ETH zürich

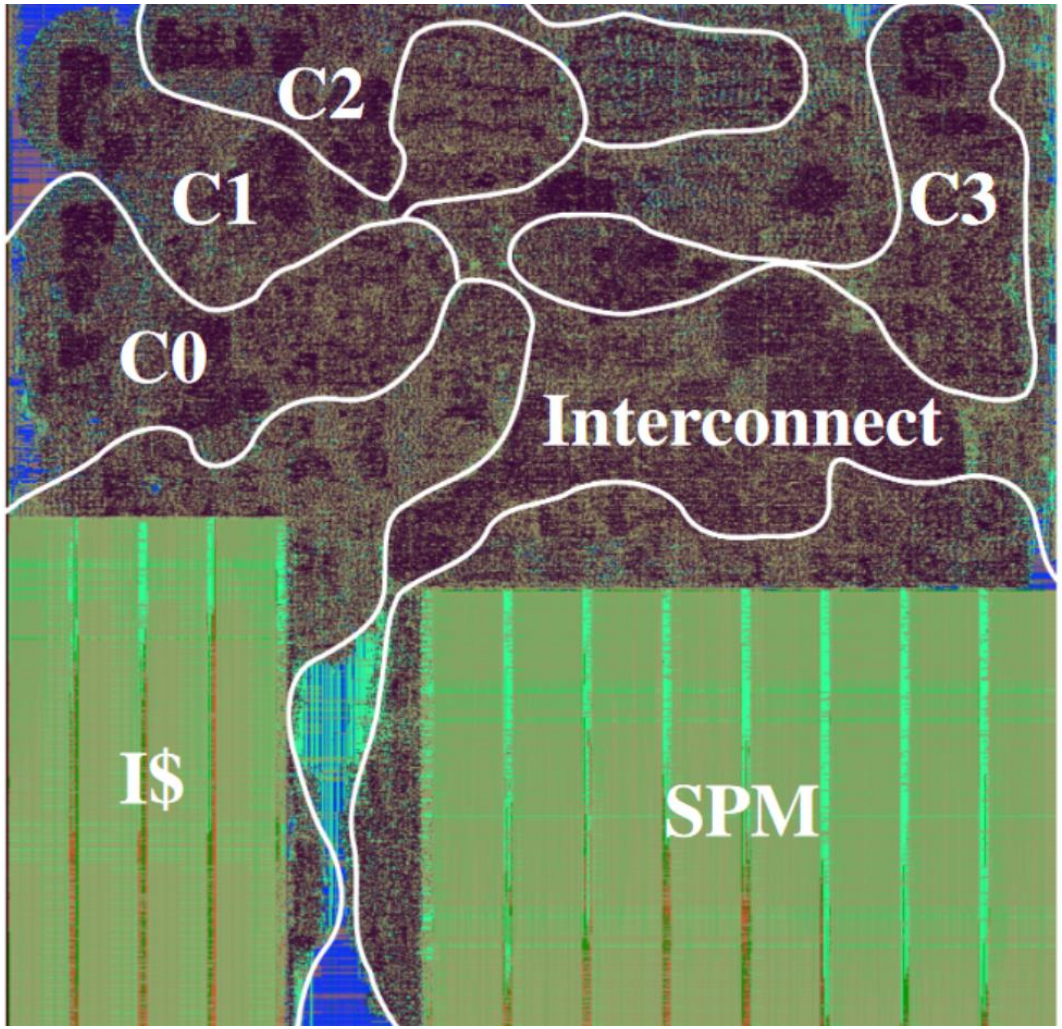
MemPool

Samuel Riedel
Matheus Cavalcante

Overview

1. Instruction Cache Optimization
2. Simulation
3. Student Projects
    1. FPGA port
    2. DSP extension in Snitch
4. Outlook

Baseline instruction cache



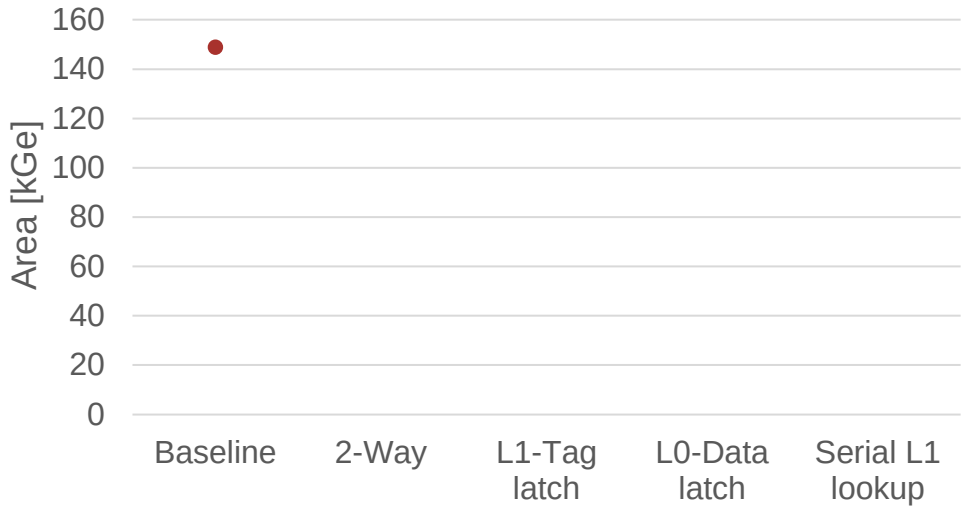
- Inefficient cache macros
  - 2KiB of L1 cache use half as much space as 16KiB of SPM
- L0 cache can not fit the hot-loop of matmul
- Cache uses 37% of the tile's power for matmul

