

PULP PLATFORM

Open Source Hardware, the way it should be!

# *Working with RISC-V*

## Part 2 of 5 : Advanced RISC-V Architectures

Luca Benini

<luca.benini@unibo.it>

Davide Rossi

<davide.rossi@unibo.it>

**ETH** zürich



<http://pulp-platform.org>



[@pulp\\_platform](https://twitter.com/pulp_platform)



[https://www.youtube.com/pulp\\_platform](https://www.youtube.com/pulp_platform)



# Summary

- Part 1 – Introduction to RISC-V ISA
- **Part 2 – Advanced RISC-V Architectures**
  - Going 64 bit
  - Bottlenecks
  - Safety/Security
  - Vector units
- Part 3 – PULP concepts
- Part 4 – PULP Extensions and Accelerators
- Part 5 – PULP based chips



# PULP RISC-V Cores from Tiny to App-level

32 bit			64 bit
<b>Low Cost Core</b> <ul style="list-style-type: none"> <li>▪ <b>Zero-riscy</b> <ul style="list-style-type: none"> <li>▪ RV32-ICM</li> </ul> </li> <li>▪ <b>Micro-riscy</b> <ul style="list-style-type: none"> <li>▪ RV32-CE</li> </ul> </li> </ul> <p><i>ARM Cortex-M0+</i></p>	<b>DSP Enhanced Core</b> <ul style="list-style-type: none"> <li>▪ <b>RI5CY</b> <ul style="list-style-type: none"> <li>▪ RV32-ICMFX                             <ul style="list-style-type: none"> <li>▪ SIMD</li> <li>▪ HW loops</li> <li>▪ Bit manipulation</li> <li>▪ Fixed point</li> </ul> </li> </ul> </li> </ul> <p><i>ARM Cortex-M4</i></p>	<b>Streaming Compute Core</b> <ul style="list-style-type: none"> <li>▪ <b>Snitch</b> <ul style="list-style-type: none"> <li>▪ RV32-ICMDFX</li> </ul> </li> </ul>	<b>Linux capable Core</b> <ul style="list-style-type: none"> <li>▪ <b>Ariane</b> <ul style="list-style-type: none"> <li>▪ RV64-IC(MA)</li> <li>▪ Full privileged specification</li> </ul> </li> </ul> <p><i>ARM Cortex-A5</i></p>



# From IoT to HPC

- **For the first 4 years of PULP, we used only 32bit cores**
  - Most IoT near-sensor applications work well with 32bit cores.
  - 64bit memory space is not affordable in an MCU-class device
- **But times change:**
  - Large datasets, high-precision numerical calculations (e.g. double precision FP) at the IoT edge (gateways) and cloud
  - Software infrastructure (OS – typically linux) with virtual memory assumes 64bit
  - High-performance computing, being hot again, requires 64bit
  - **Research question – pJ/OP on 64bit data+address space is possible? How?**

# An application class processor

- **Virtual Memory**

- Multi-program environment
- Efficient sharing and protection

- **Operating System**

- Highly sequential code
- Increase frequency to gain performance

- **Large software infrastructure**

- Drivers for hardware (PCIe, ethernet)
- Application SW (e.g.: Tensorflow, ...)



- **Larger address space (64-bit)**

- **Requires more hardware support**

- MMU (TLBs, PTW)
- Privilege Levels
- More Exceptions (page fault, illegal access)

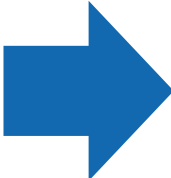
→ **Ariane** an application class processor



**NOT an ARM Cortex-A killer!**

**“Controller” core with must-have features for 64bit OSes**

# ARIANE: Linux Capable 64-bit core

- Application class processor
  - Linux Capable
    - M, S and U privilege modes
    - TLB
    - Tightly integrated D\$ and I\$
    - Hardware PTW
  - Optimized for 1+GHz clock speed
    - **Frequency: 1+ GHz** (22 FDX)
    - **Area: 100s kGE** (200-400)
    - **Critical path: ~ 25-30** logic levels
- 
- 6-stage pipeline
    - In-order issue
    - Out-of-order write-back
    - In-order commit
  - Branch-prediction
    - RAS
    - Branch Target Buffer
    - Branch History Table
  - Scoreboarding
  - Designed for extensibility

F. Zaruba and L. Benini, "The Cost of Application-Class Processing: Energy and Performance Analysis of a Linux-Ready 1.7-GHz 64-Bit RISC-V Core in 22-nm FDSOI Technology," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 27, no. 11, pp. 2629-2640, Nov. 2019



# Absolute minimum necessary to boot Linux?

## ■ Hardware

- 64 or 32 bit Integer Extension
- Atomic Extension
- Privilege levels U, S and M
  - MMU
- FD Extension or out-of-tree Kernel patch
- 16 MB RAM
- Interrupts
  - Core local interrupts (**CLINT**) like timer and inter processor interrupts
- Serial

## ■ Software

- Zero Stage Bootloader
- Device Tree Specification (DTS)
- RAM preparation (zeroing)
- Second stage bootloader
  - BBL
  - Uboot
  - ...
- Linux Kernel
- User-space applications (e.g.: Busybox) or distro



# Ariane $\mu$ ARC

## 1. PC Gen

- Select PC

## 2. Instr. Fetch

- TLB
- Query I\$

## 3. Instr. Decode

- Re-align
- De-compress
- Decode

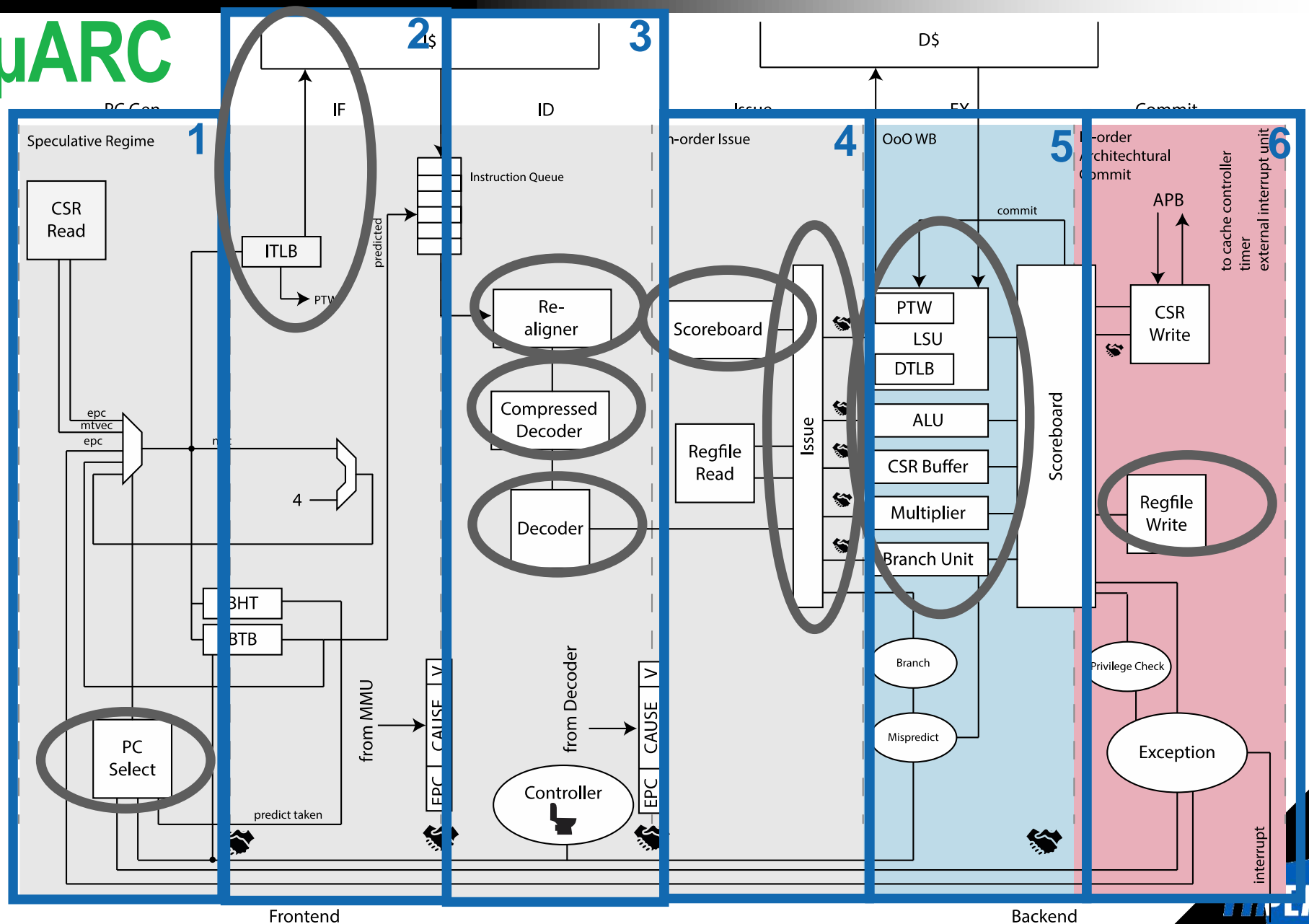
## 4. Issue

- Select FU
- Issue

## 5. Execute

## 6. Commit

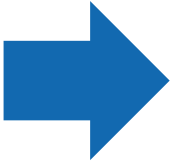
- 1. Write state







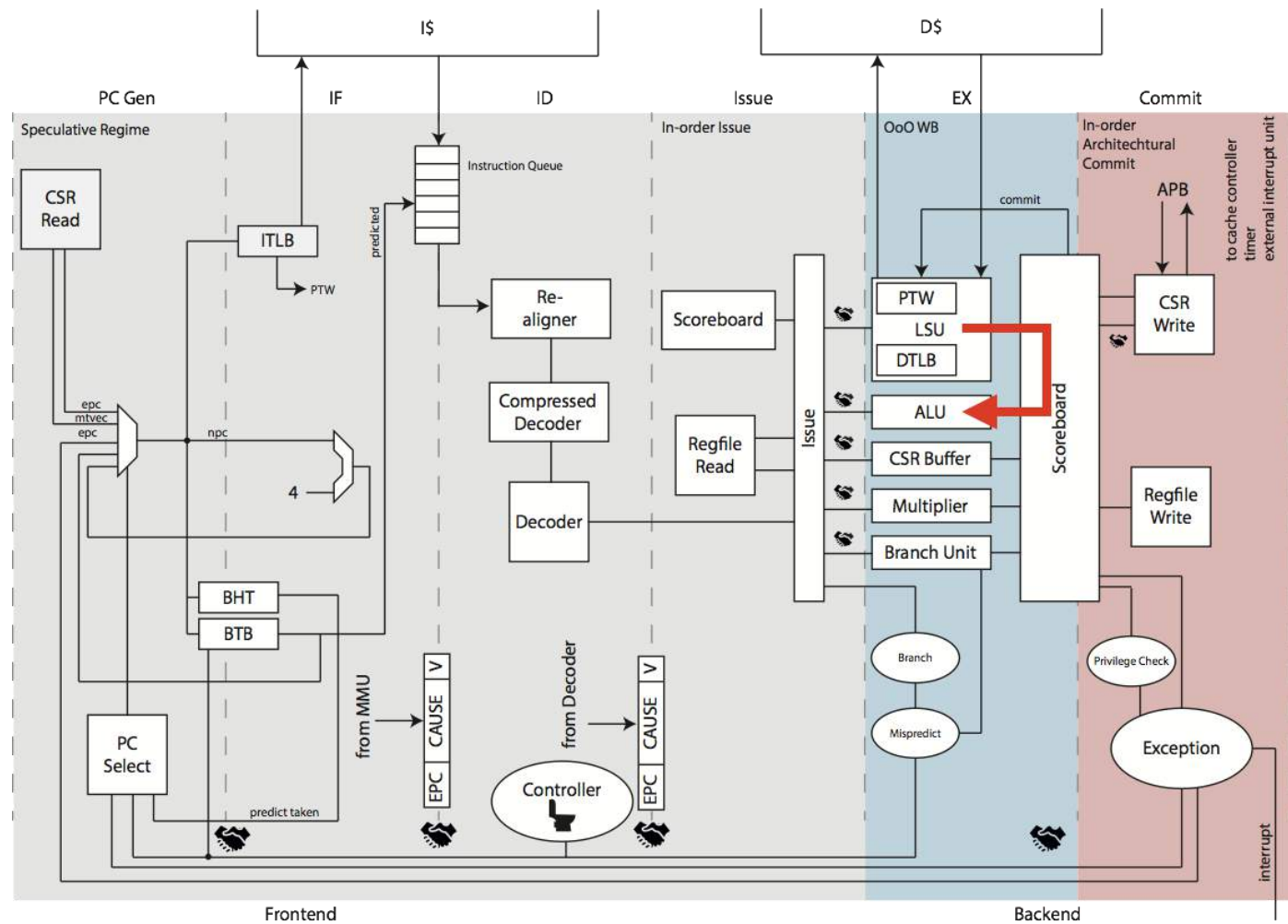
# Frequency-IPC trade-off

- Frequency:
    - Increase frequency through pipelining
    - Modern Intel CPUs have around 10 - 20 pipeline stages
  - Adds significant complexity on the cache interfaces
- 
- Increased bubbles due to:
    - Data Hazards → **Forwarding**
    - Structural Hazards → **Scoreboard**
    - Control Hazards → **Branch Prediction**

# Data Hazards - Forwarding

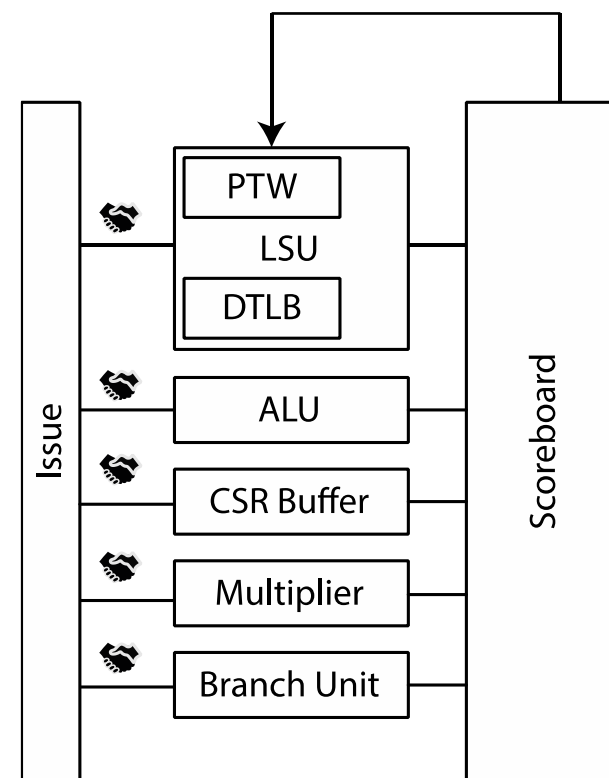
Data Hazards → Forwarding

ld x2, 0(x7)  
addi x3, x2, 10



# Scoreboarding

- Hide latency of multi-cycle instructions
- Clean and modular interface to functional units → scalability (FPU)
- Add issue port: Dual-Issue implementation
- Split execution into four steps:
  - **Issue**: Relatively complex issue logic (extra pipeline-stage)
  - **Read Operands**: From register file or forwarded
  - **Execute**
  - **Write Back**: Mitigate structural hazards on write-back path
- Mitigate structural hazards on write-back port
- Implemented as a **circular buffer**



