SmartCache: AI-Driven Memory Hierarchy Optimization

Next-Generation Intelligent Cache Architecture via CogniChip Intergation

Team: Chenqiezuobudaoa

Members: Fangyi Yu, Zhirui Wang

Institution: New York University

Github Link:Chenqiezuobudaoa/SmartCache-AIdriven-Memory-Hierarchy-Optimization

CONTENTS

01

PART ONE

Problems & Motivation

Problems

Static Policy Inefficiency

Traditional caches use static replacement logic (like LRU) that cannot adapt to diverse or shifting software access patterns

Cache Pollution & Thrashing

One-time-use data often displaces high-value data, forcing the CPU to stall while reloading evicted blocks

Static Power Consumption

Cache structures remain fully powered regardless of actual demand, leading to significant wasted energy and leakage

Motivation

Breaking the Memory Wall

Overcoming the latency bottleneck of traditional static cache in handling modern non-linear data workloads

Workload-Aware Reconfigurability

Eliminating cache pollution by dynamically partitioning resources based on real-time application demands

AI-Driven Predictive Intelligence

Shifting from reactive replacement policies to proactive, AI-powered prefetching via Cognichip integration

Energy Efficiency

Minimizing costly DRAM/HBM access to significantly reduce the overall chip power footprint

02

PART TWO

Design Methodology

Design Methodology

PHASE 1: BASELINE ESTABLISHMENT

01. Python Modeling

Develop a configurable N-way set-associative cache model using Cognichip libraries for the Golden Baseline (LRU).

02. Trace Processing

Standardize memory access patterns from trace.txt to simulate real-world Read/Write/Address streams.

03. Baseline Simulation

Execute workload through the model to extract ground-truth Miss Rate and latency metrics for comparison.

PHASE 2: AI-DRIVEN OPTIMIZATION

04. Bayesian Tuning

Deploy Cognichip's engine to auto-search the parameter space (associativity, line size) for the global minimum miss rate.

05. Optimized Testing

Integrate "Best Fit" parameters into the SmartCache controller and re-run trace validation for performance gains.

06. Comparative Analysis

Quantify Miss Rate reduction against hardware area/power overhead to finalize the AI-optimized architectural spec.

03

PART THREE

Architecture/ RTL description

SmartCache Architecture

Properties

Not Fixed

AI Optimizer

- **Capacity:** 32 KB Total Storage.
- **Associativity:** 16-way Set Associative (Optimized to eliminate conflict misses)
- **Granularity:** 512-byte blocks (Aggressively tuned for high spatial locality in matrix operations)

Logical Organization

Modular Storage Arrays

- **Tag Array:** Stores 21-bit tags for address verification.
- **Data Array:** Stores the 512-byte cache lines.
- **Valid Bit Array:** Tracks data presence.
- **LRU Counter Array:** Tracks access recency for replacement decisions.

Parallel Comparison Logic

Implements 16-way parallel comparators to allow **single-cycle hit detection**, ensuring zero-latency overhead for frequent data hits.