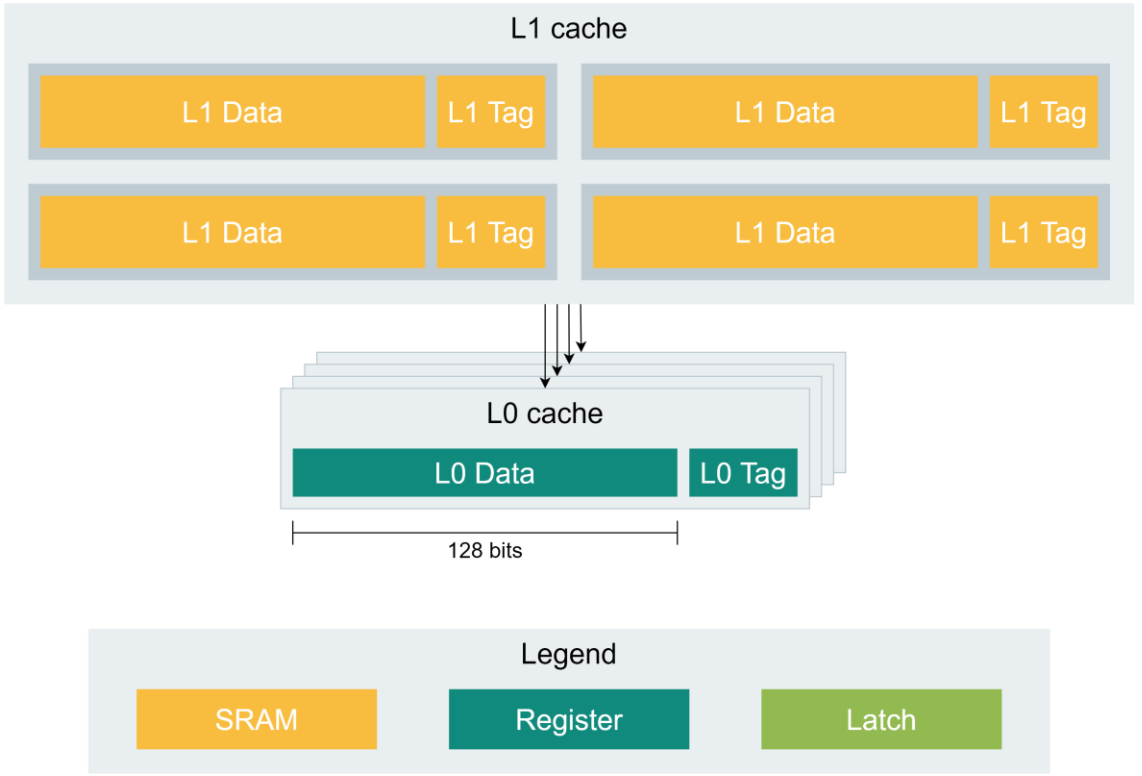
→ Increase the L0 size to fit the hot-loop

→ Change the cache's architecture to reduce it's footprint

Baseline architecture

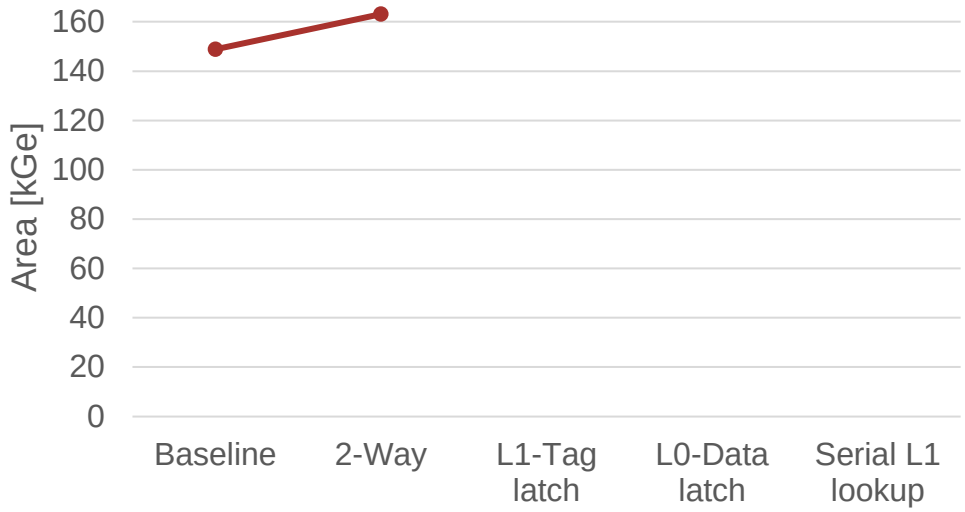
- L1: 4-way set associative
- Cache line width: 128 bits
- Private L0 cache per core
  - Four lines