2 cycles

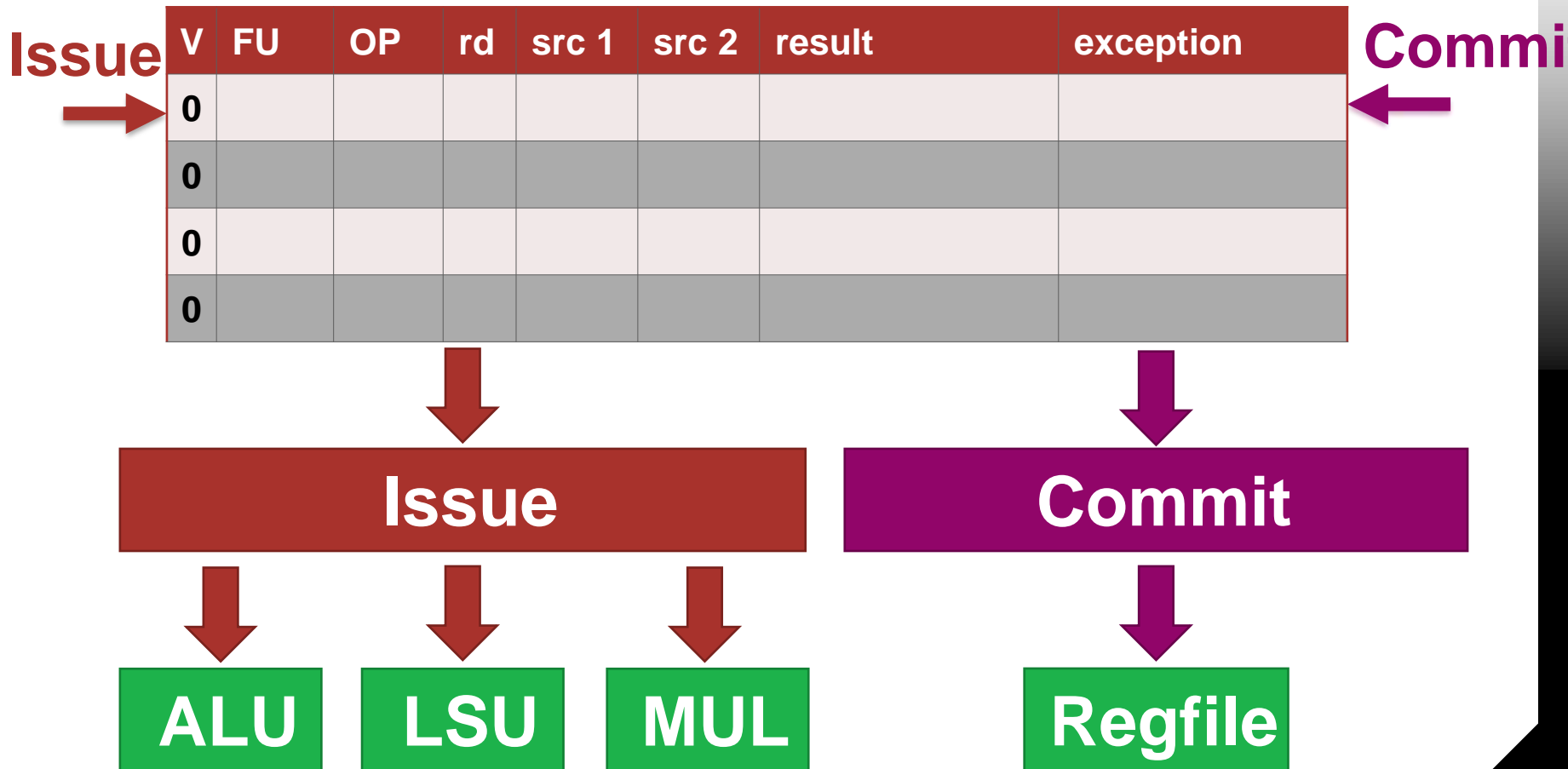
**mul** x4, x4, x5  
**addi** x3, x2, 10

retire same cycle

# Scoreboard

Decode:

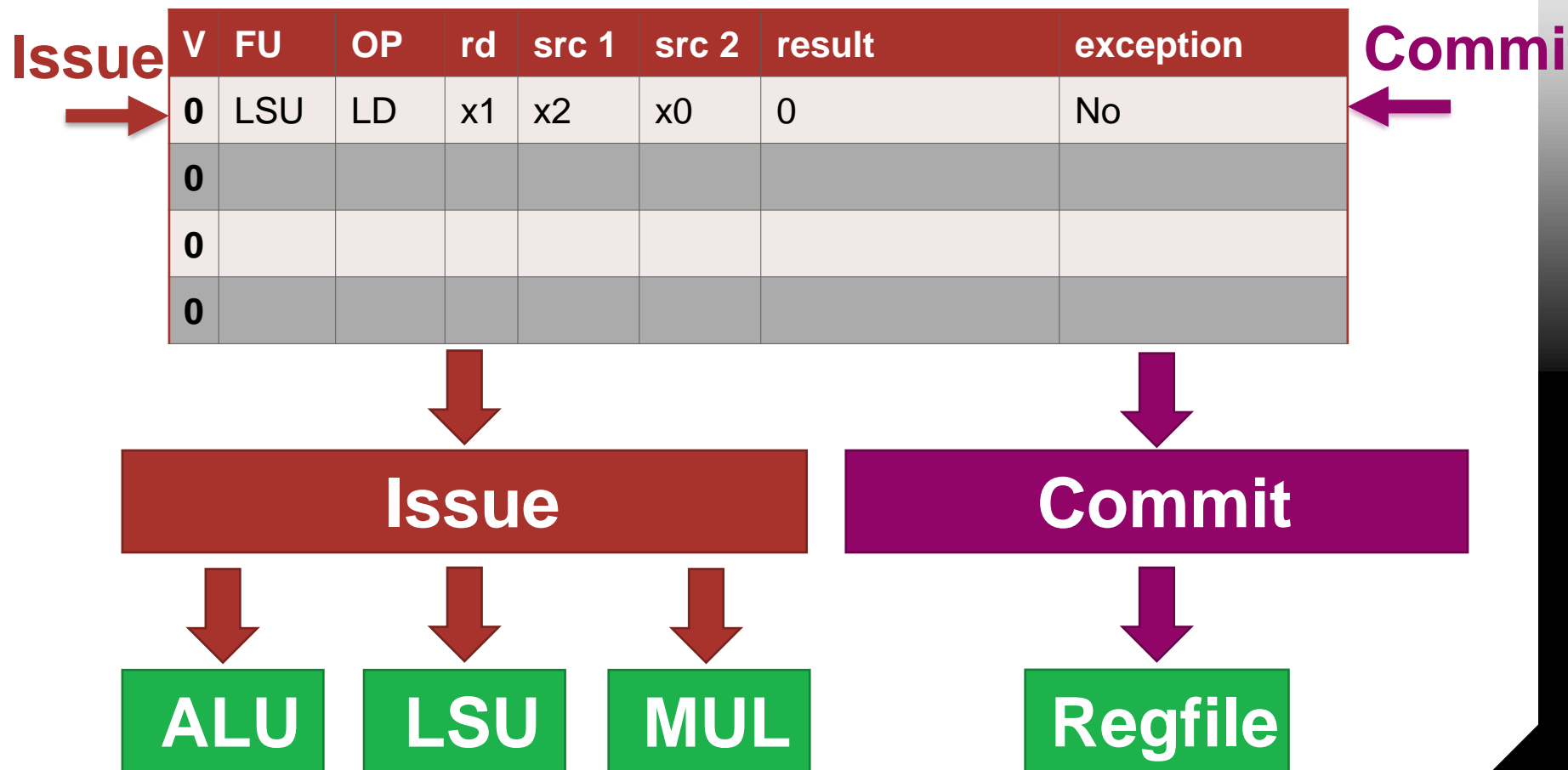
ld x1 ← x2(0)



# Scoreboard – 3 cycle instruction (LD)

Decode:

ld x1 ← x2(0)



# Scoreboard – 2 cycle instruction (MUL)

Decode:

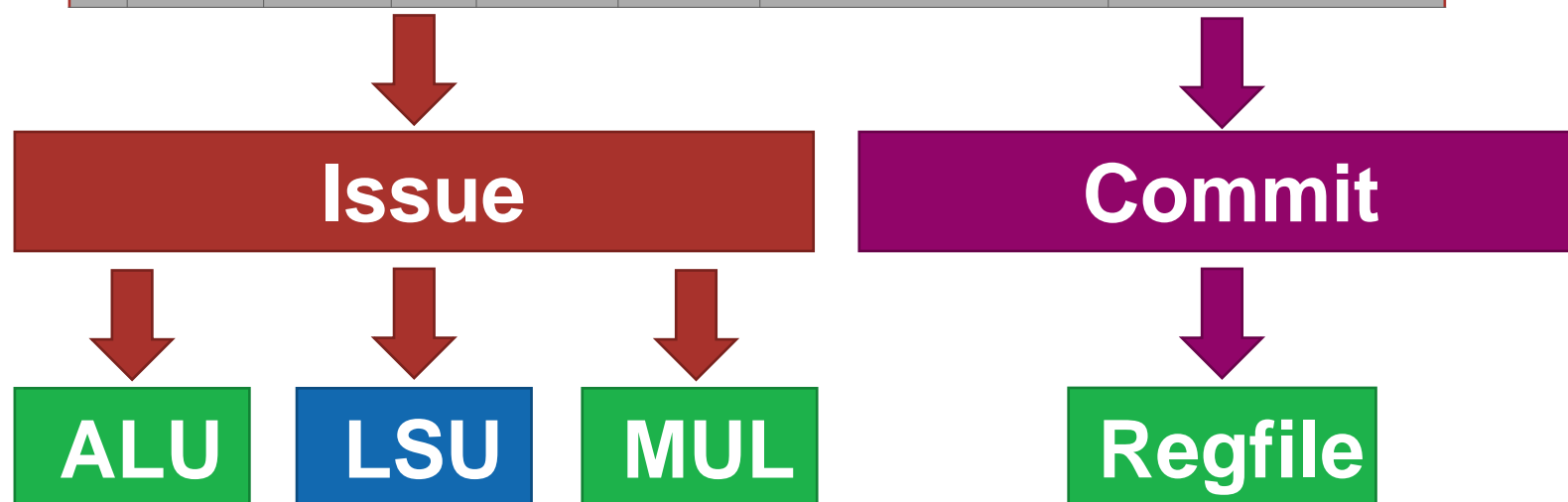
ld x1 ← x2(0)

mul x2 ← x2, x2

Issue

V	FU	OP	rd	src 1	src 2	result	exception
0	LSU	LD	x1	x2	x0	0	No
0	MUL	MUL	x2	x2	x2	0	No
0							
0							

Commit



# Scoreboard - single cycle instruction (ALU)

Decode:

ld x1 ← x2(0)

mul x2 ← x2, x2

addi x3 ← x0, 16

Issue

V	FU	OP	rd	src 1	src 2	result	exception
0	LSU	LD	x1	x2	x0	0	No
0	MUL	MUL	x2	x2	x2	0	No
0	ALU	ADD	x3	x0	x0	16	No
0							

Commit

Issue

Commit

ALU

LSU

MUL

Regfile

# Scoreboard – Multiple Write Back

Decode:

ld x1 ← x2(0)

mul x2 ← x2, x2

addi x3 ← x0, 16

addi x2 ← x3, x1

Issue



V	FU	OP	rd	src 1	src 2	result	exception
1	LSU	LD	x1	x2	x0	0x5555	No
1	MUL	MUL	x2	x2	x2	0xAAAA	No
1	ALU	ADD	x3	x0	x0	16	No
0	ALU	ADD	x2	x3	x1	0	No

forward



Issue



ALU

LSU

MUL

Commit



Regfile

Commit





# Scoreboard - Commit

Decode:

ld x1 ← x2(0)

mul x2 ← x2, x2

addi x3 ← x0, 16

addi x2 ← x3, x1

ld x1 ← x2(128)

Issue



V	FU	OP	rd	src 1	src 2	result	exception
0	LSU	LD	x1	x2	x0	128	No
1	MUL	MUL	x2	x2	x2	0xAAAA	No
1	ALU	ADD	x3	x0	x0	16	No
1	ALU	ADD	x2	x3	x1	0xAABA	No

Commit



Issue

Commit

ALU

LSU

MUL

Regfile

# Scoreboard – Exception

Decode:

ld x1  $\leftarrow$  x2(0)

mul x2  $\leftarrow$  x2, x2

addi x3  $\leftarrow$  x0, 16

addi x2  $\leftarrow$  x3, x1

ld x1  $\leftarrow$  x2(128)

Issue



V	FU	OP	rd	src 1	src 2	result	exception
1	LSU	LD	x1	x2	x0	128	Yes
0							
1	ALU	ADD	x3	x0	x0	16	No
1	ALU	ADD	x2	x3	x1	0xAABA	No

Commit



Issue

Commit

ALU

LSU

MUL

Regfile

# Scoreboard - Commit

Decode:

ld x1  $\leftarrow$  x2(0)

mul x2  $\leftarrow$  x2, x2

addi x3  $\leftarrow$  x0, 16

addi x2  $\leftarrow$  x3, x1

ld x1  $\leftarrow$  x2(128)

Issue



V	FU	OP	rd	src 1	src 2	result	exception
1	LSU	LD	x1	x2	x0	128	Yes
0							
0							
1	ALU	ADD	x2	x3	x1	0xAABA	No