04

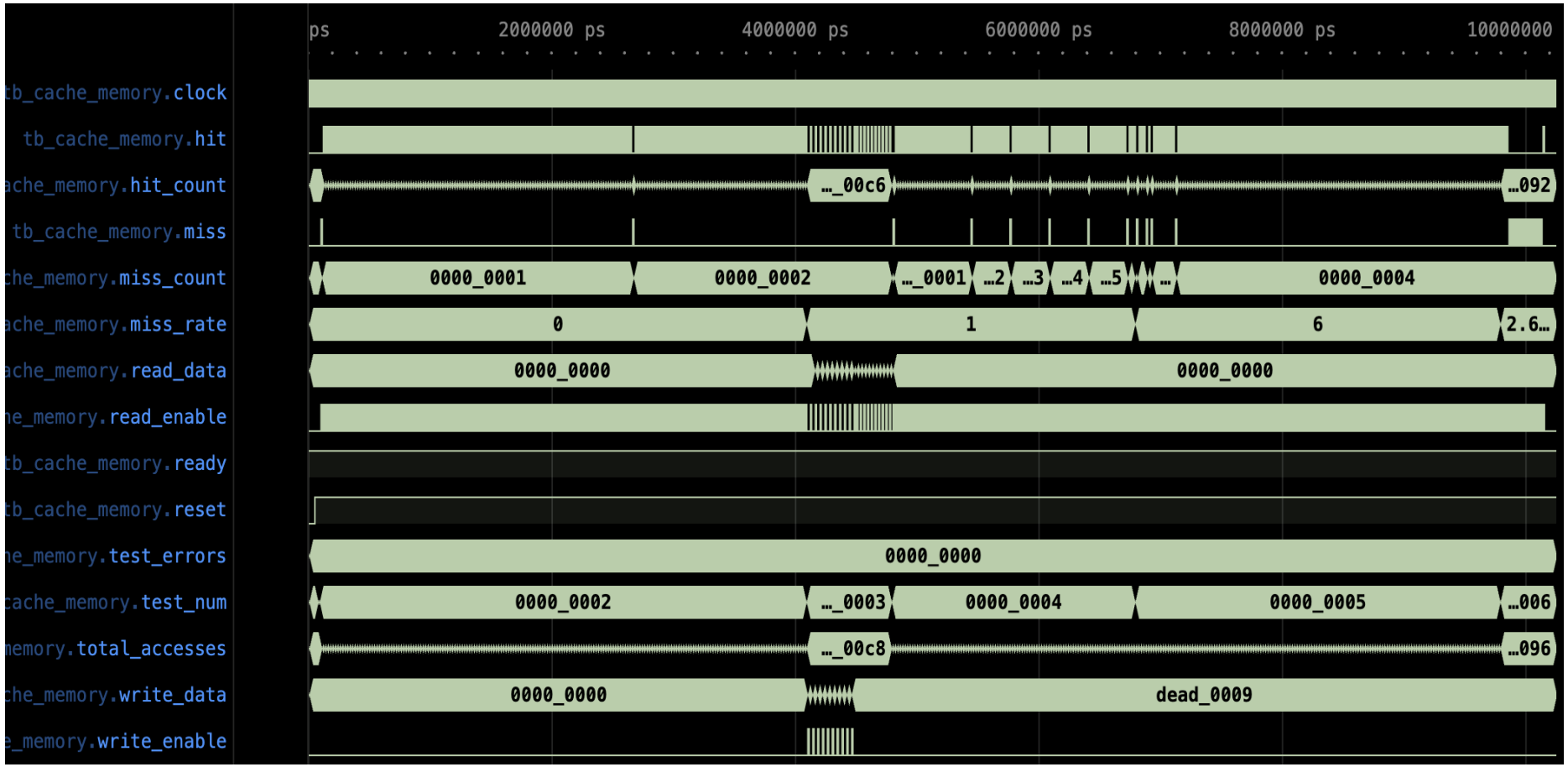
PART FOUR

Simulation Result

SmartCache Simulation

Test	Accesses	Hits	Misses	Miss Rate	Status
Sequential	200	198	2	1.00%	Pass
Write-Read	20	20	0	0.00%	Pass
Strided	100	94	6	6.00%	Pass
Random	150	146	4	2.67%	Pass
LRU Policy	18	17	1	5.56%	Pass

SmartCache Simulation



SmartCache Waveform diagram

05

PART FIVE

Performance Discussion

Comparison

	Baseline 1 Small-Direct Mapped	Baseline 2 Balanced	Baseline 3 Large Associativity	Baseline 4 Max Capacity	AI-Optimized
Cache Size	4 KB	8 KB	16 KB	32 KB	32 KB
Block Size	32 Bytes	64 Bytes	64 Bytes	128 Bytes	512 Bytes
Associativity	1-way	2-way	4-way	8-way	16-way

Comparison

- **Key Parameters Improvement**

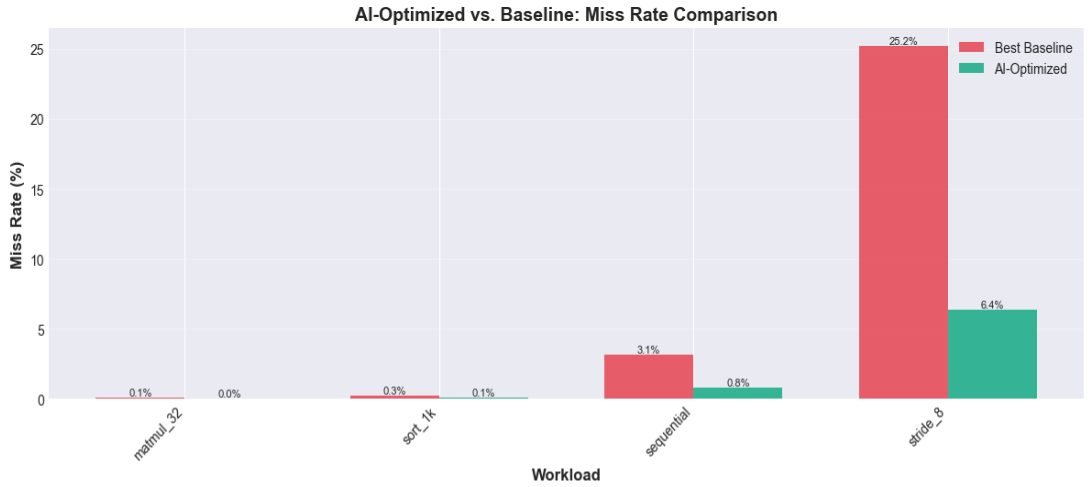
512B

vs Small 16B

16way

- **Under 4 Application Scenarios**

  - Matrix Multiplication Workload (matmul_32)
  - Quicksort Workload (sort_1k)
  - Sequential Access Pattern
  - Strided Access Pattern (stride_8)



