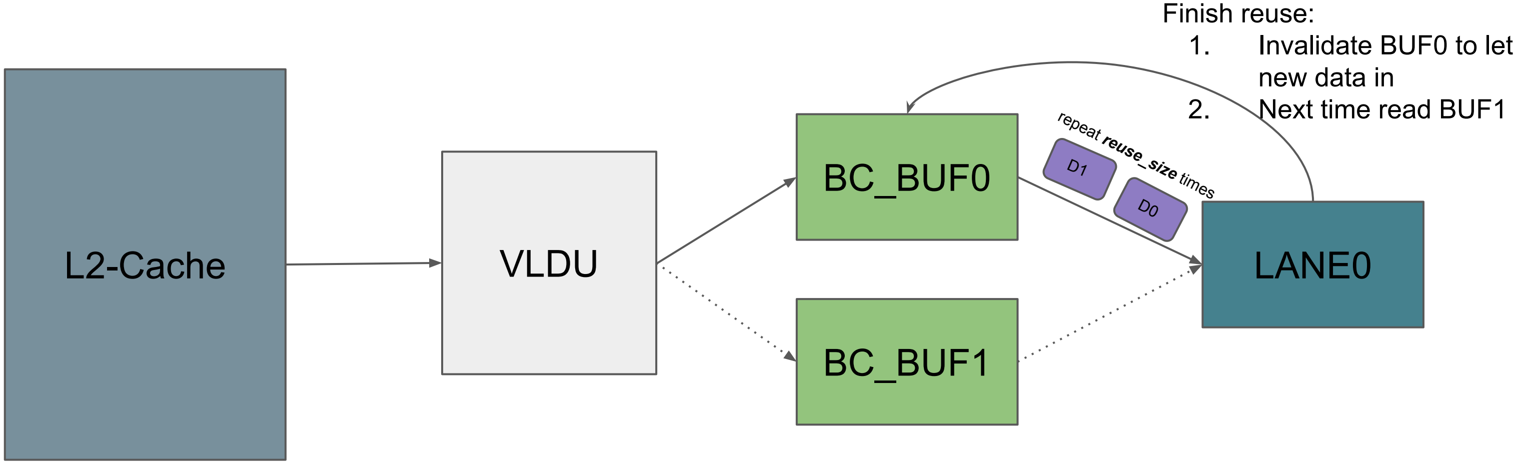
Week 12

- GOAL:
  - Improve utilization of single Ara with different lane configurations
- TODO:
  - Software:
    - Add support for Spike simulator (optional)
    - Add custom instructions to LLVM compiler
  - Hardware:
    - Architecture modification proposal
    - Design sub-modules
    - Test riscv-tests and other benchmarks
    - Test new MatMul performance

Architecture Modification

- Broadcast buffer
  - b_vec reuse: #lanes * reuse_size
  - Load b_vec from L2-cache and store it in bc_buffer for reuse
  - Ping-Pang style
    - While using data in buf0, VLDU can still loads data to BUF1