Commit



Issue

Commit

ALU

LSU

MUL

Regfile

# Scoreboard - Commit

Decode:


ld x1  $\leftarrow$  x2(0)

mul x2  $\leftarrow$  x2, x2


addi x3  $\leftarrow$  x0, 16

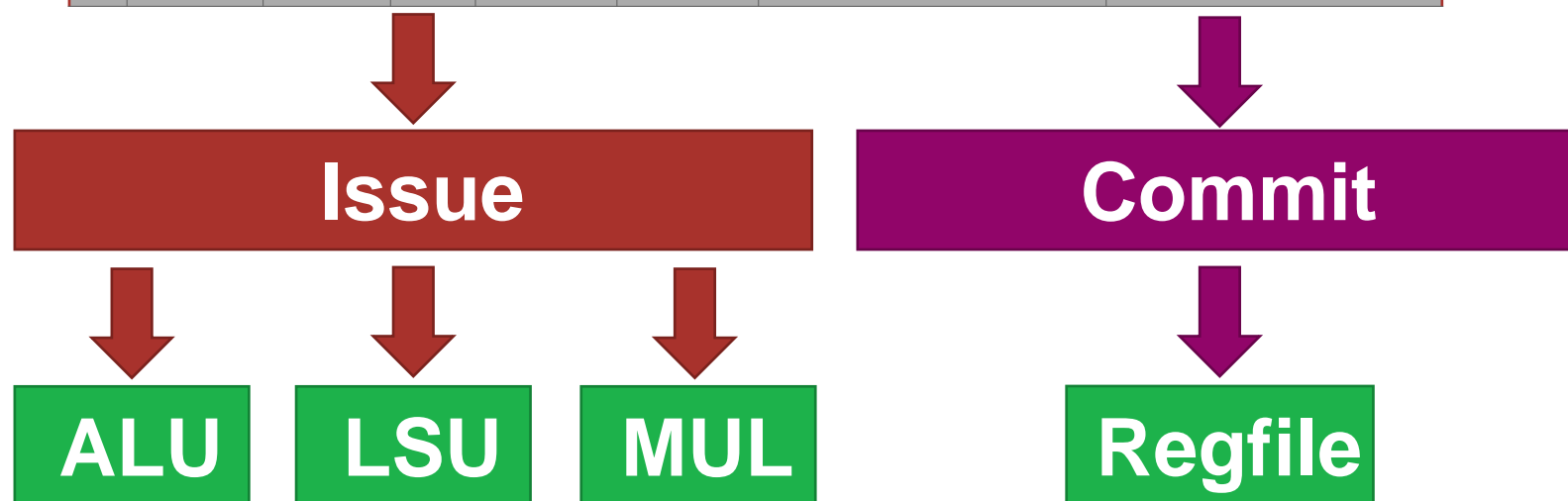
addi x2  $\leftarrow$  x3, x1

ld x1  $\leftarrow$  x2(128)

**Issue** 

V	FU	OP	rd	src 1	src 2	result	exception
1	LSU	LD	x1	x2	x0	128	Yes
0							
0							
0							

**Commit** 



# Scoreboard - Commit

Decode:

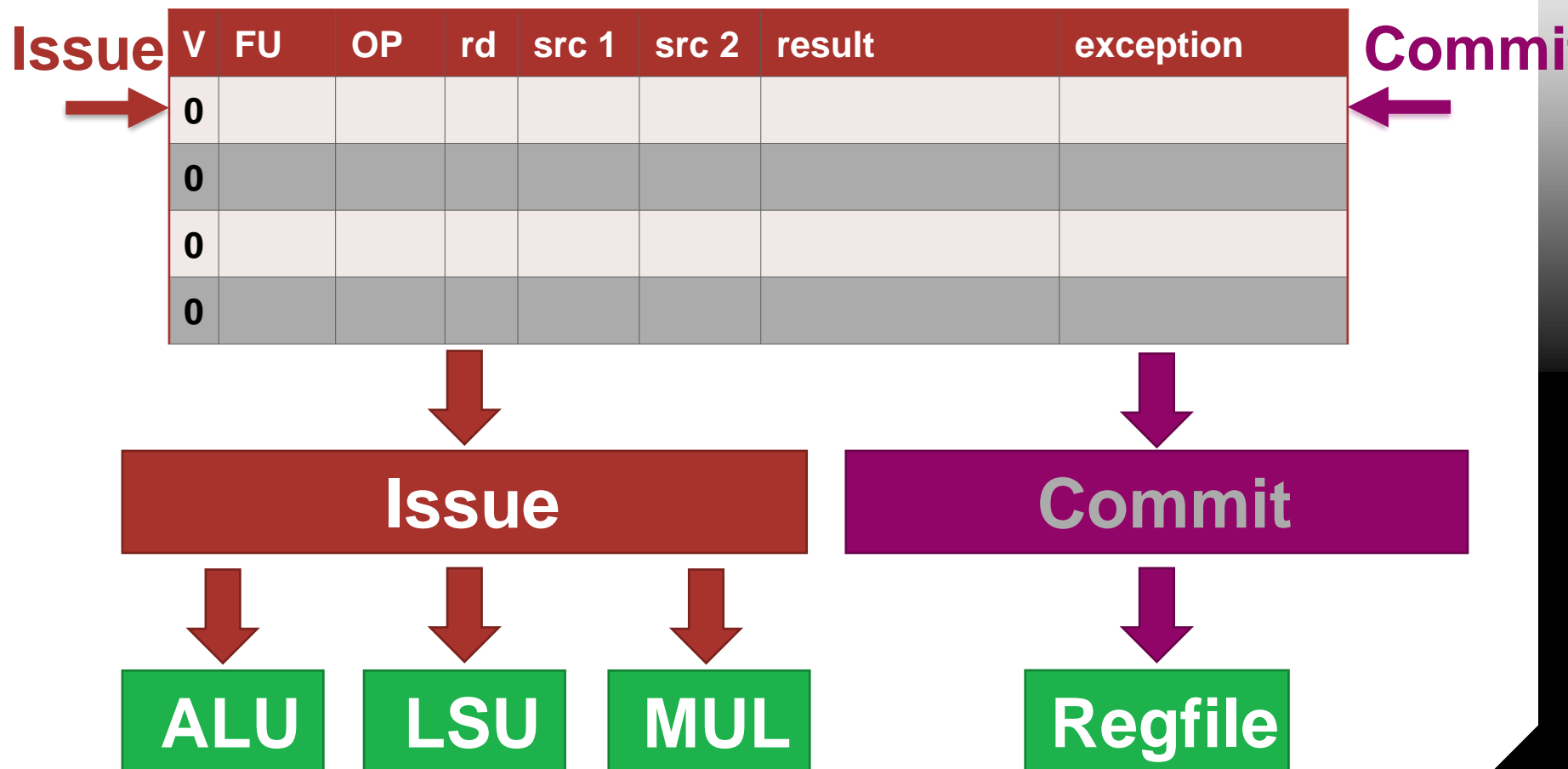
ld x1 ← x2(0)

mul x2 ← x2, x2

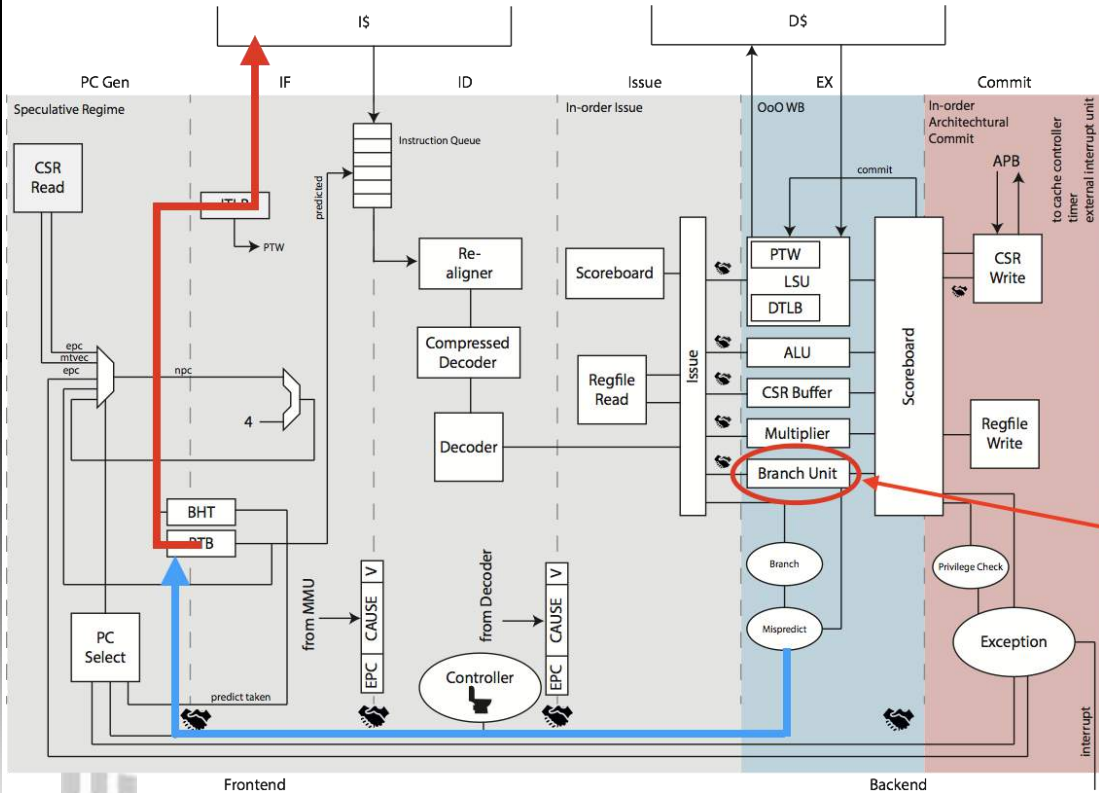
addi x3 ← x0, 16

addi x2 ← x3, x1

ld x1 ← x2(128)



# Branch Prediction

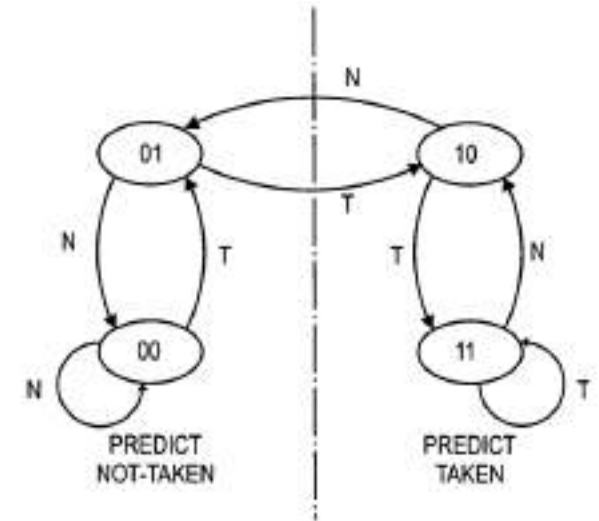


Control Hazards

→ Branch Prediction

taken?  
**bneq** x4, x6, pc + 16  
**addi** x3, x2, 10

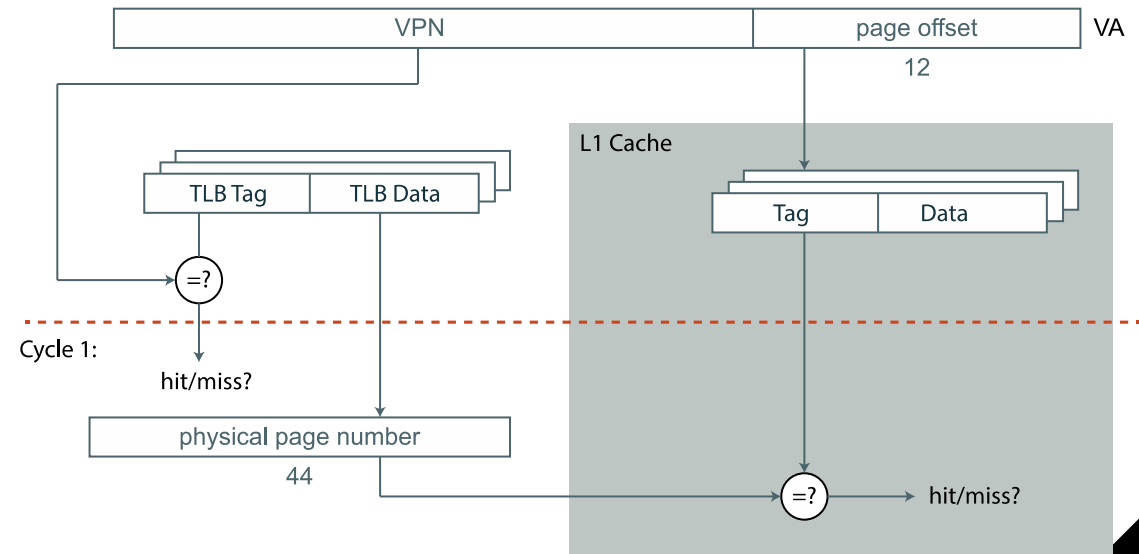
- Branch Target Buffer
- Branch History Table
- 2-bit saturation counter
- Anti-aliasing bits



# Caches

- Caches are a **necessity** for larger systems
- Private L1 caches
  - I\$ (16 kByte, 32 entries, 16 byte cache line, 4 way) - **1,41% MR** (Linux Boot)
  - D\$ (32 kByte, 32 entries, 16 byte cache line, 8 way) - **3,17% MR** (Linux Boot)
- L2 cache (outside core domain)
- **SRAMs** (cache memories) are **slow** compared to regular logic
- Virtually indexed, physically tagged data cache

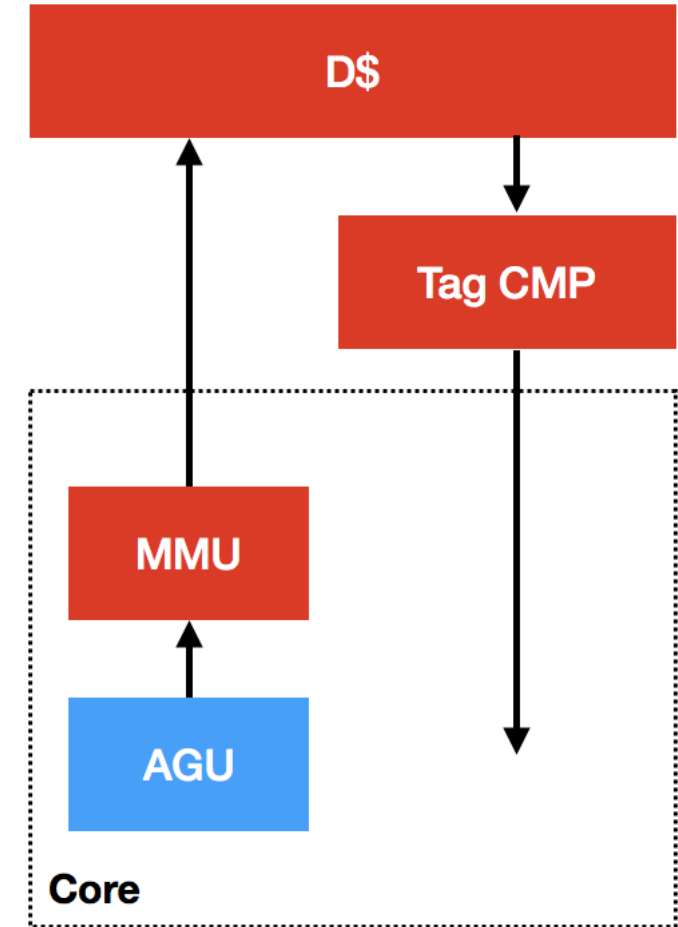
Cycle 0:





# Memory Interfaces

- **Load** and **stores** are very **common** in RISC architectures
- **Caches add** (costly) tag-comparison
- **Address translation adds** to this already critical path
- A fast CPU design needs to account for these effects as much as possible
  - Virtually indexed, physically tagged caches
  - De-skewing