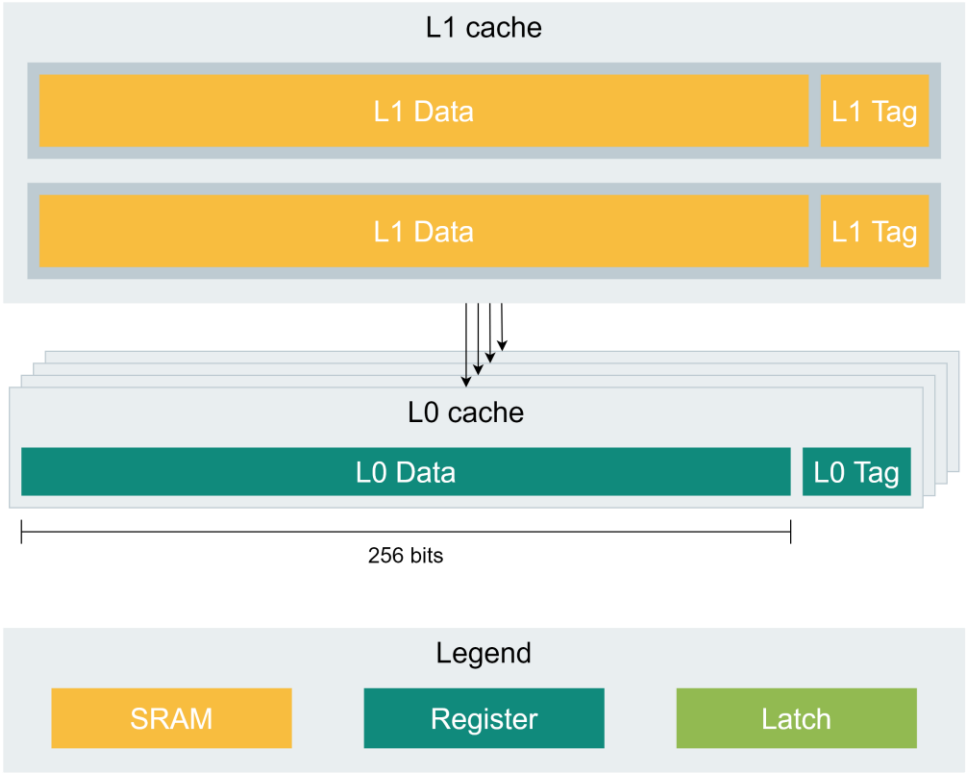




Double the L0 size

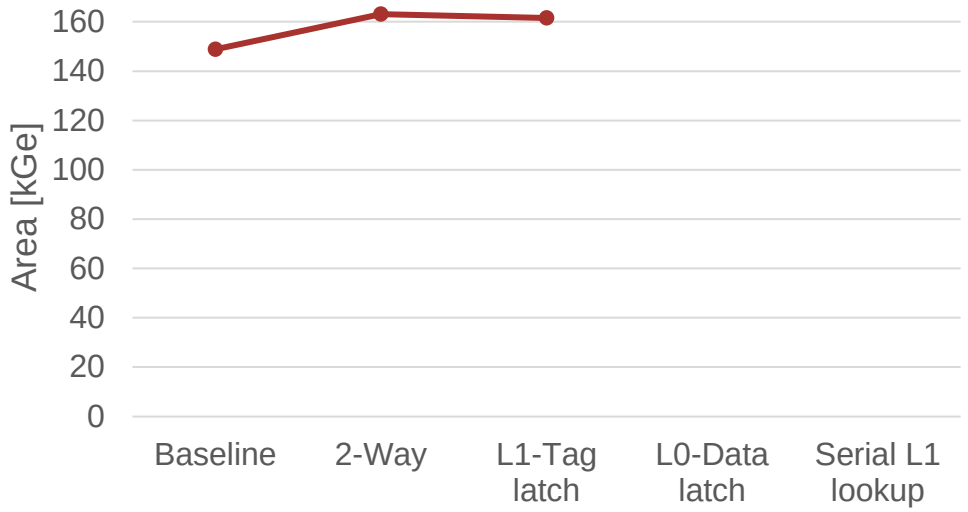
- Double the line-width → Double the L0 size
- L1: **2-way** set associative
  - Keep the same banks
  - Reduce the number of fetches

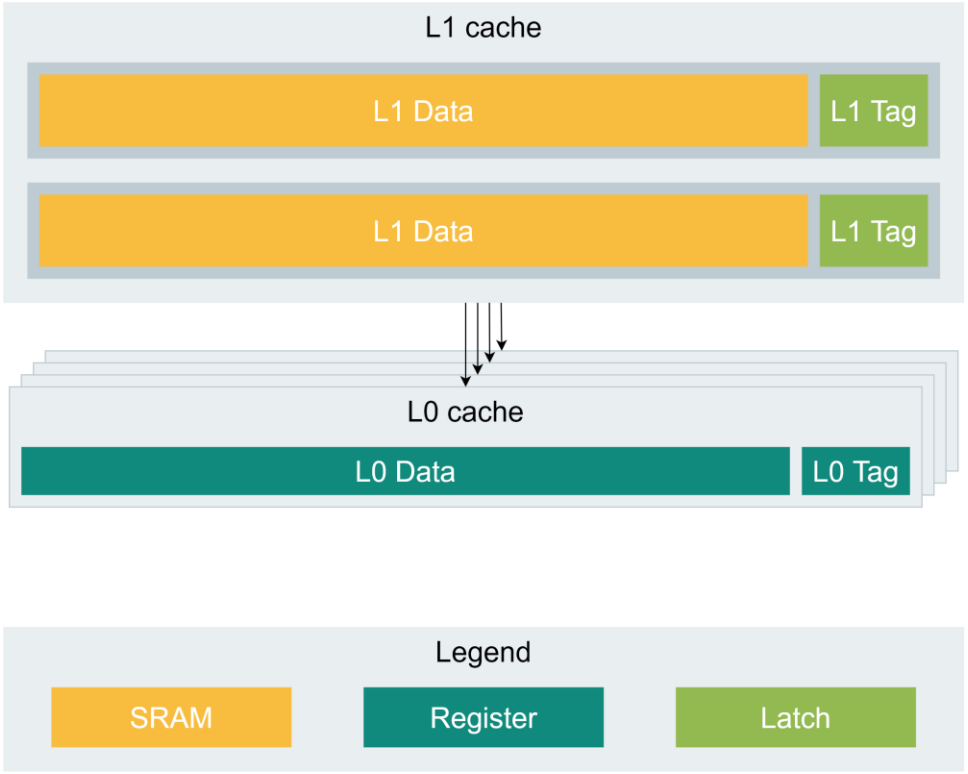




Make the L1's tag latch-based

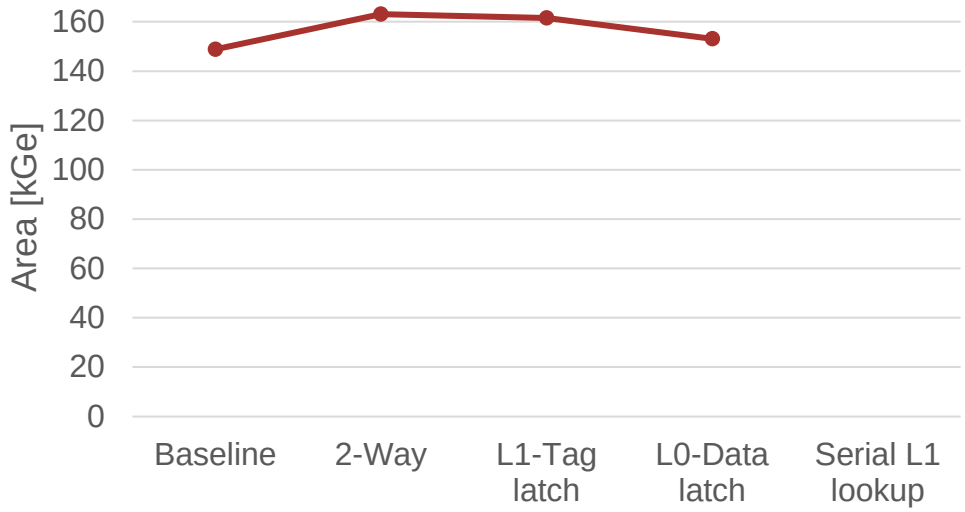
- Remove the small L1 tag SRAM banks
- Latch-based SCM is more area efficient for this size

