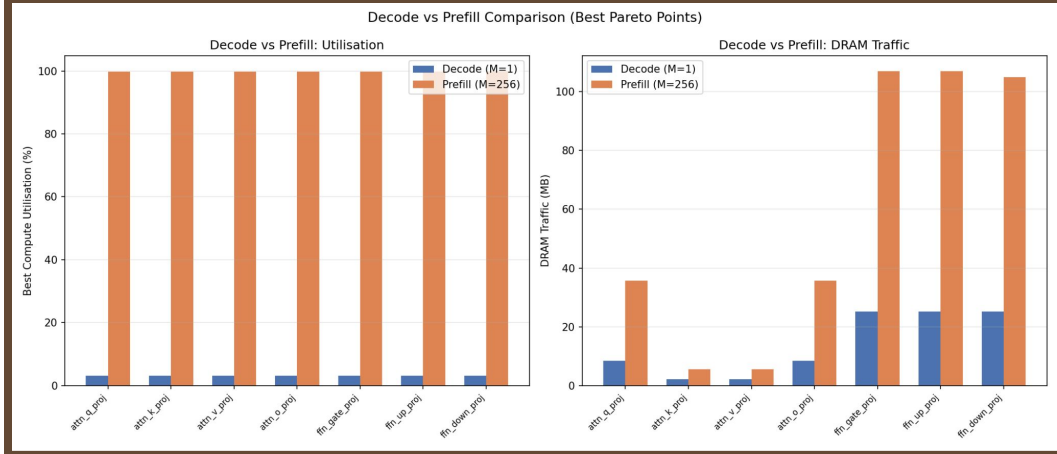
The model estimates memory traffic, latency, and compute utilization across different buffer sizes, tile granularities, and prefetch strategies.

Outcome

We selected a small set of high-value configurations for RTL implementation and cycle-level simulation.

Analytical Performance Model





The optimal parameters are GEMM-specific tiling configurations selected per layer from the Pareto frontier, optimised primarily for the **prefill** stage.

Analytical Performance Model

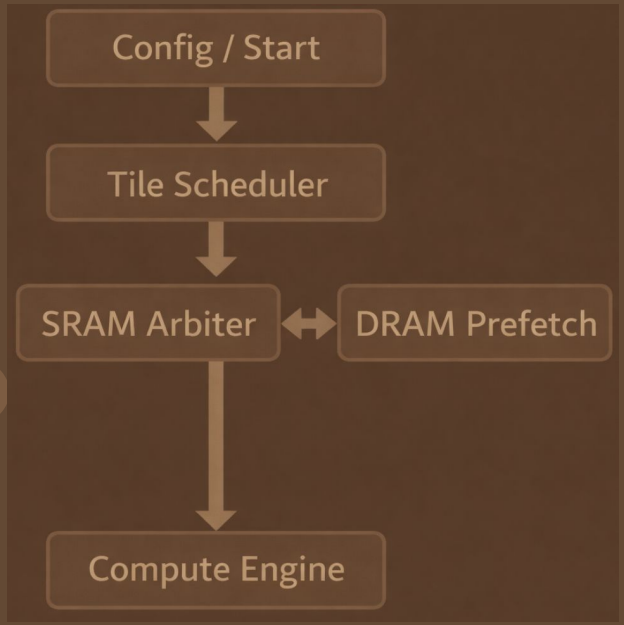
Metric	Baseline	Optimized	Improvement
DRAM traffic	28.5 GB	14.4 GB	**−49.3%**
Compute utilization	61.7%	99.8%	**+38.1 pp**
Per-layer latency	3.65 ms	2.26 ms	**1.62× faster**
Full model latency	131.6 ms	81.4 ms	**1.62× faster**

By applying per-GEMM, prefill-focused tiling optimisation, we reduce DRAM traffic by **49.3%**, increase compute utilisation to near-peak **(99.8%)**, and achieve a **1.62×** end-to-end latency reduction.

Architecture

We target a representative edge SoC configuration with **2 MB** on-chip SRAM (4 × 512 KB banks) and **50 GB/s LPDDR5** DRAM, running quantized (INT4/INT8) billion-parameter Transformer models.