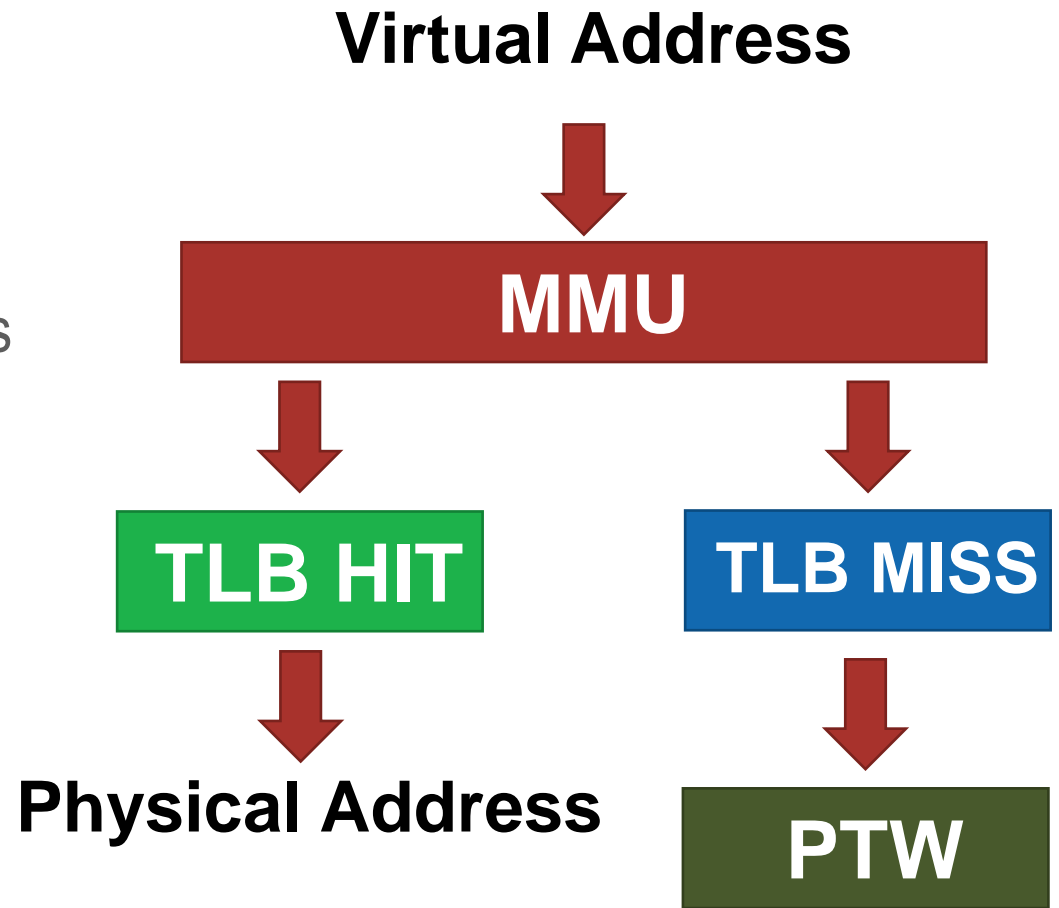






# Memory Management Unit (MMU)

- Essential for supporting Linux
- **Ariane** implements 39-bit page based address translation (**SV39**)
- SV39 supports three levels of page tables
  - 1<sup>st</sup> level: 1 gigabit-pages
  - 2<sup>nd</sup> level: 2 megabit-pages
  - 3<sup>rd</sup> level: (regular) 4 kilobit-pages
- Configurable number of TLB entries
- **Hardware page table walker** allows for efficient TLB miss management





# Hardware Page Table Walker (HPTW)

Virtual Page Number 2

Virtual Page Number 1

Virtual Page Number 0

Physical Page Number Base (CSR)

V	R	W	X	A	D	G	PPN
1	0	0	0	0	0	0	0xDEADBEEF
:	:	:	:	:	:	:	:
1	1	0	0	0	0	0	0xCOFFEEBABB

Pointer to next level PTE

Leaf Node (1 Gigapage)



# Second Level Page Table

Virtual Page Number 2

Virtual Page Number 1

Virtual Page Number 0

V	R	W	X	A	D	G	PPN
1	0	0	0	0	0	0	0xDEADBEEF
:	:	:	:	:	:	:	:
1	1	0	0	0	0	0	0xCOFFEEBABE

Analogous with third  
and fourth level

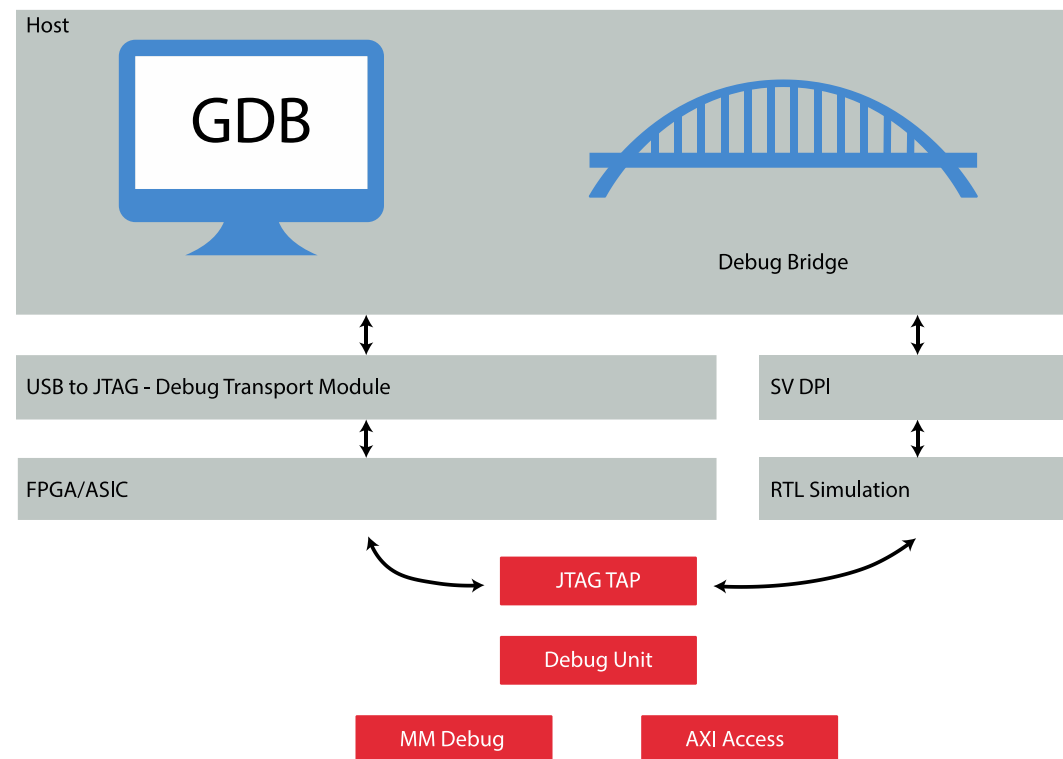
V	R	W	X	A	D	G	PPN
1	0	0	0	0	0	0	0xAAAAAAAAAA
:	:	:	:	:	:	:	:
1	1	0	0	0	0	0	0xBBBBBBBBBBBB

Pointer to next level PTE

Leaf Node (2 Megapage)

# Full Debug support

- JTAG interface
- OpenOCD support
- **Debug Bridge** to communicate with hardware
- Allows for:
  - run-control
  - single-step
  - inspection
  - (hardware) breakpoints
- Essential for SW debug and hardware bring-up

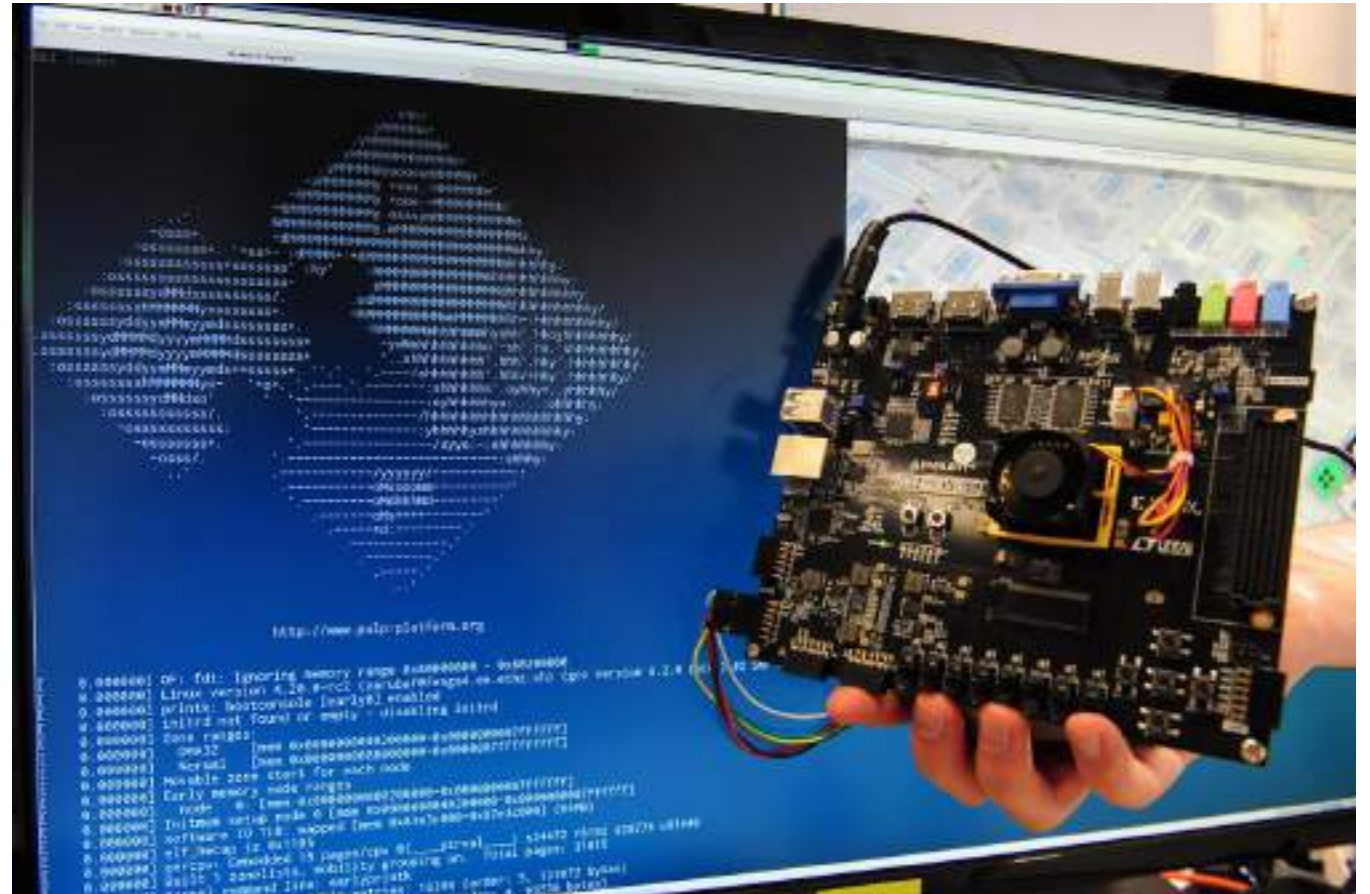


- **16 performance counters - not yet RISC-V standard**
- **Trace task group working on PC tracing (we participated)**



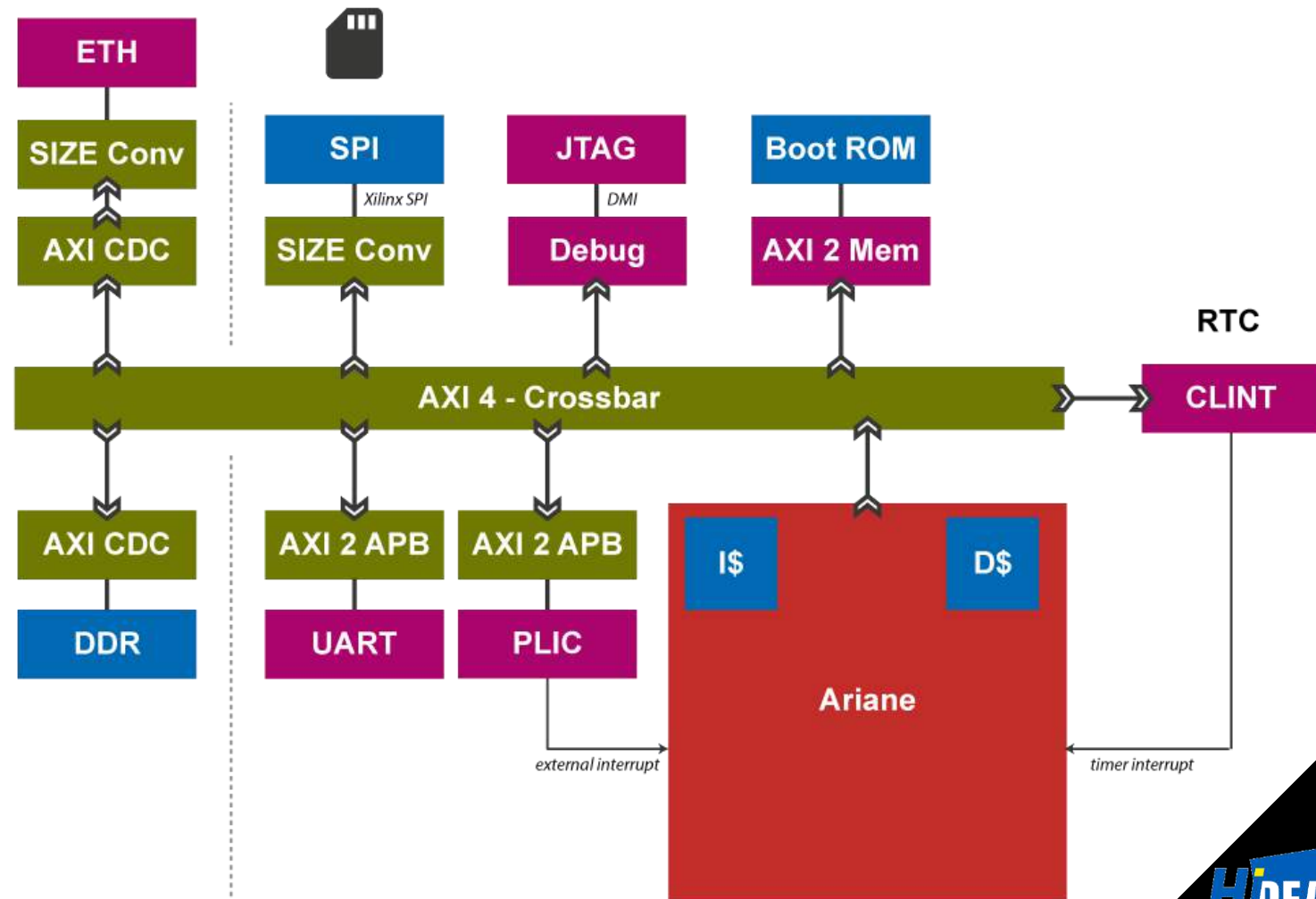
# Verification

- RISC-V Tests
- Torture-Tests: constrained random verification
- Google UVM-based Verification framework
- CI-tests
- FPGA mapping