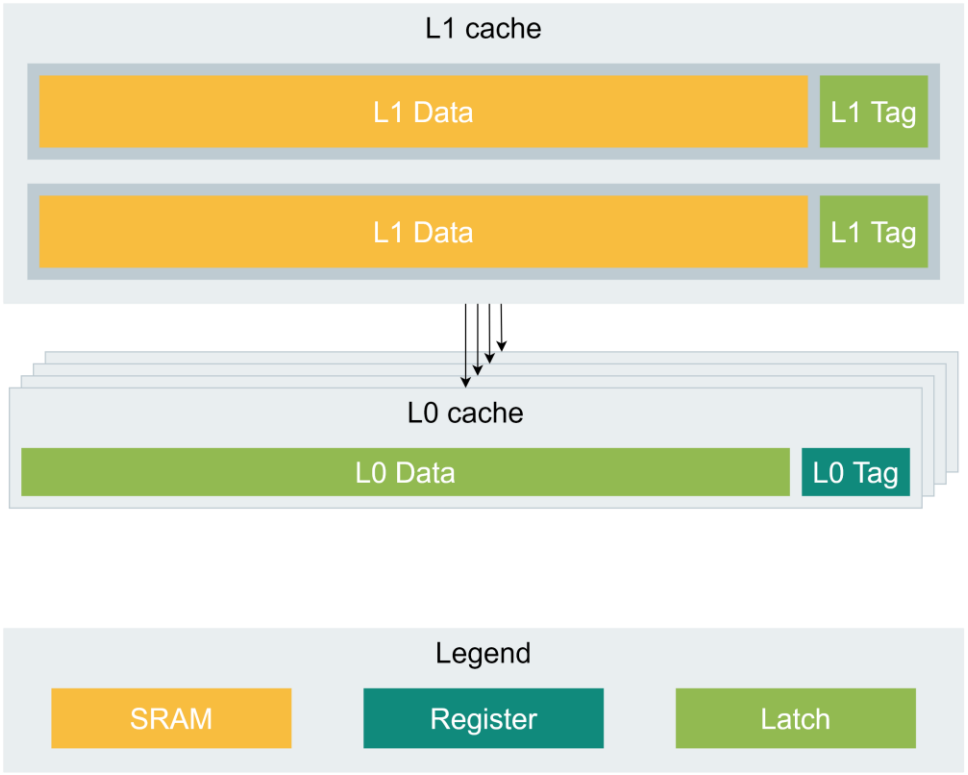




Make the L0's tag latch-based

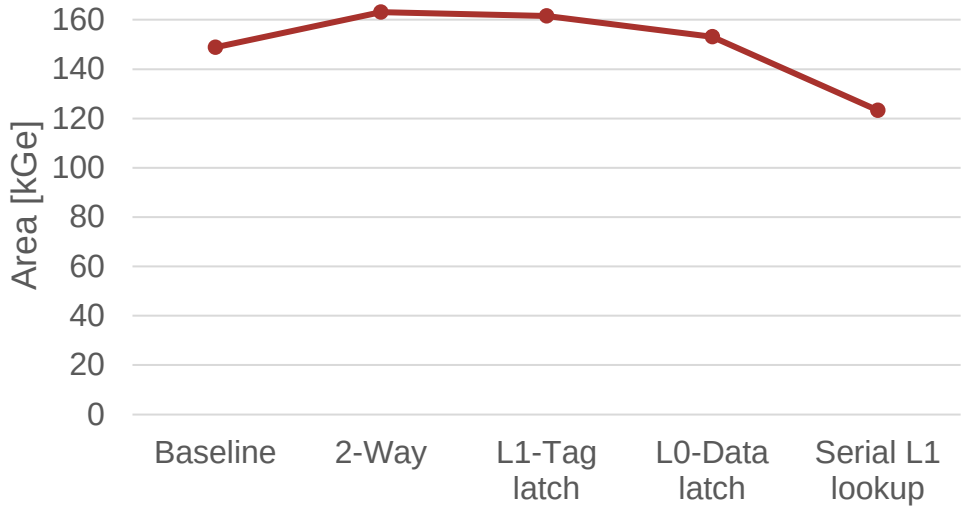
- Replace the L0's registers with latches