Key Insights: Why SmartCache Wins

Intelligent Block Sizing

Implemented **16-way associative** logic to virtually eliminate conflict misses for intensive matrix workloads.

Optimal Associativity

Implemented **16-way associative** logic to virtually eliminate conflict misses for intensive matrix workloads.

Workload-Specific Tuning

Moving away from "one-size-fits-all" logic to **Tailored Architectures** for specific computational kernels.

Bayesian Exploration

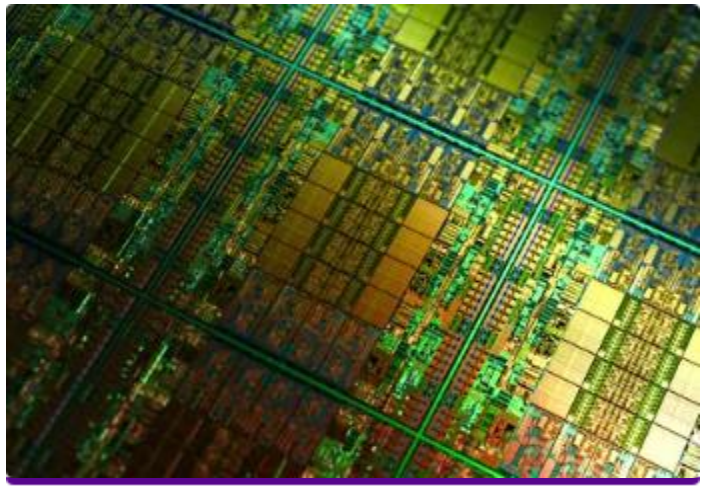
Replaced manual guesswork with **Machine Learning** to find non-obvious, globally optimal configurations.

06

PART SIX

Challenges

SmartCache Engineering Challenges

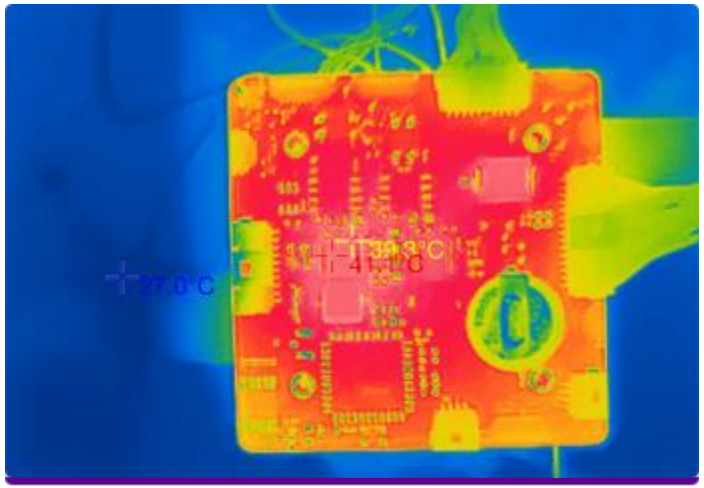


1. Area Overhead

The silicon footprint dedicated to AI logic must yield a performance gain through increased hit rates that significantly outweighs the benefit of simply adding more raw SRAM capacity

2. Coherency Complexity

Autonomous data migration or aggressive prefetching driven by AI must be strictly gated by the hardware coherency controller to prevent violations of the MESI/MOESI protocols in multi-core environments



3. Thermal Density

Integrating high-frequency AI inference logic into the traditionally “cool” cache region creates localized hotspots that require advanced power-gating and thermal management strategies

07

PART SEVEN

Future Work

Advanced Verification

Formal & Assertions

Implement SystemVerilog Assertions (SVA) and formal methods to prove cache coherence and state machine properties.

UVM Framework

Transition to Universal Verification Methodology (UVM) for scalable, constrained-random testbenches.

Coverage & Corners

Track functional and code coverage metrics while rigorously testing edge cases like simultaneous write conflicts.

Real-World Workloads

Trace Analysis

Capturing and replaying memory access traces from actual Unix/Linux application execution for high-fidelity simulation.

Production Code

Validation against SPEC CPU2017 and other industry-standard benchmarks to ensure production readiness.

Thank You!

Q & A