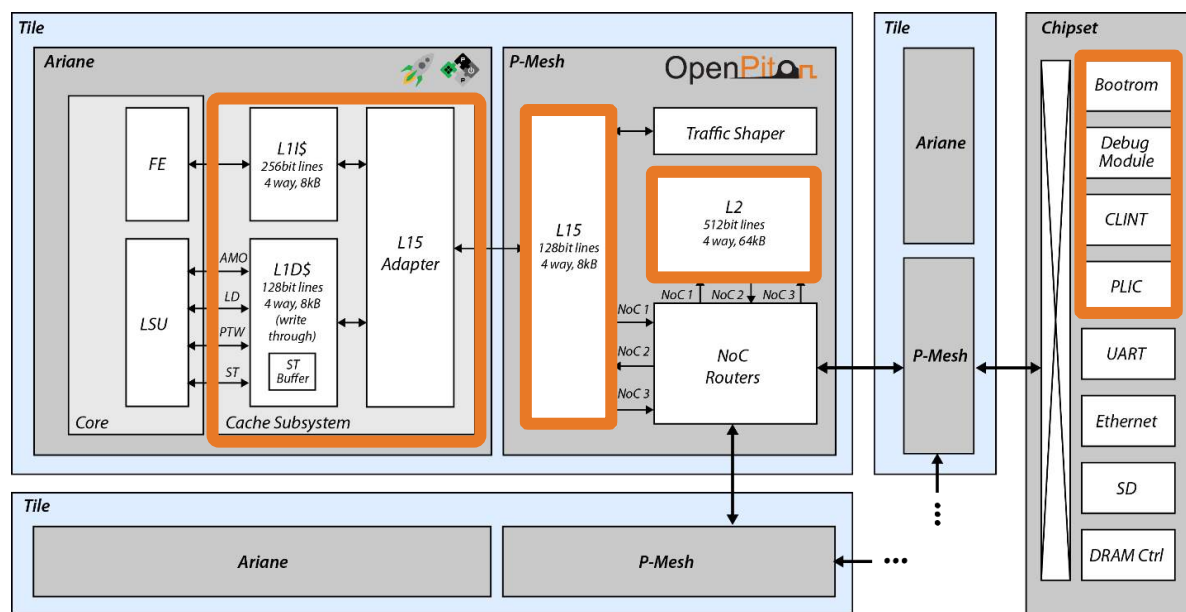
# Minimal Ariane SoC

- Minimum set of peripherals to boot Linux
- Code is on SD Card
- Zero-stage bootloader is on SystemVerilog boot ROM
- Serial I/O



# OpenPiton + Ariane

If you are really passionate about cache coherent “scalable” machines...



- Boots SMP Linux
- New write-through cache subsystem with invalidations and the TRI interface
- LR/SC in L1.5 cache
- Fetch-and-op in L2 cache
- RISC-V Debug
- RISC-V Peripherals

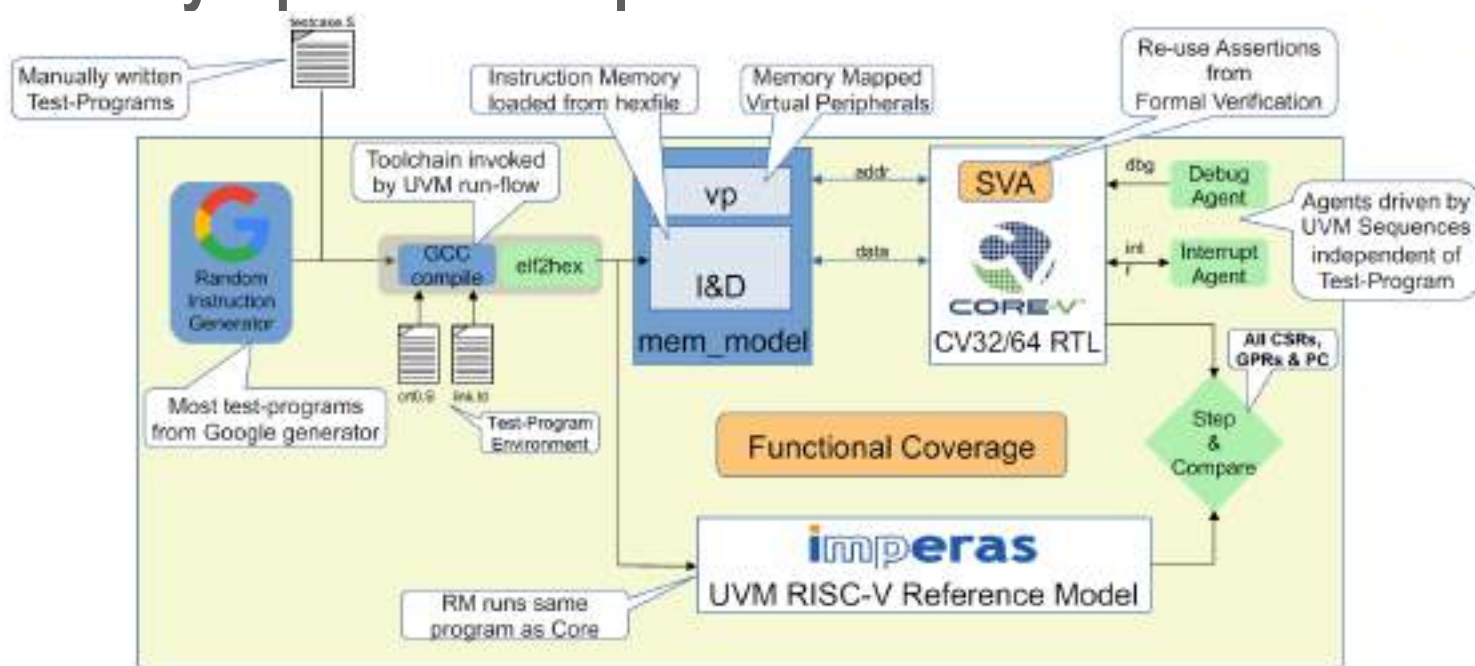
[OpenPiton+ Ariane: The First Open-Source, SMP Linux-booting RISC-V System Scaling From One to Many Cores](#)

OpenPiton



# Open-sourcing

- Ariane has been **open-sourced in February 2018**
  - Continued development on public GitHub servers
- We provided a **Verilator port** for an easy first evaluation
- Now supported by OpenHWGroup → **CV6A**



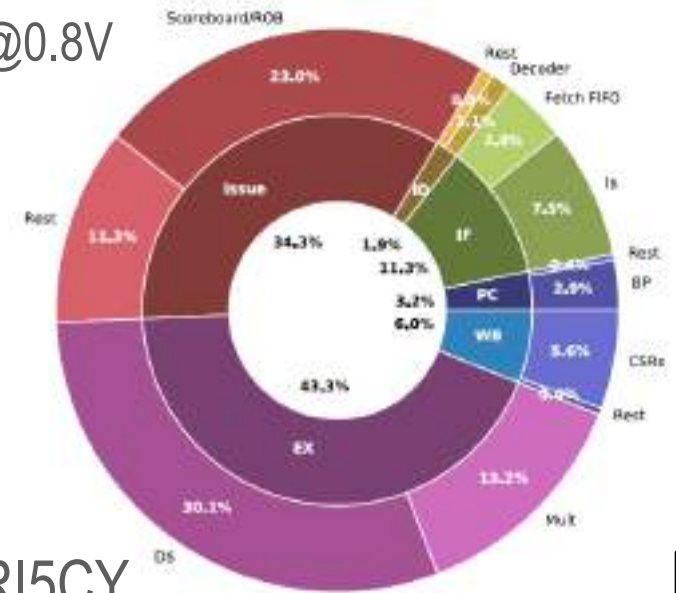
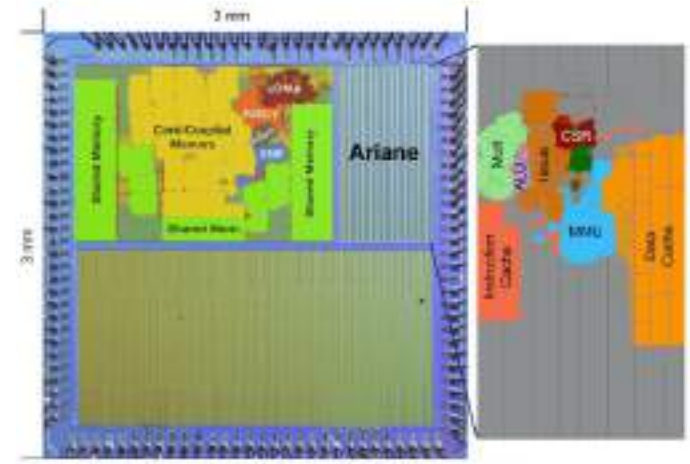
make SIMULATOR=<sim> +UVM\_TEST=riscv-dv-test



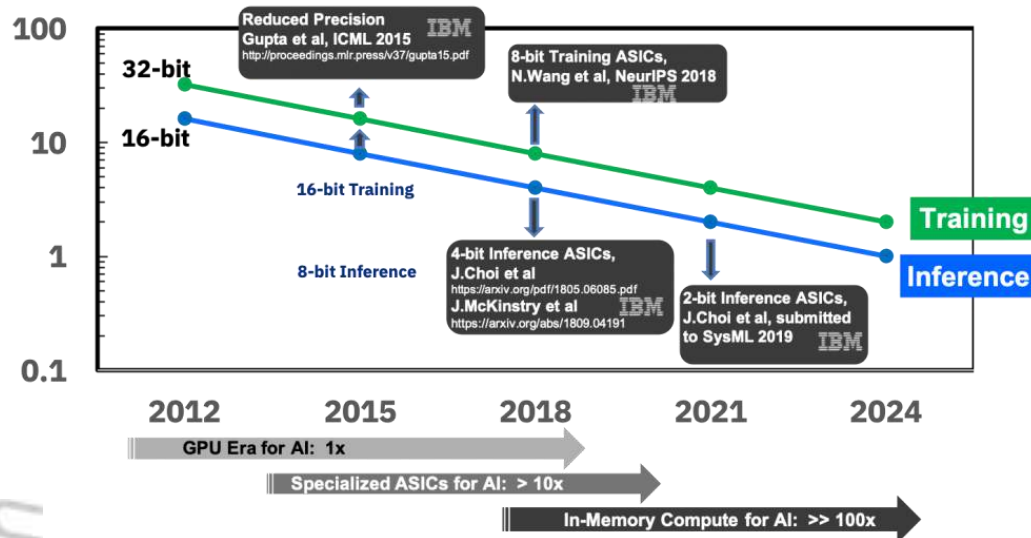


# Ariane on Silicon

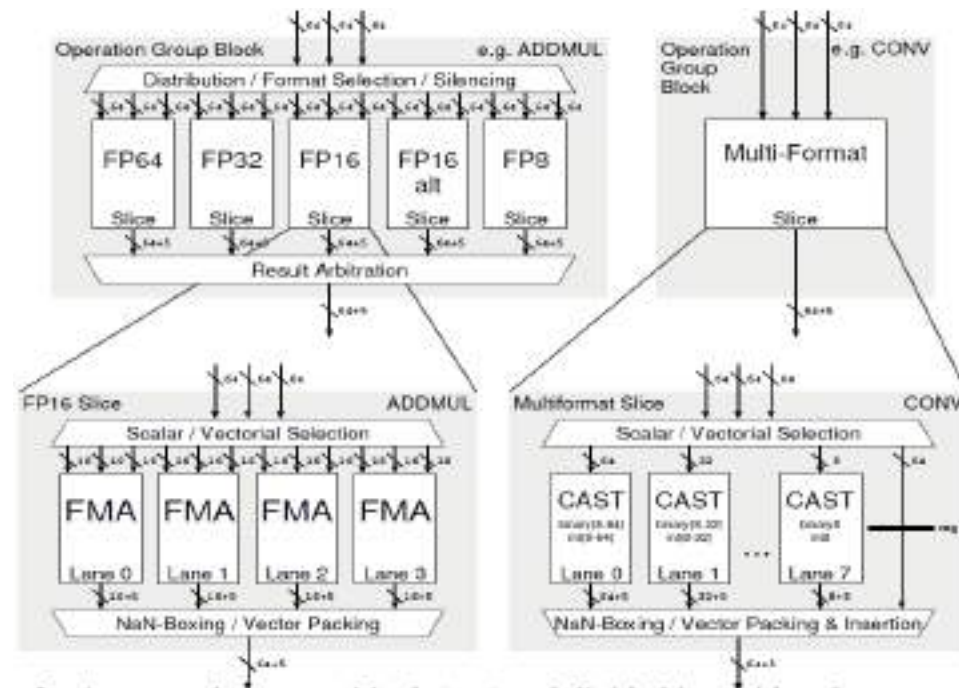
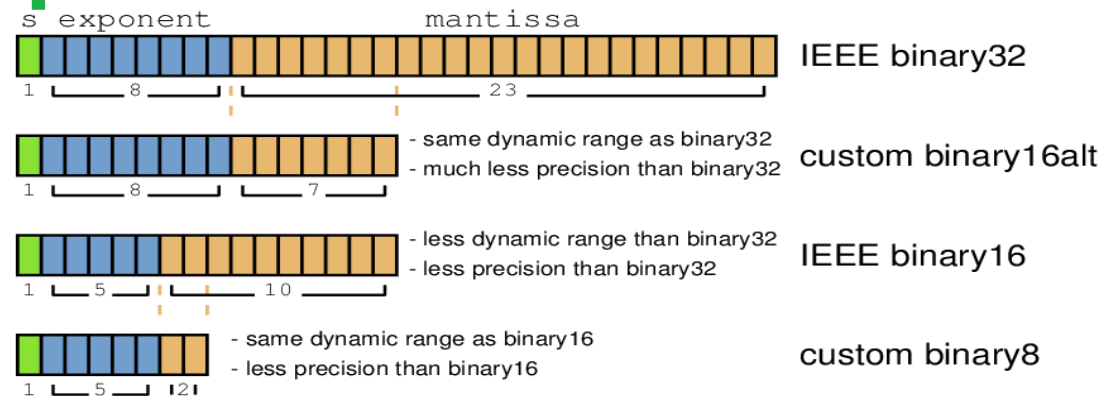
- **6-stage, integrated cache**
  - In order issue, out-of-order write-back, in-order-commit
  - Supports privilege spec 1.11, M, S and U modes
  - Hardware Page Table Walker
- **Implemented in GF 22FDX (Poseidon, Kosmodrom, Baikonur), UMC65 (Scarabaeus)**
  - In 22nm: ~0.9GHz WC (SSG, 125/-40C, 0.72V), 1.1GHz Meas @0.8V
  - 8-way 32kByte D\$, 4-way 32kByte I\$
  - Core area: 175 kGE (210 with TP FPU)
- **Application-class features are not cheap**
  - 38% area in TLB, PTW, 23% scoreboard
  - **51.8**pJ/op vs. **10**pJ/OP in 22FDX @ 0.8V vs. RI5CY
  - IPC 0.85 vs. 0.94, 1.7GHz vs. 690, just **2.1** faster than RI5CY