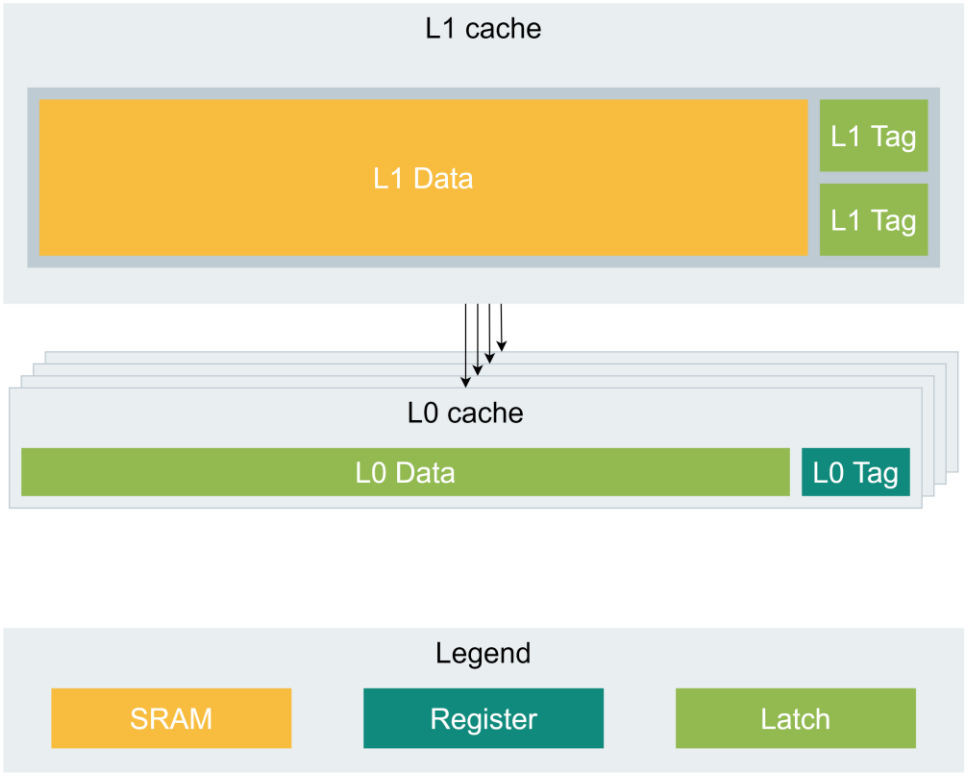




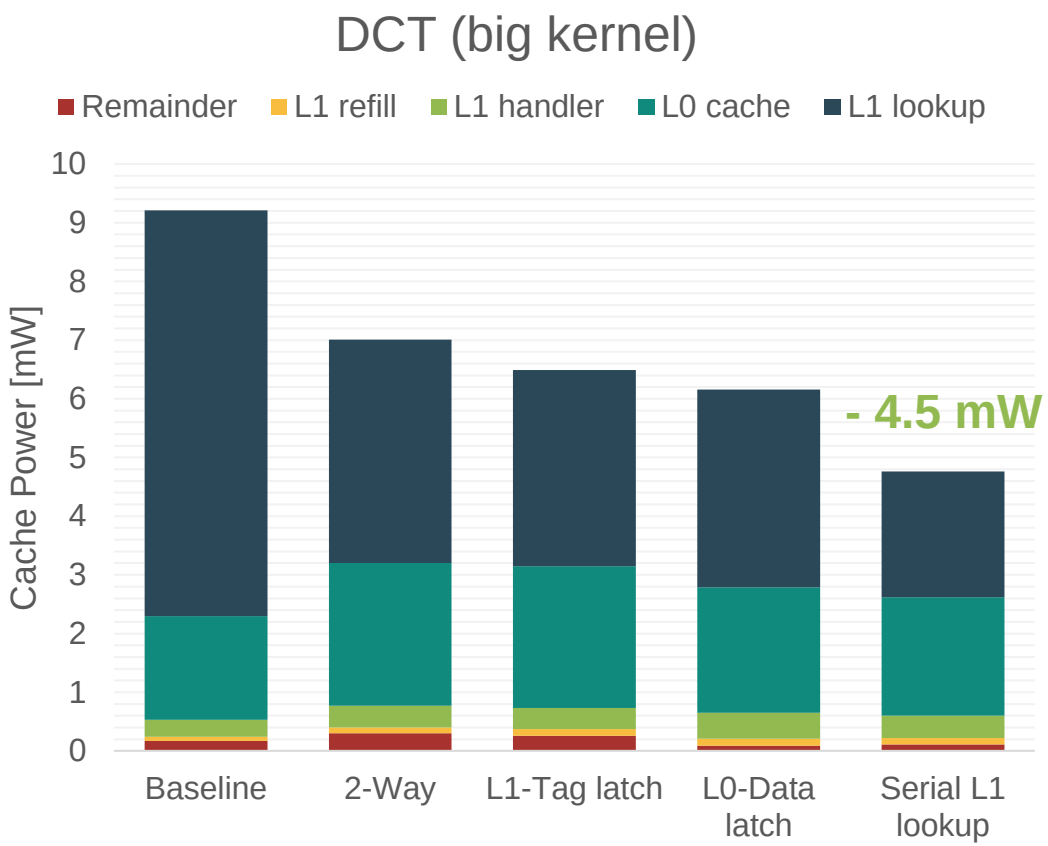
Serialize the L1 lookup

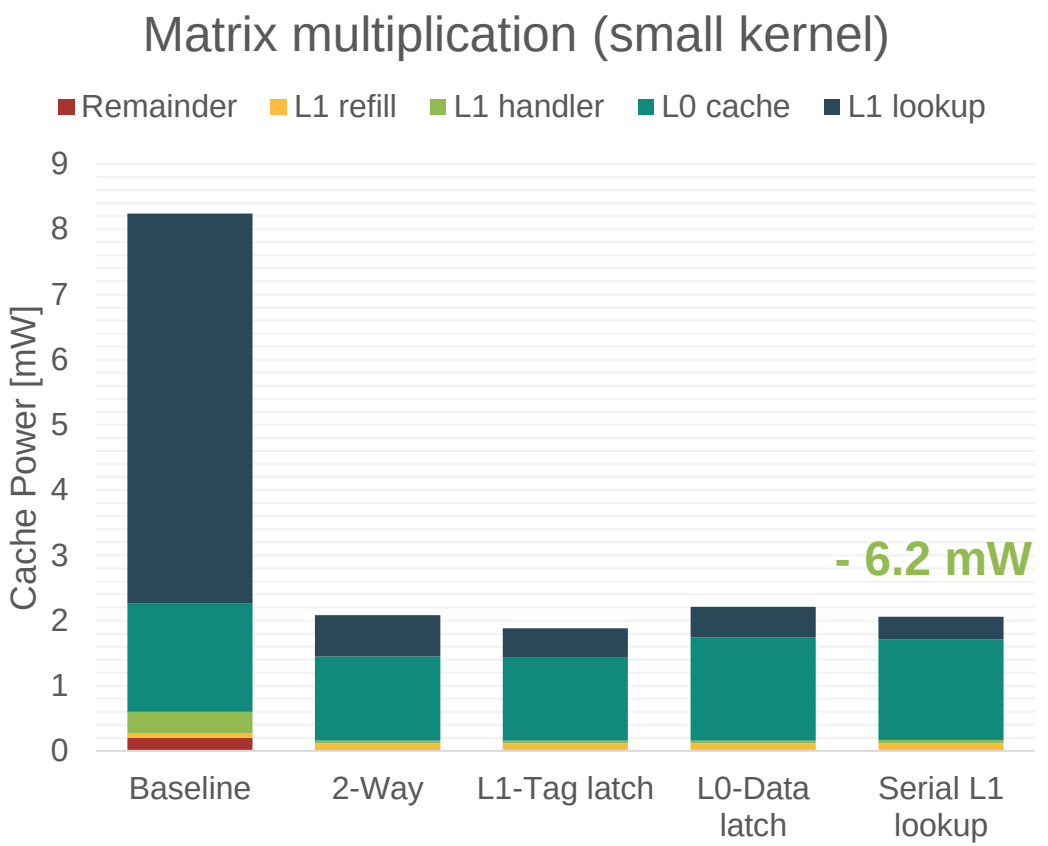
- Merge all L1 data banks
- Doubling the SRAM depth increases the area by only 15%