RTL Control & Data flow Implementation



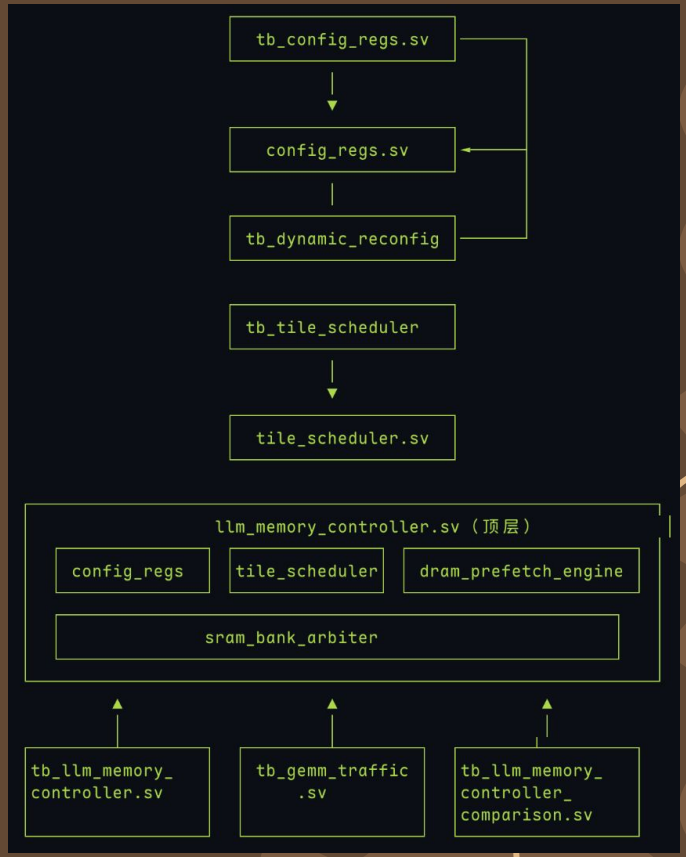
- Implements a centralized hardware control plane for tiled GEMM execution.
- Separates configuration, scheduling, arbitration, and prefetch into cycle-accurate RTL stages.
- Enforces execution correctness via hardware-gated start, busy tracking, and error handling.
- Overlaps computation and memory movement through ping-pong buffering and prefetch control.
- Exposes performance counters to support profiling and optimization of memory-bound execution.