# Low-Bitwidth Floating point Formats

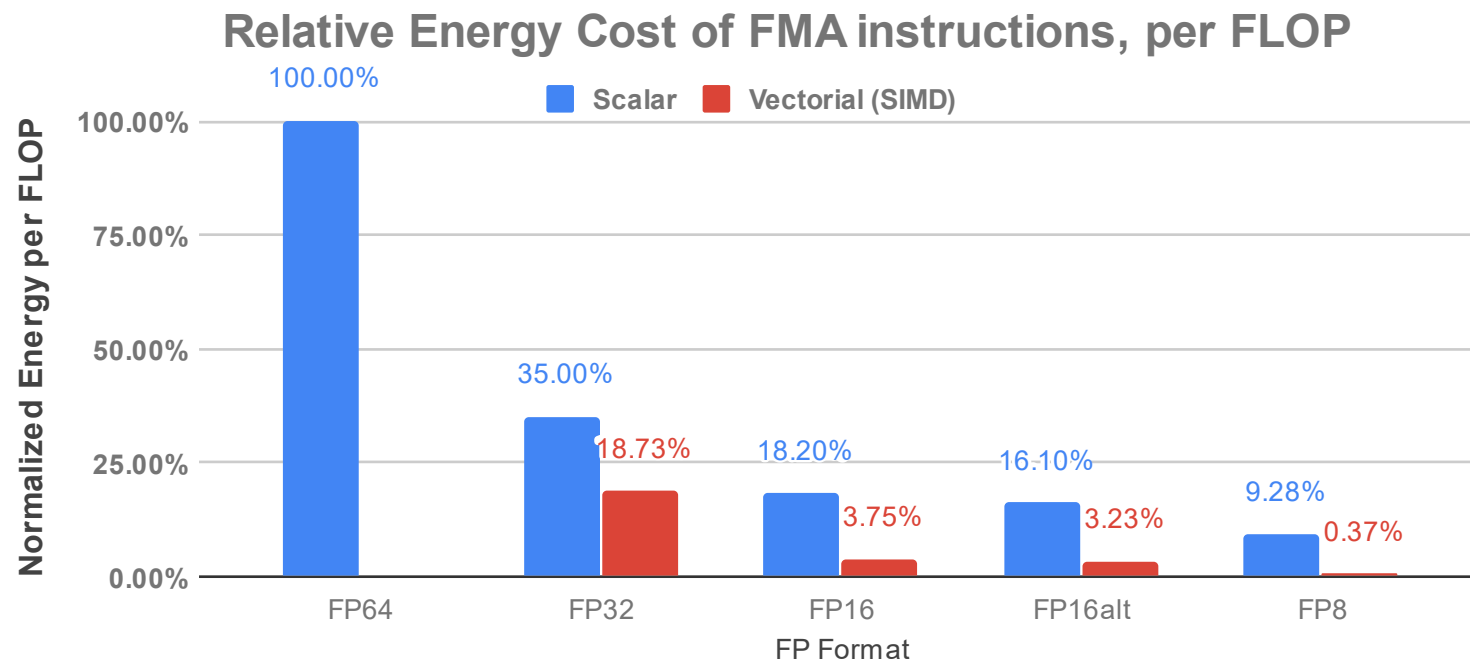


- Flexible (cycle by cycle) precision modulation (FP) → **transprecision**
- Save precious DRAM bandwidth
  - Custom number formats
  - Use float8, float16, float16alt

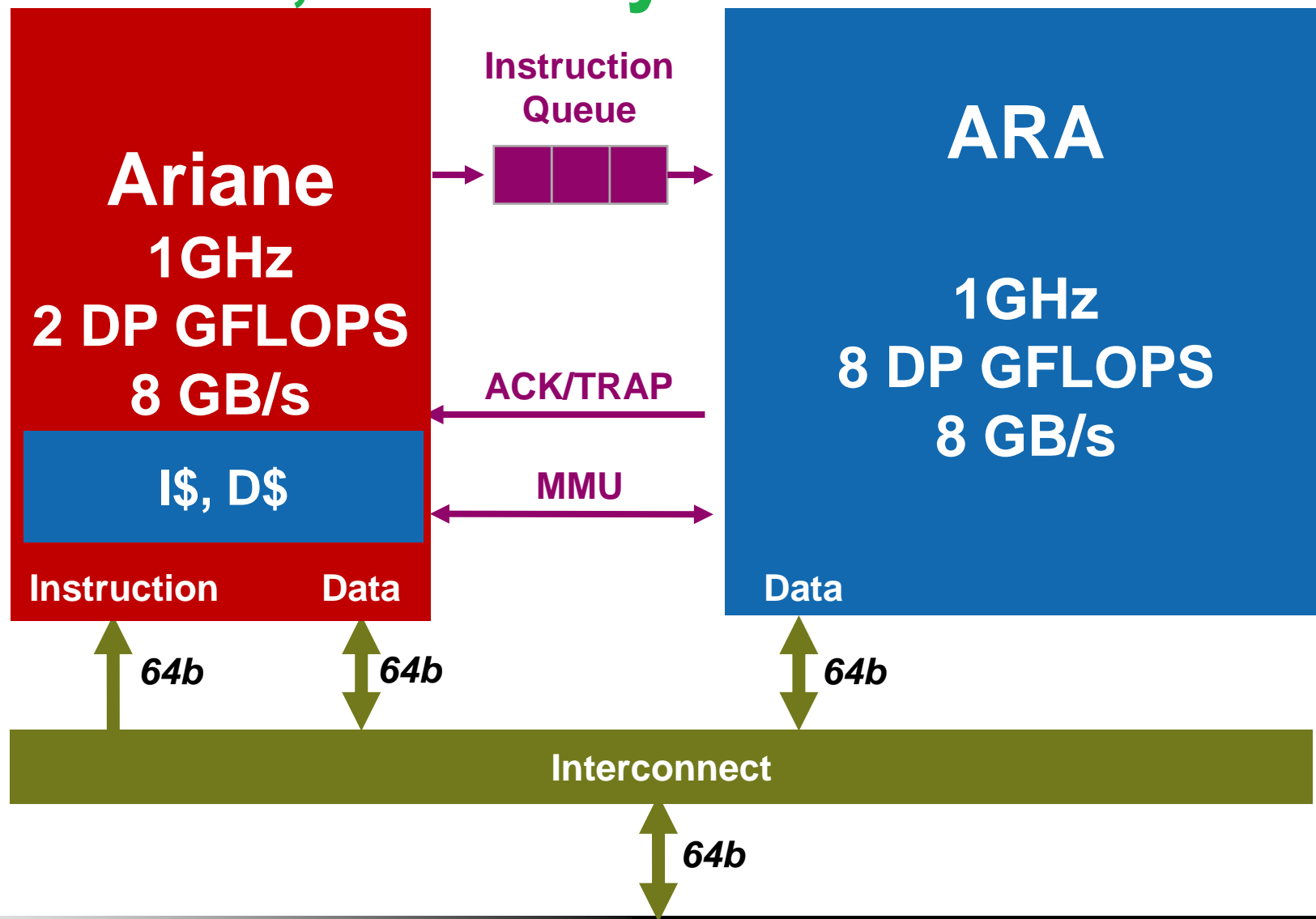


# FP Precision and Energy trade-off

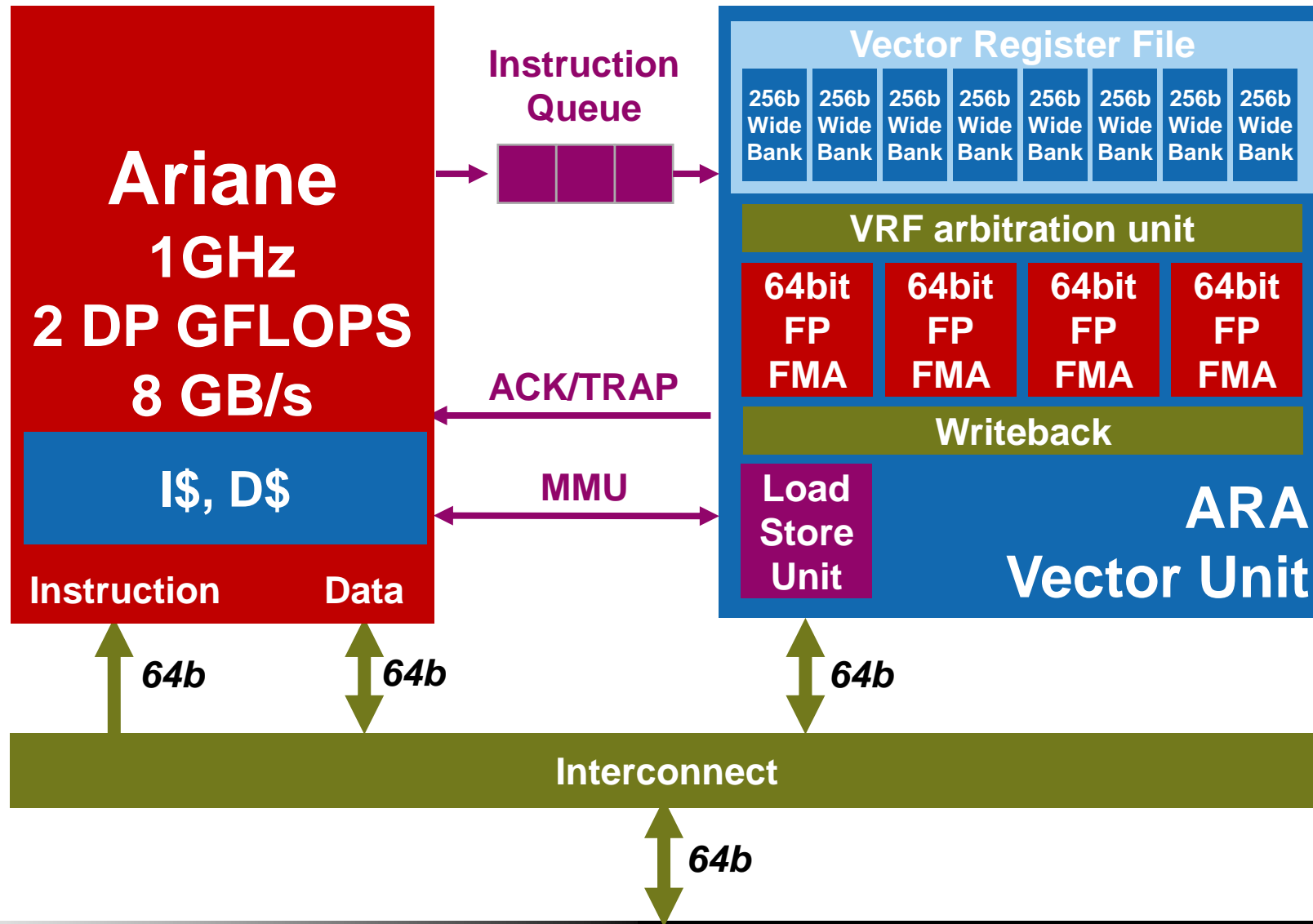
- Trade-off floating point precision for instruction energy
- Energy cost of FP operations is super linearly proportional to data width
- Smaller FP formats take less latency to complete
- SIMD style vectors yield higher throughput
- Improve energy to solution and time to solution up to **7.95x** and **7.6x** for FP8 workloads



# More FP Perf, efficiency: The “V” Extension

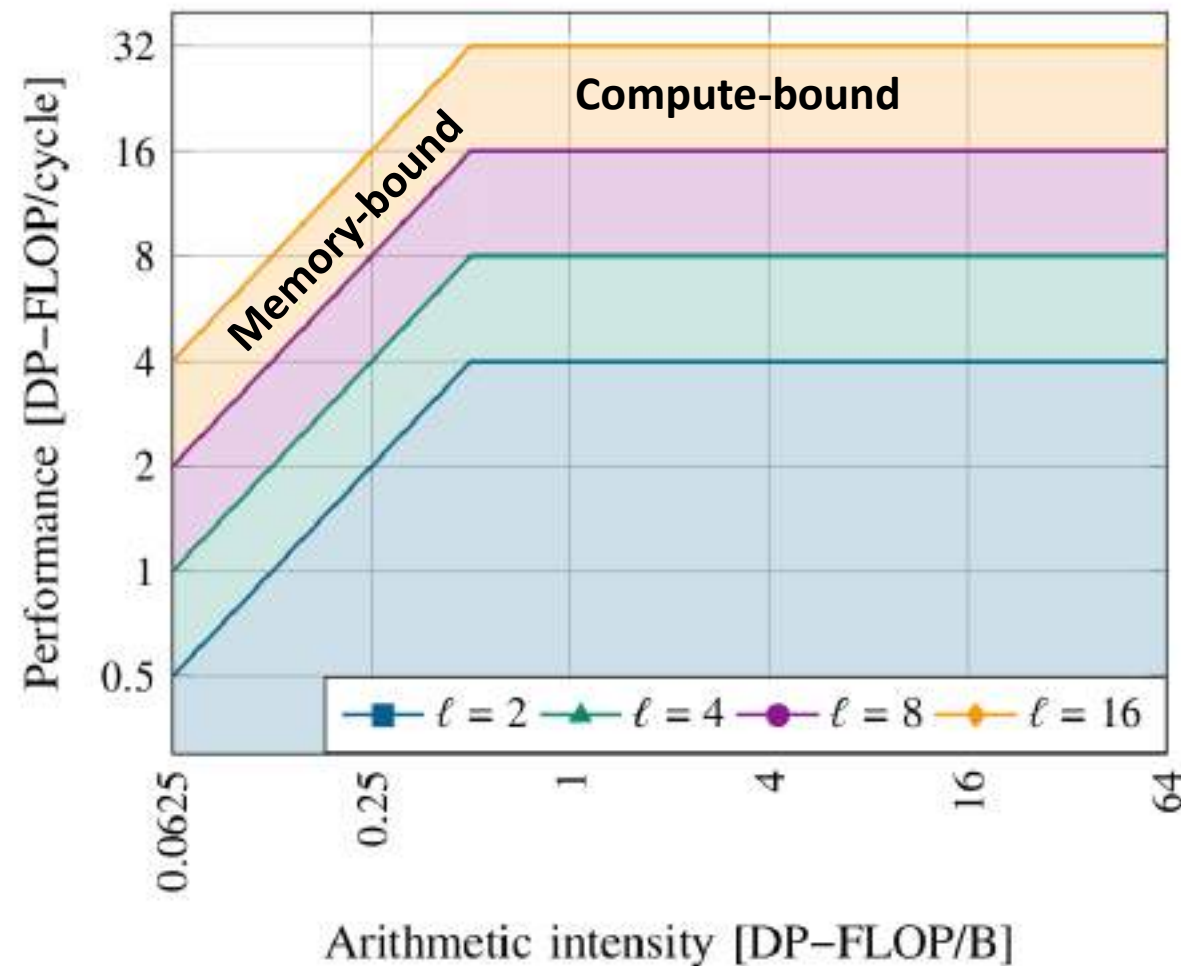


# Extreme FP Performance: The “V” Extension



# Memory Bandwidth

- Arithmetic intensity  
Operations per byte: data reuse of algorithm  
One FMA  $\rightarrow$  two operations
- Memory-boundness and compute-boundness
- Ara targets 0.5 DP-FLOP/B  
Memory bandwidth scales with the number of physical lanes