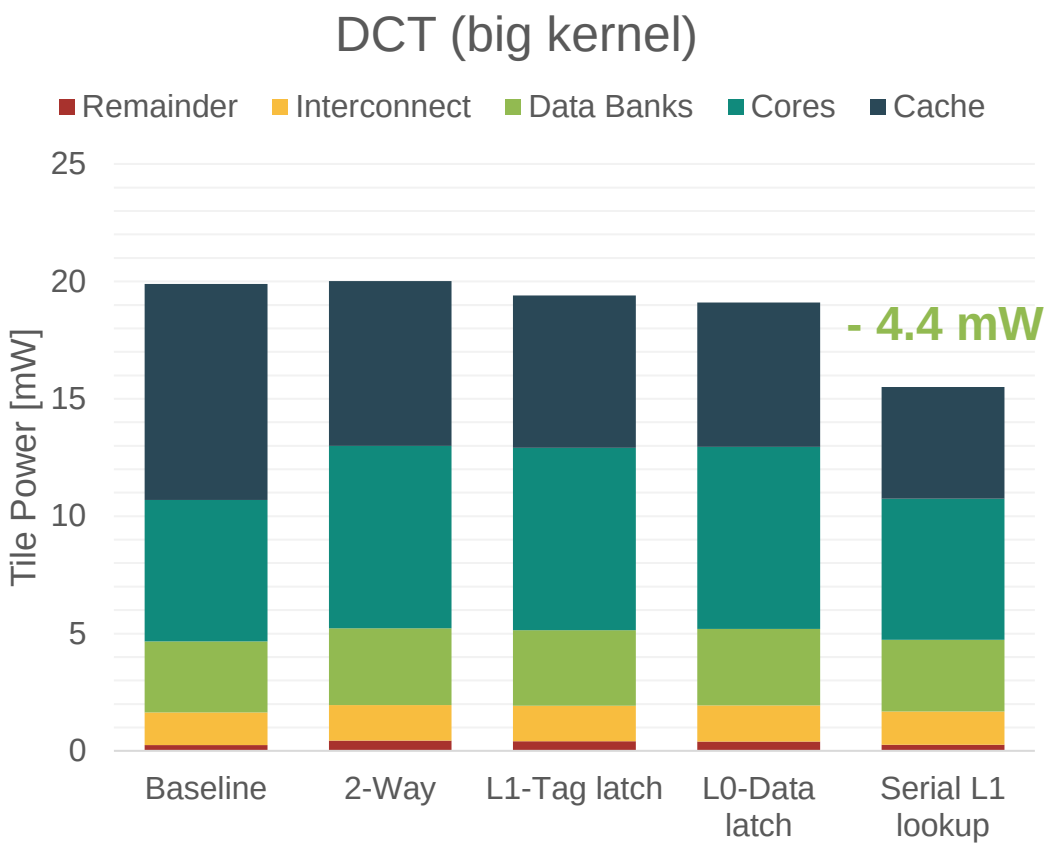


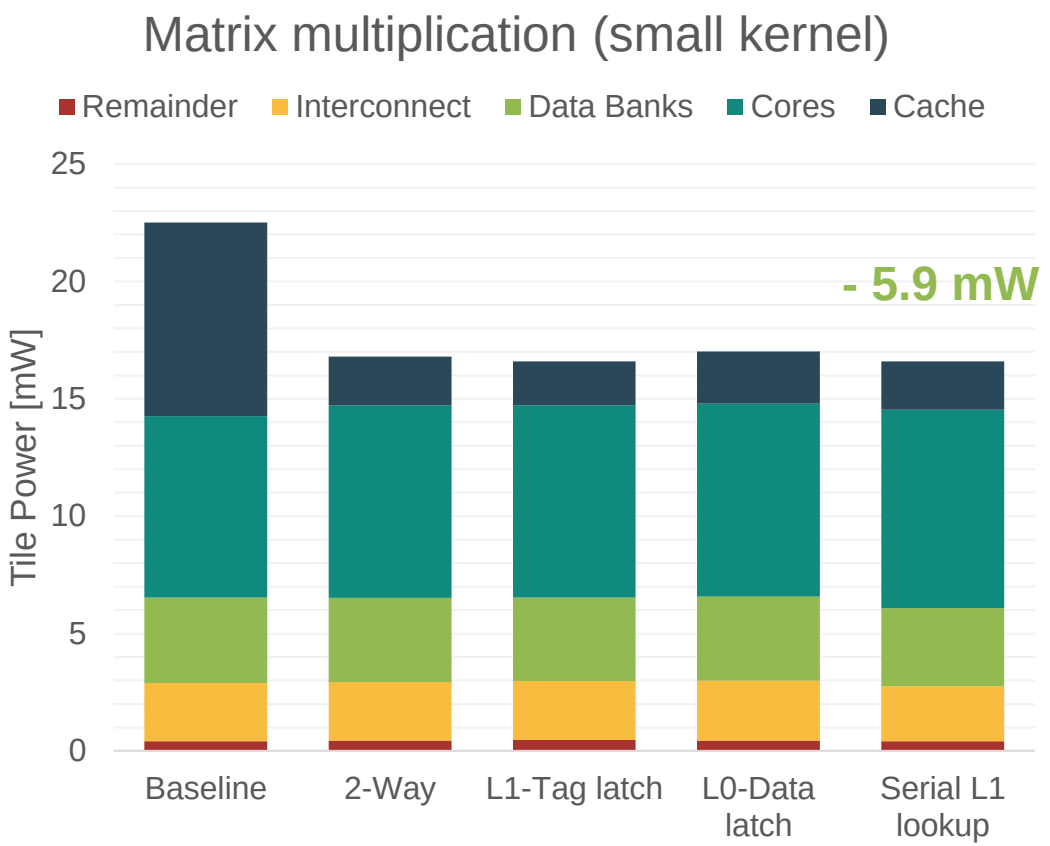
What do we gain in terms of power?





What about the whole tile?





Cache summary

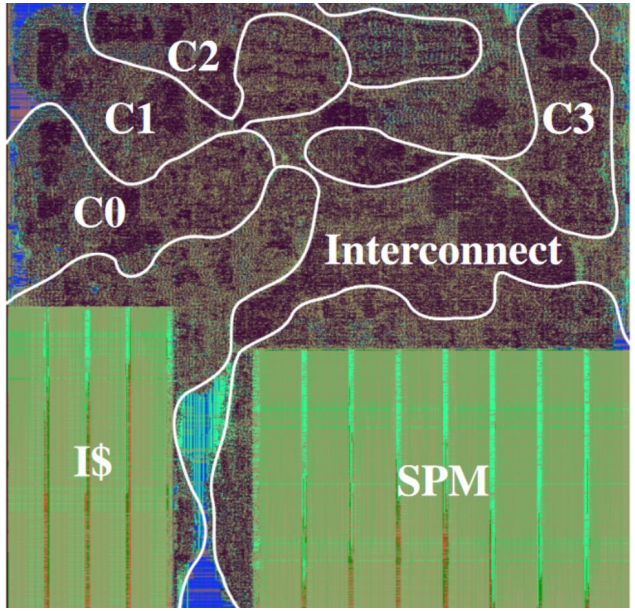
- Double the L0 cache size
- Make the L1 cache 2-way set-associative and serial

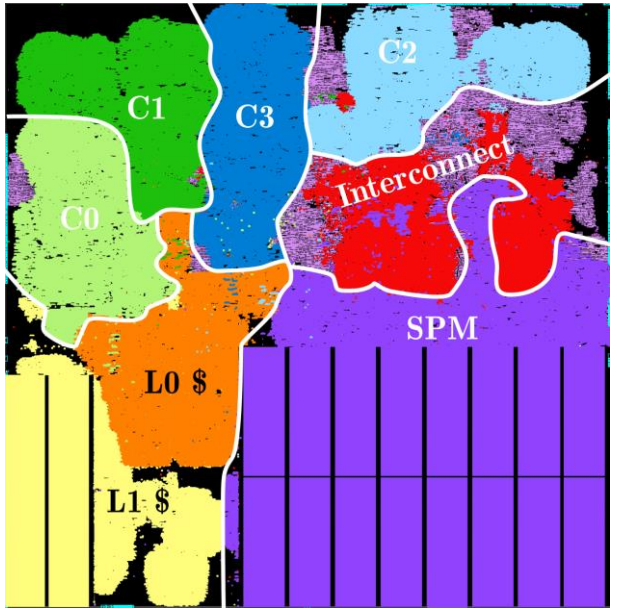
→ Save between 4.4 to 5.9 mW per tile

→ For 64 tiles this is: **282 to 378 mW**

→ The tile became 24 kGE or 4% smaller

→ Benchmarks became slightly faster (~ 2%)