Simulation results

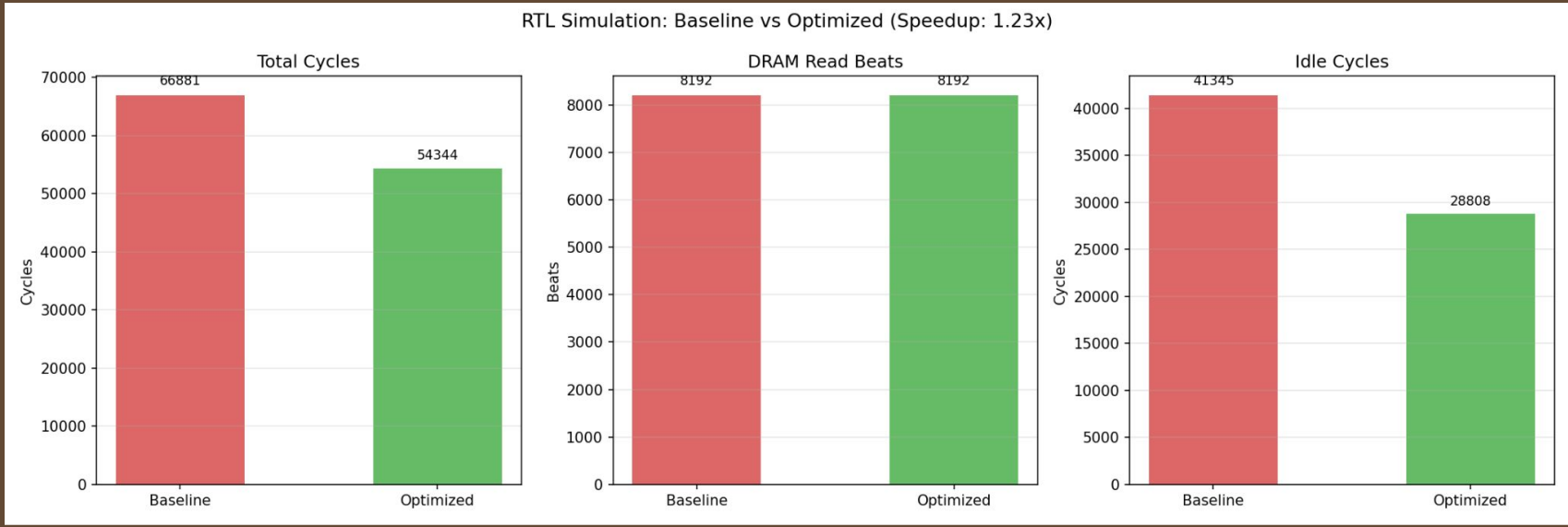
8 Testbench files

50+ Tests pass

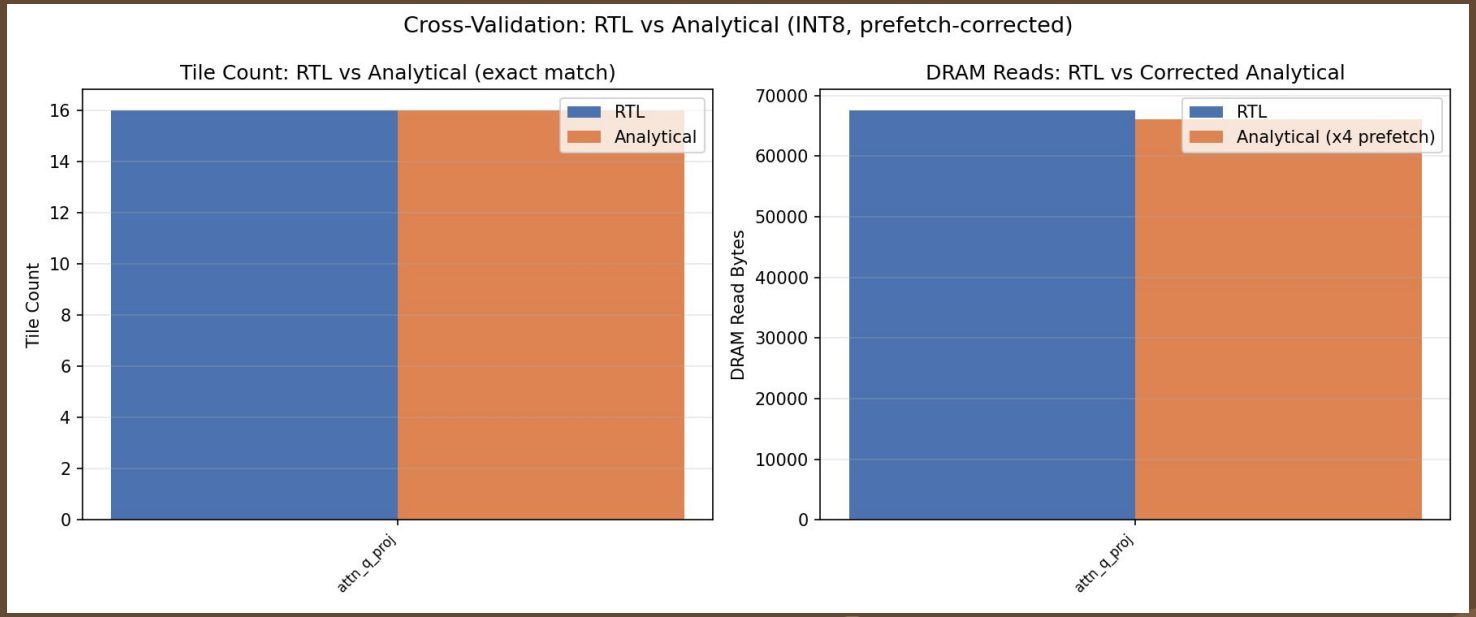
ensuring high coverage
and robust RTL validation.



Simulation results



Simulation results



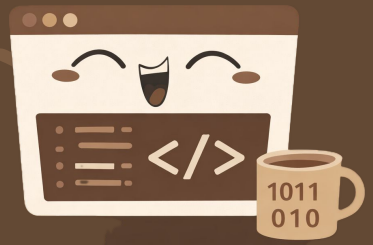
Challenges & lessons learned

Challenges

- Large design space with expensive RTL simulation
- Memory behavior dominates performance and is hard to model accurately
- Coordinating parameters across RTL, testbench, and analytical model

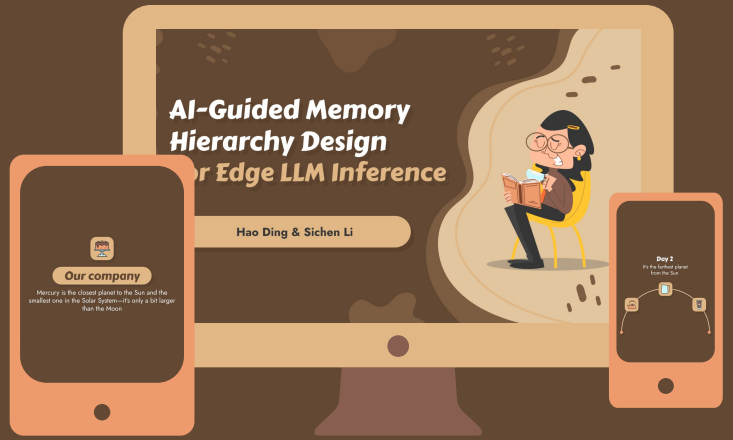
Lessons Learned

- Early analytical modeling is critical for feasibility
- Clear module boundaries simplify iteration and debugging
- Performance counters are essential for understanding bottlenecks



Future work

- Explore adaptive or runtime-tunable memory management policies
- Extend the analytical model to larger LLM layers and multi-layer pipelines
- Run additional RTL simulations across more configurations



Thanks!