# RISC-V Vector Extension

- RISC-V “V” Extension

Cray-like vector processing, opposed to packed-SIMD



- Ara is based on the version 0.5

Work is being done to update it to the latest version

Open-source in 2020 (Q3)



# Ara main datapath elements

- ALU, MUL and FPU
- Transprecision functional units

Throughput of 64 bit per cycle

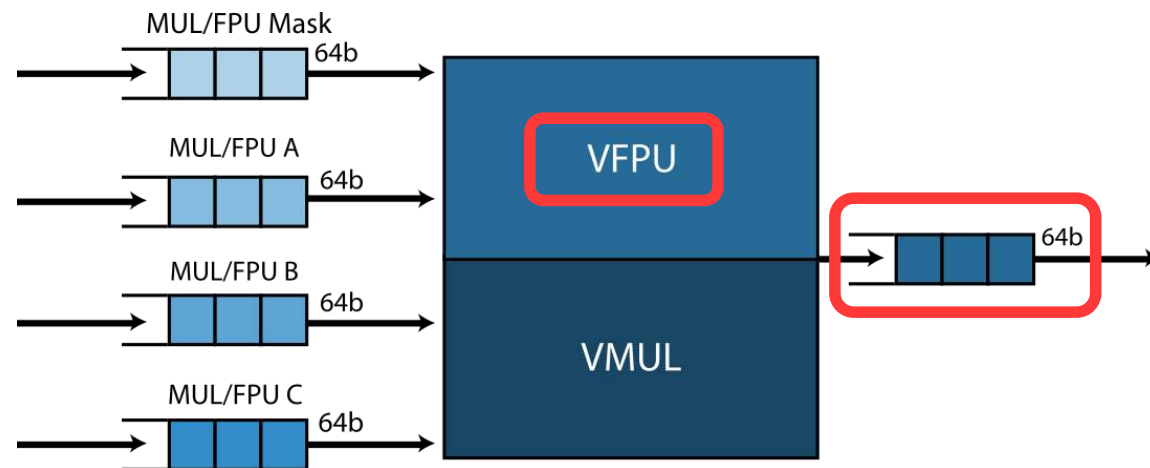
Packed-SIMD approach

- FPU

FP64, FP32, FP16, bfloat16

Independent pipelines for each data type

- Each with a different latency

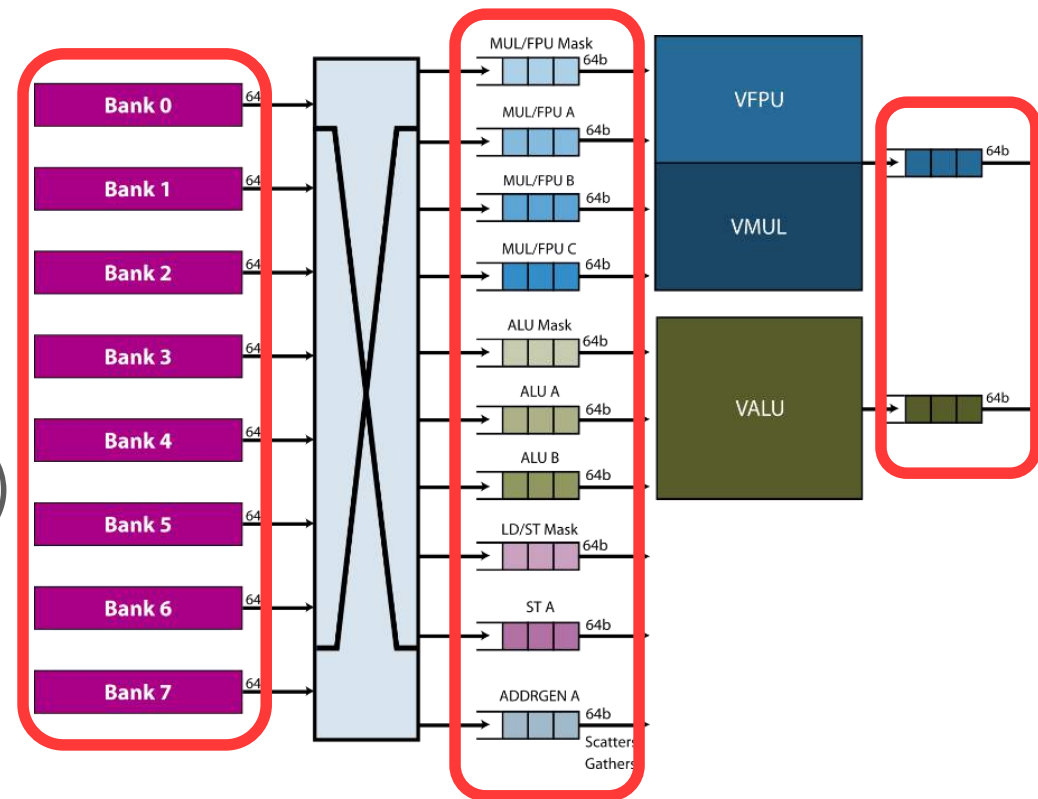


M. Cavalcante, F. Schuiki, F. Zaruba, M. Schaffner and L. Benini, "Ara: A 1-GHz+ Scalable and Energy-Efficient RISC-V Vector Processor With Multiprecision Floating-Point Support in 22-nm FD-SOI," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 28, no. 2, pp. 530-543, Feb. 2020.



# Vector Lane: base computational unit

- **Per-lane Vector Register File**  
8 x 1RW SRAM banks  
Functional units only access their own section of the VRF  
Requires an arbiter (banking conflicts)
- **Operand queues**  
Hide latency due to banking conflicts on the VRF  
One FIFO per operand per datapath unit:  
10 x 64b queues  
Similar queues for output operands

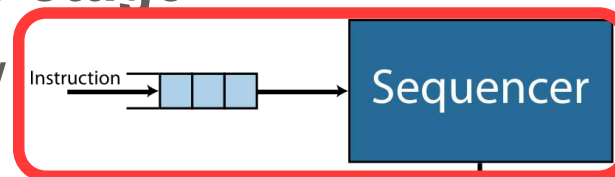


# Ara with $N$ identical vector lanes

- **Instruction forked from Ariane's issue stage**

Instructions are issued non-speculatively

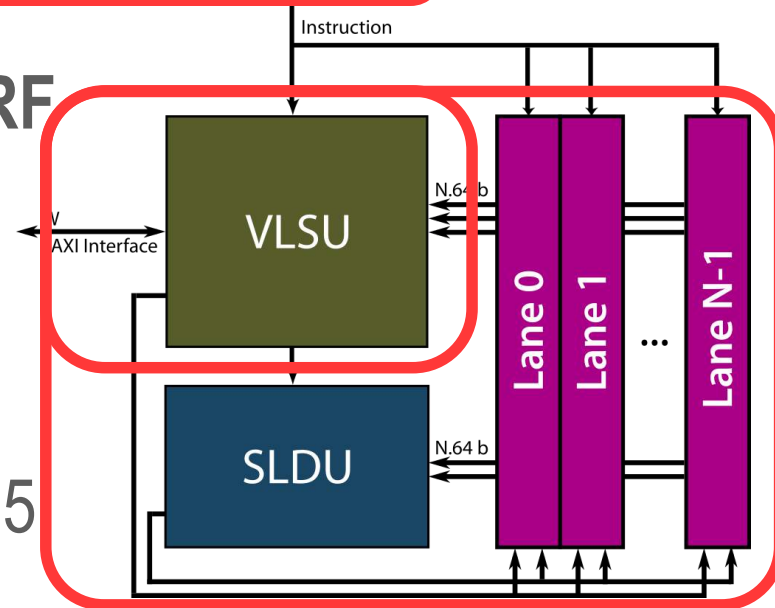
Bookkeeping by the sequencer



- **Load/Store and Slide Units access all the VRF**

Connected to each lane

Scalability issue



- **$W = 32.N$  bits wide memory interface**

Keep Ara performance per bandwidth ratio at 0.5 DP-FLOP/B



# Matrix multiplication on Ara

- Standard algorithm (row times column + reduction) is slow  
Highly sequential
- Use a vector of reductions instead

a00	a01	a02	a03		b00	b01	b02	b03
					b10	b11	b12	b13
				x	b20	b21	b22	b23
					b30	b31	b32	b33



# Matrix multiplication on Ara

- Standard algorithm (row times column + reduction) is slow  
Highly sequential
- Use a vector of reductions instead

a00	a01	a02	a03

x

b00	b01	b02	b03
b10	b11	b12	b13
b20	b21	b22	b23
b30	b31	b32	b33

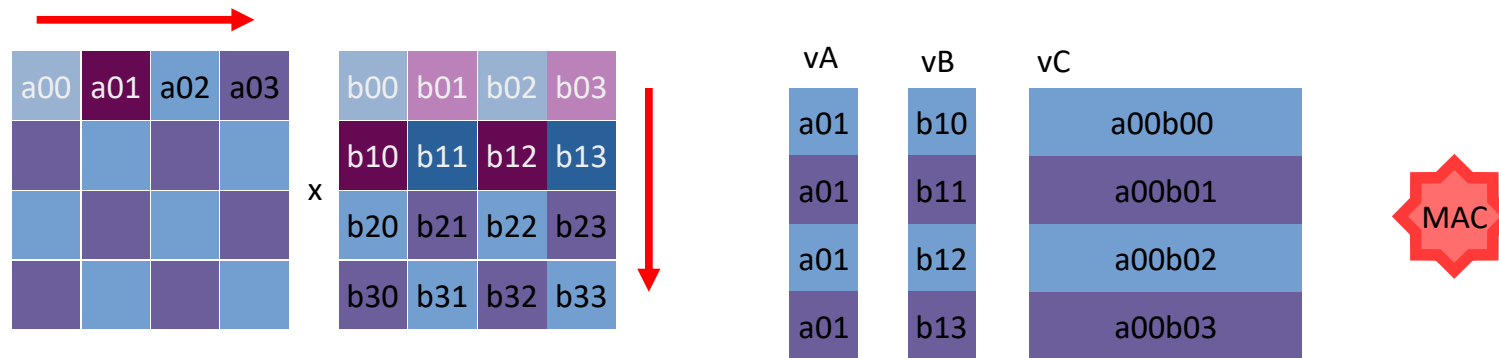
vA	vB	vC
a00	b00	a00b00
a00	b01	a00b01
a00	b02	a00b02
a00	b03	a00b03





# Matrix multiplication on Ara

- Standard algorithm (row times column + reduction) is slow  
Highly sequential
- Use a vector of reductions instead





# Matrix multiplication on Ara

- Load row  $i$  of matrix  $B$  into  $vB$
- for (int  $j = 0$ ;  $j < n$ ;  $j++$ )  
Load element  $A[j, i]$

Broadcast it into  $vA$

$vC \leftarrow vA \cdot vB + vC$

`vld vB, 0(addrB)`

*(Unrolled Loop)*

`ld t0, 0(addrA)`

`addi addrA, addrA, 8`

`vins vA, t0, zero`

**`vmadd vC, vA, vB, vC`**

`ld t0, 0(addrA)`

`addi addrA, addrA, 8`

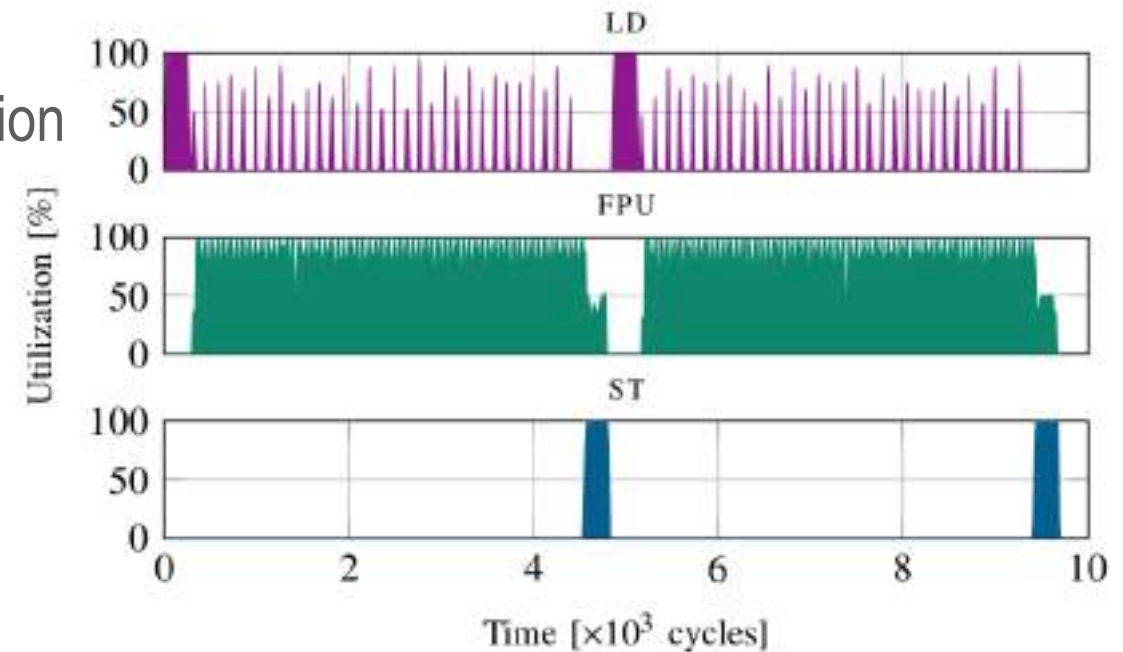
`vins vA, t0, zero`

**`vmadd vC, vA, vB, vC`**



# Matrix Multiplication on ARA

- DP-MATMUL  
n x n double-precision matrix multiplication  
 $C \leftarrow A \cdot B + C$
- $32n^2$  bytes of memory transfers  
and  $2n^3$  operations  
n/16 DP-FLOP/B  
Compute-bound in Ara for  $n > 8$