Simulating MemPool

- Until recently, we only used commercial RTL simulator
  - \+ Cycle accurate
  - \+ Fast compilation
  - \+ Full visibility
  - – Slow simulation
  - → Works well for hardware development
- What about benchmarking and software development?
- Benchmarking:
  - – Accurate time modelling
  - – Only execution trace visibility
  - – Fast simulation
- Software development:
  - – Accurate architecture model
  - – Software traces/debugging capabilities
  - – Fast simulation

How to run software on MemPool?

Benchmarking

- Verilator[1]
  - \+ Cycle accurate
  - \+ Fast simulation
  - – Slow compilation

https://www.veripool.org/wiki/verilator

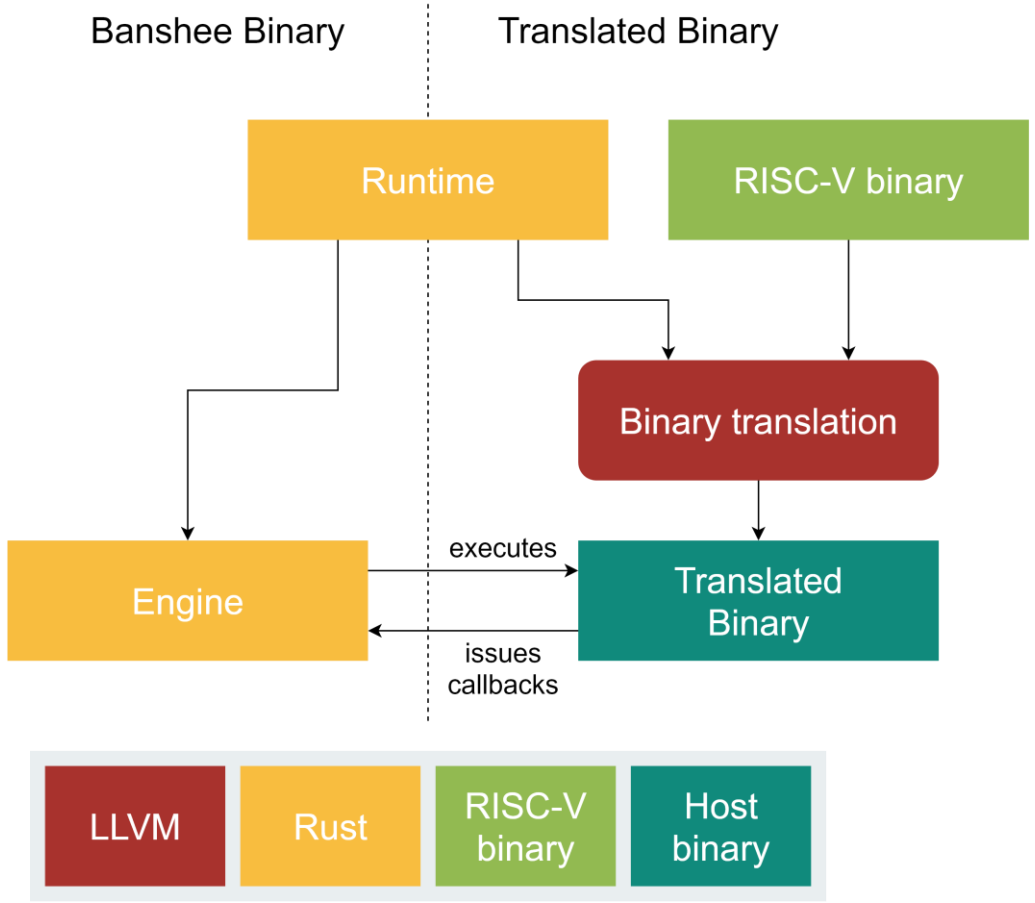
Software development:

- Banshee[2]
  - \+ Extremely fast simulation
  - \+ Fast compilation
  - \+ Tracing and debugging capabilities
  - \+ Full architectural model
  - – Limited time model

[1] https://www.veripool.org/wiki/verilator
[2] https://github.com/pulp-platform/snitch

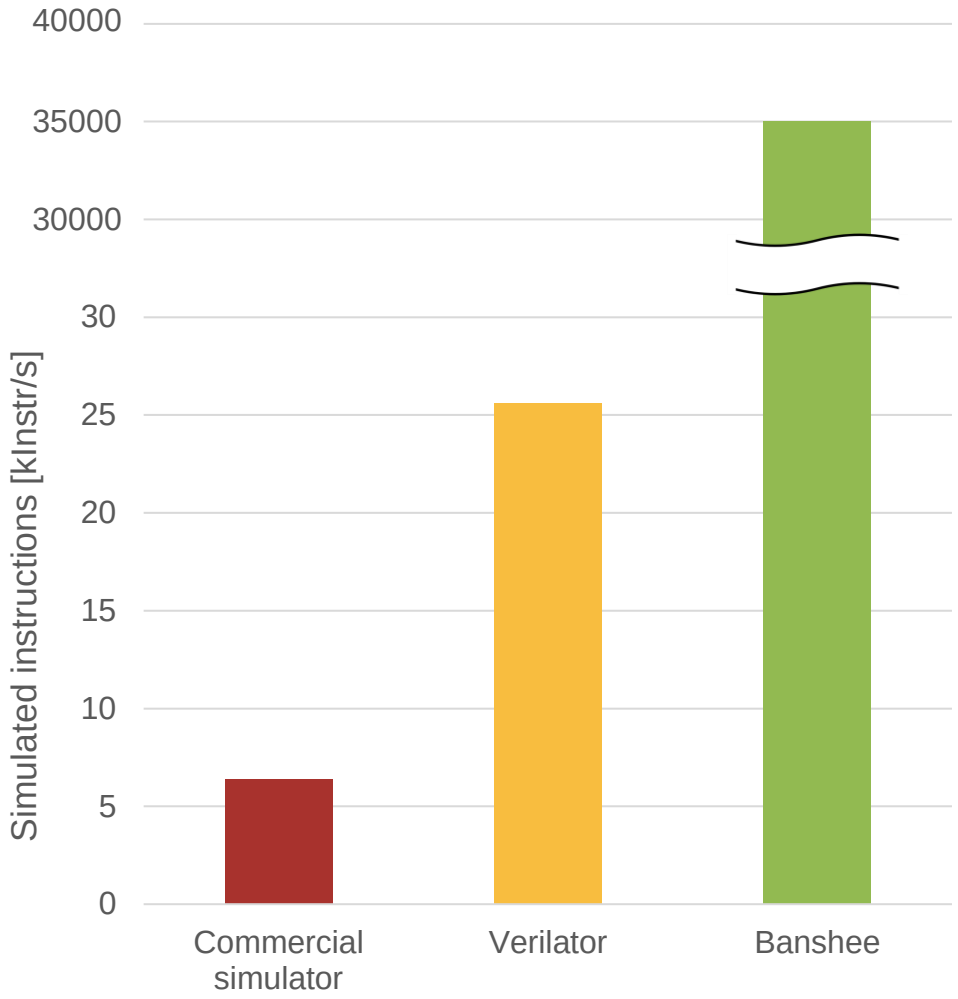
How did we get these tools working?