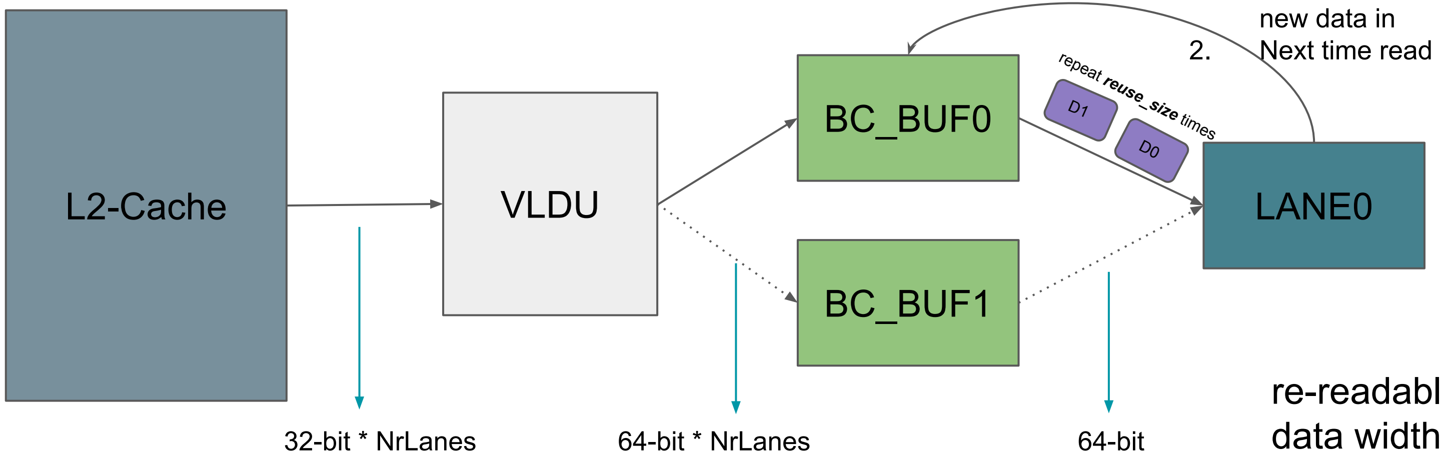


Finish reuse:

1. Invalidate BUF0 to let new data in
2. Next time read BUF1

Architecture Modification

- Broadcast buffer
  - b_vec reuse: #lanes * reuse_size
  - Load b_vec from L2-cache and store it in bc_buffer for reuse
  - Ping-Pang style
    - While using data in buf0, VLDU can still loads data to BUF1



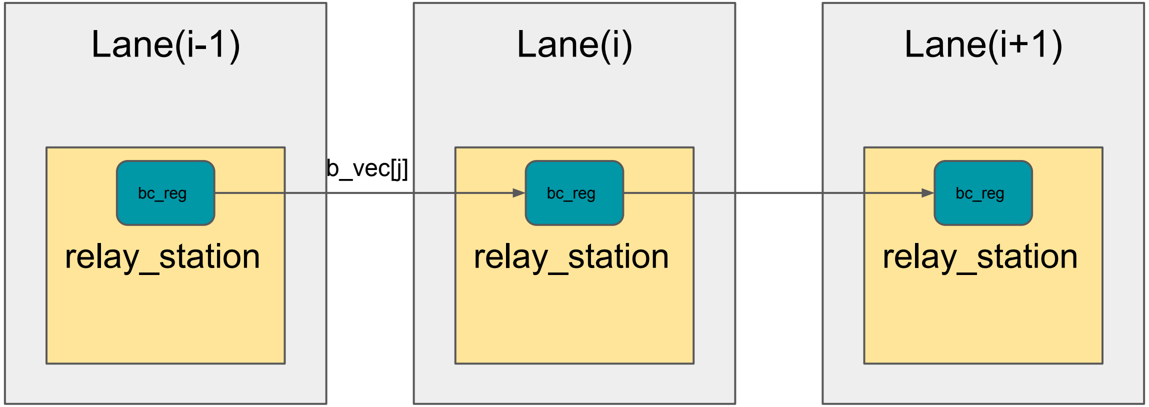
Finish reuse:

1. Invalidate BUF0 to let new data in
2. Next time read BUF1

re-readable FIFO with data width conversion

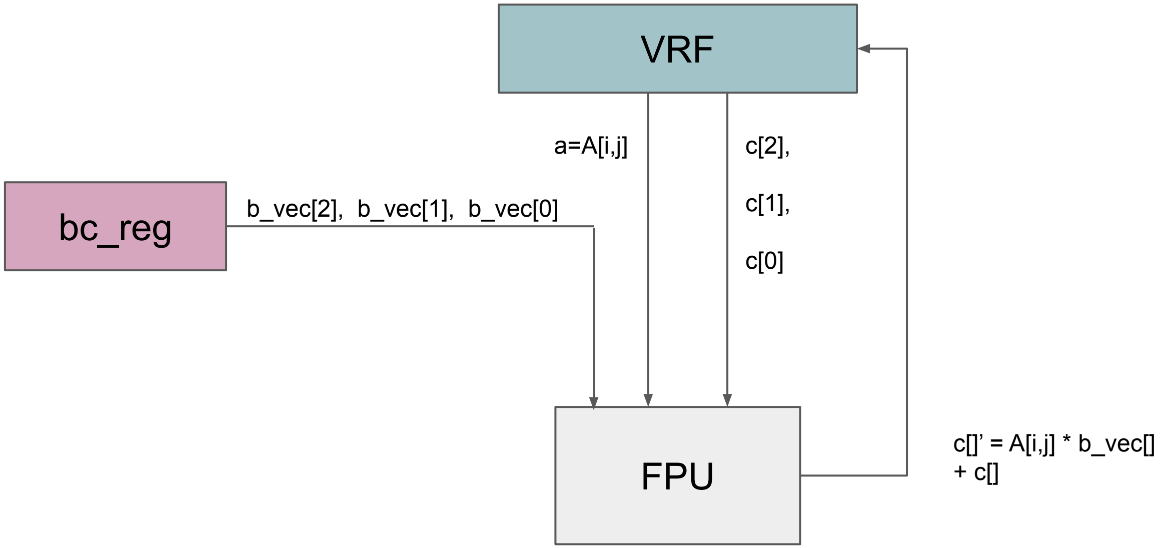
Architecture Modification

- Relay-Station
  - Receive the bc data from the previous lane (except the first lane)
  - Forward it to the next lane (except the last lane)



Architecture Modification

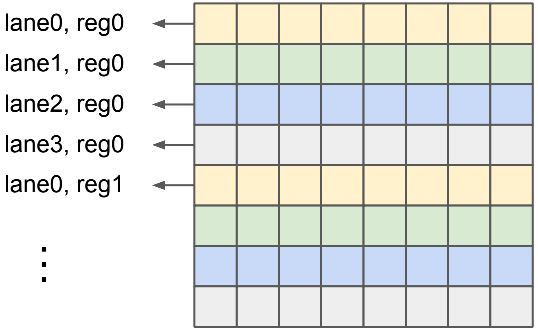
- VMFPU operands:
    - a: elements of matrix A, stride loaded, vl = reuse_size
        - one element per b_vec, need a counter (BLEN)
    - b: vector of matrix B, broadcast, vl = BLEN
    - c: accumulated result, normal vector register
        - one element per b element



Architecture Modification

- Store results
  - In order:
    - reg0: lane0, lane1, …
    - reg1: lane0, lane1, …

Result Matrix C

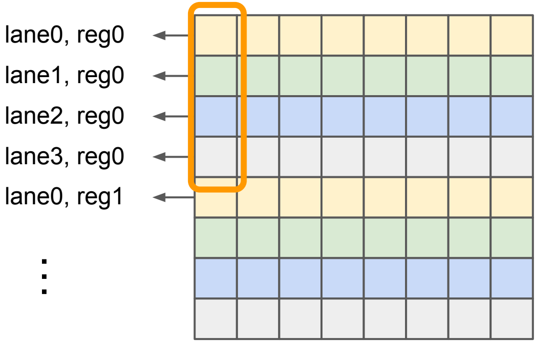


4 lanes,
reuse_size=2

Architecture Modification

- Store results
  - In order:
    - reg0: lane0, lane1, …
    - reg1: lane0, lane1, …
  - Solution: stride store
    - Poor performance
    - Multi-banking L2-Cache, enable simultaneous write & read

Result Matrix C



4 lanes,
reuse_size=2