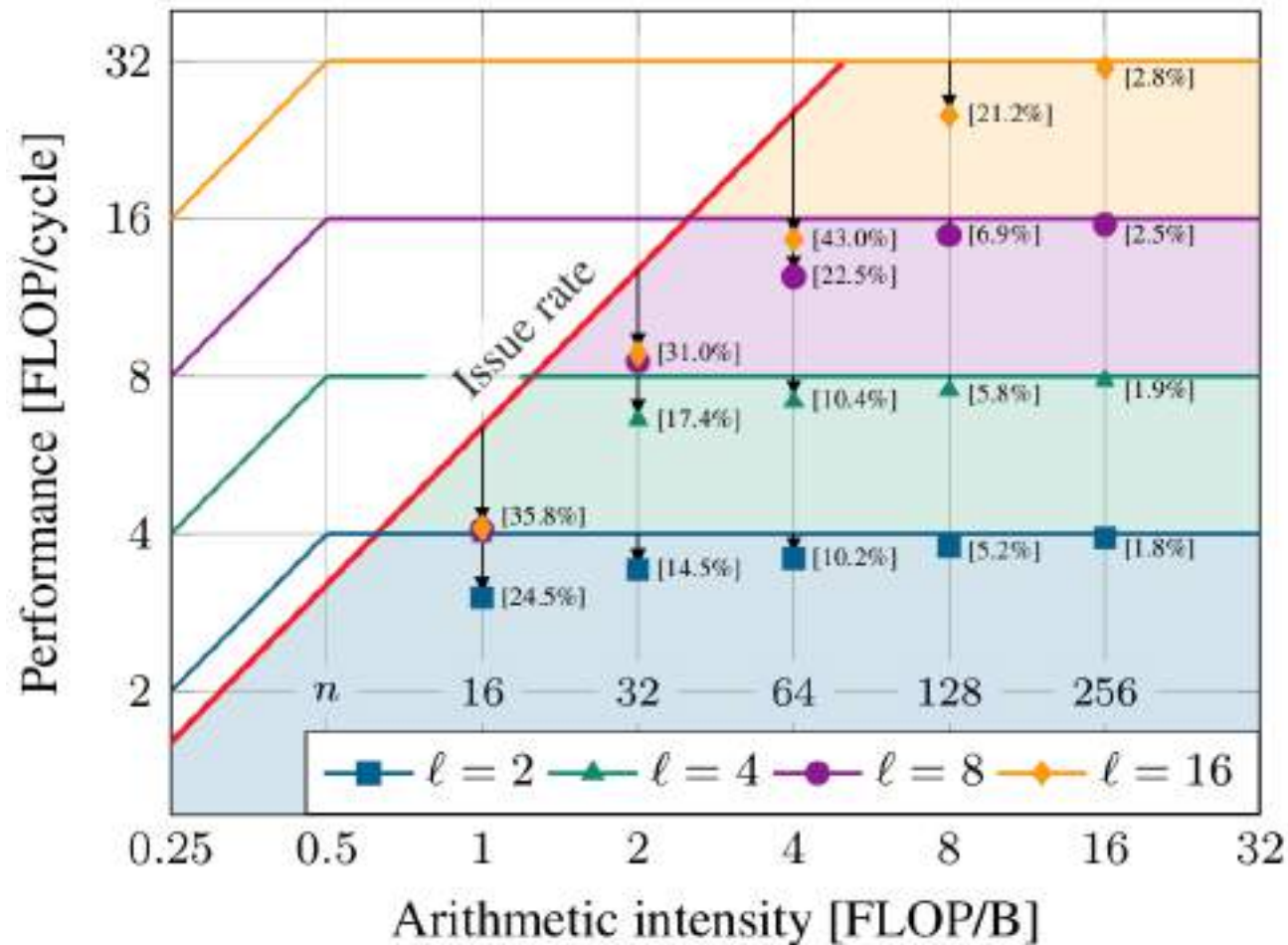


Functional unit's utilization for a 16x16 DP-MATMUL



# Issue rate performance limitation

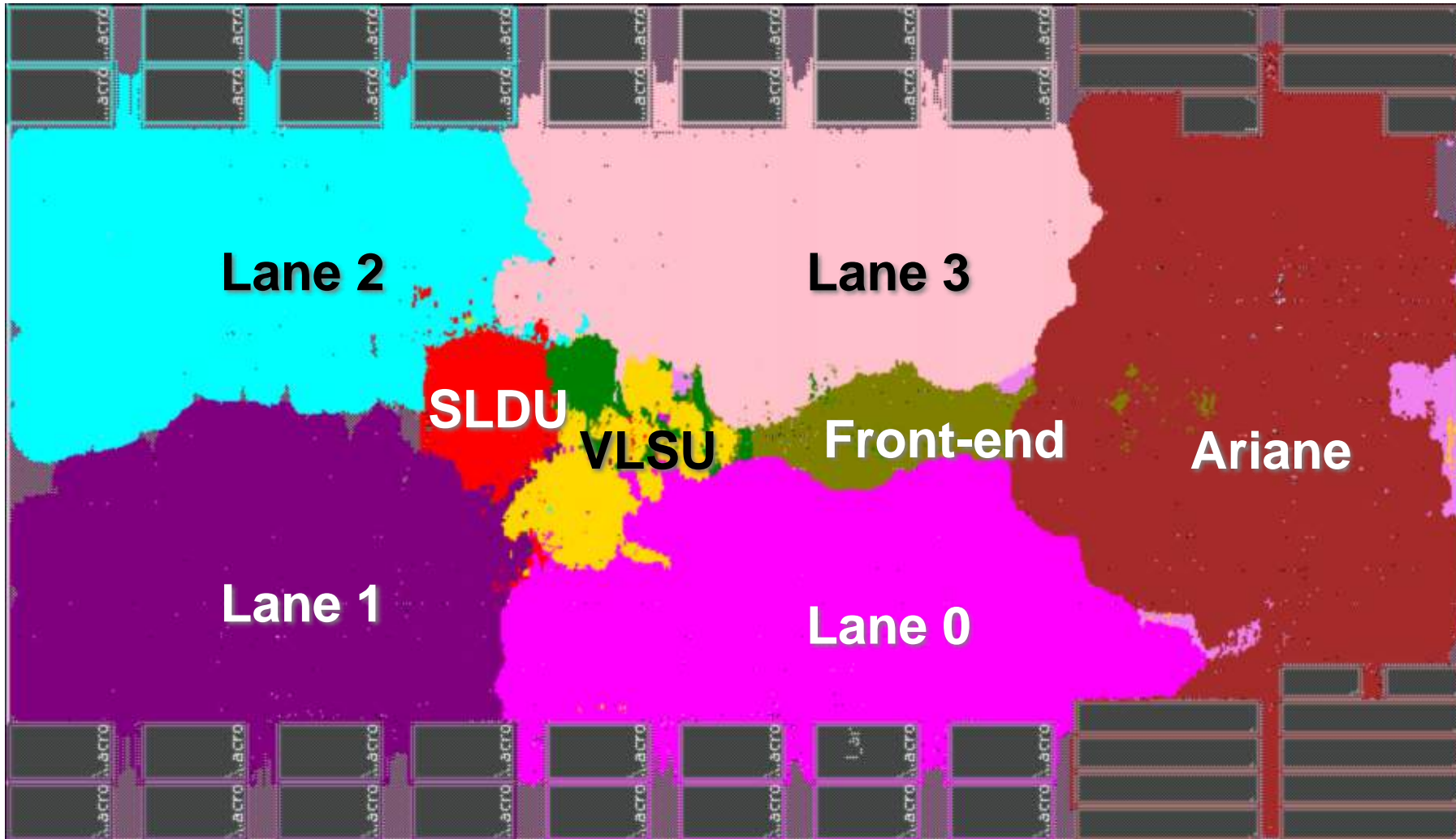
- `vmadds` are issued at best every four cycles  
Since Ariane is single-issue
- If the vector MACs take less than four cycles to execute, the FPUs starve waiting for instructions  
Von Neumann Bottleneck
- This translates to a boundary in the roofline plot





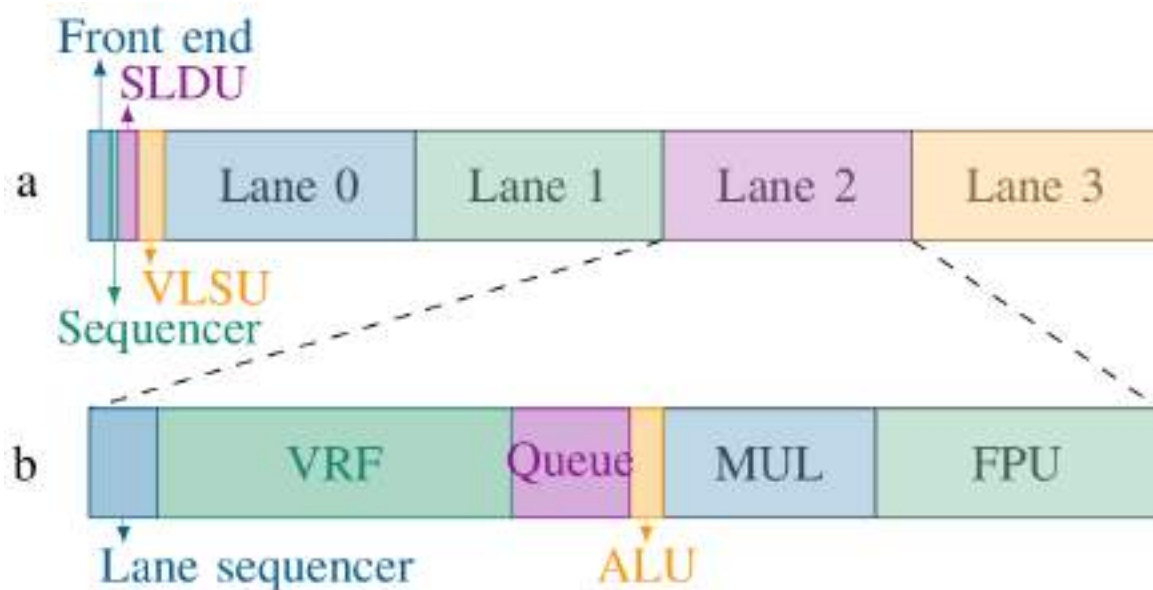
# Ara: 4 lanes GF 22FDX 1.25 GHz implementation

(TT, 0.80V, 25°C)



# Ara: Figures of Merit

## Area breakdown

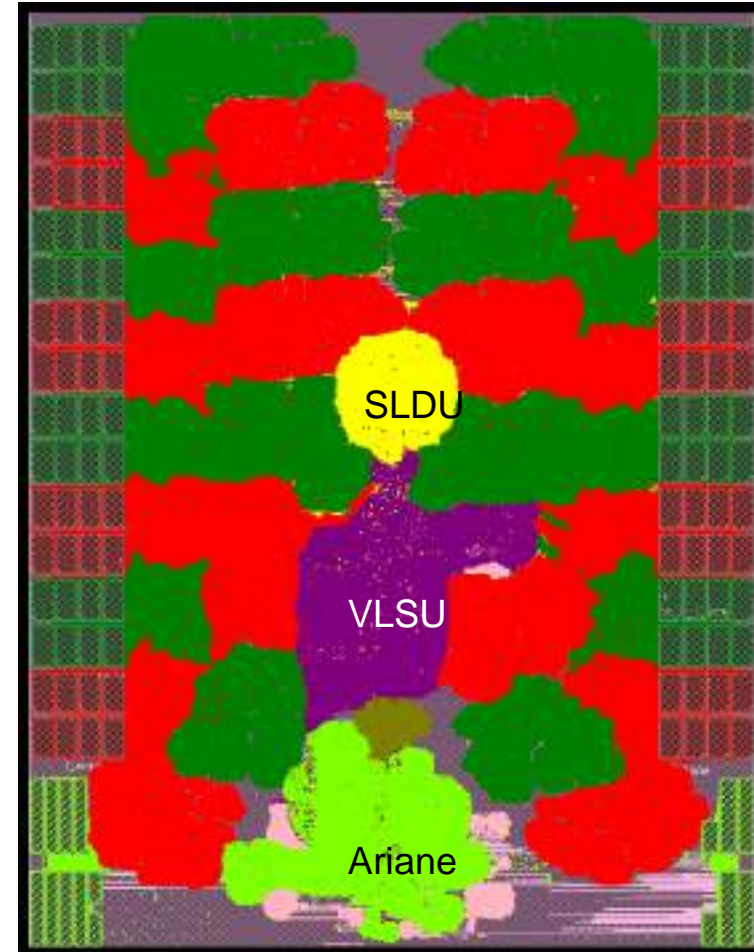


- **Clock frequency**  
1.25 GHz (nominal),  
0.92 GHz (worst condition)
- **Area:**  
3400 kGE, 0.68 mm<sup>2</sup>
- 256 x 256 MATMUL  
Performance: 9.8 DP-GFLOPS  
Power: 259 mW  
Efficiency: 38 DP-GFLOPS/W  
~**2.5x** better than Ariane on same benchmark



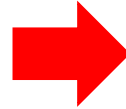
# Ara: Scalability

- Each lane is almost independent  
Contains part of the VRF and its functional units
- Scalability limitations  
VLSU and SLDU: need to communicate to all banks
- Instance with 16 lanes:  
1.04 GHz (nom.), 0.78 GHz (worst case)  
10.7 MGE (2.13mm<sup>2</sup> in GF22)  
32.4 DP-GFLOPS  
40.8 DP-GFLOPS/W (peak)



**16 ARAs give you 1TFLOP at 12W - NOT BAD!**

# HPC Vertical: The European Processor Initiative



## Europe Needs its own Processors

- Processors now control almost every aspect of our lives
- Security (back doors etc.)
- Possible future restrictions on exports to EU due to increasing protectionism
- A competitive EU supply chain for HPC technologies will create jobs and growth in Europe
- Sovereignty (data, economical, embargo)
- High Performance General Purpose Processor for HPC
- **High-performance RISC-V based accelerator**
- Computing platform for autonomous cars
- Will also target the AI, Big Data and other markets in order to be economically sustainable





# PULP

Parallel Ultra Low Power

Luca Benini, Davide Rossi, Andrea Borghesi, Michele Magno, Simone Benatti, Francesco Conti, Francesco Beneventi, Daniele Palossi, Giuseppe Tagliavini, Antonio Pullini, Germain Haugou, Manuele Rusci, Florian Glaser, Fabio Montagna, Bjoern Forsberg, Pasquale Davide Schiavone, Alfio Di Mauro, Victor Javier Kartsch Morinigo, Tommaso Polonelli, Fabian Schuiki, Stefan Mach, Andreas Kurth, Florian Zaruba, Manuel Eggimann, Philipp Mayer, Marco Guermandi, Xiaying Wang, Michael Hersche, Robert Balas, Antonio Mastrandrea, Matheus Cavalcante, Angelo Garofalo, Alessio Burrello, Gianna Paulin, Georg Rutishauser, Andrea Cossettini, Luca Bertaccini, Maxim Mattheeuws, Samuel Riedel, Sergei Vostrikov, Vlad Niculescu, Hanna Mueller, Matteo Perotti, Nils Wistoff, Luca Bertaccini, Thorir Ingulfsson, Thomas Benz, Paul Scheffler, Alessio Burello, Moritz Scherer, Matteo Spallanzani, Andrea Bartolini, Frank K. Gurkaynak, and many more that we forgot to mention



<http://pulp-platform.org>



@pulp\_platform