- Verilator
  - Write C++ testbench (reuse lowRISC's open-source library)
  - Rewrite unsupported SystemVerilog code
  - Fix additional linter warnings
- Banshee
  - Binary translator
  - Each core is an independent thread
  - Engine models architecture specific callbacks
    - Memory map
    - Core status
  - Pipeline stalls are modelled



How do they compare?

- Commercial simulator: Hardware development
  - Slowest
- Verilator: Cycle accurate benchmarking
  - Four times faster
- Banshee: Software development
  - 1000 times faster



Getting MACs to MemPool and MemPool to the FPGA

- Ported MinPool onto a Zynq UltraScale+ MPSoC
  - 16 Snitch cores
  - Basic offload capabilities from host
- Extended Snitch with custom DSP instructions
  - Selection of instructions from the *xpulp* instruction set
  - Fully supported GCC toolchain
  - Implemented instructions in co-processor

Student Project: DSP extension

Generic arithmetical operations

- Set less equal
- Absolute value
- Minimum, maximum
- Sign/zero-extension
- Clip operations
- Immediate branching

Extended load/store addressing modes

- Post-increment load/store
- Register-register load/store

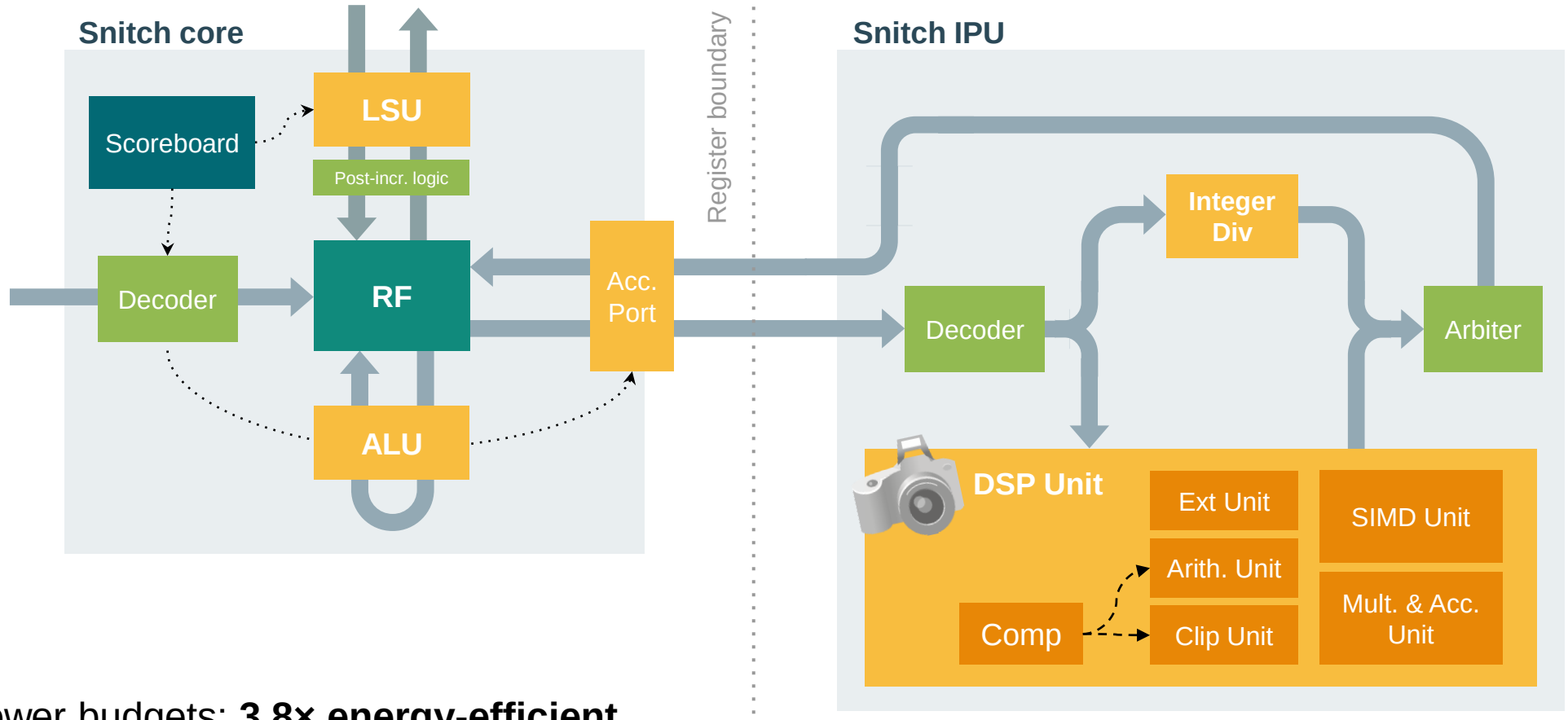
Multiply-accumulate

- Multiply and accumulate
- Multiply and subtract

Packed-SIMD

- 8-bit and 16-bit
- Vectorial and scalar replication mode
- Dot-product
- Addition, subtraction
- Average
- Abs, min, max
- Shift operations
- Logical operations
- Extract, insert, shuffle

Student Project: DSP extension



- Better for tight power budgets: **3.8× energy-efficient**
- Better for strict timing constraints: up to **4.4× speed-up**

Outlook

- Instruction path
  - Hierarchical AXI interconnect
  - Add a cache hierarchy
- Add DMAs
- Do a detailed evaluation with more benchmarks