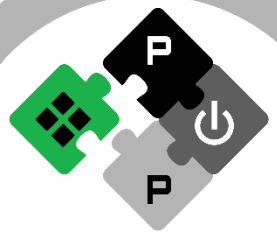


OpenPiton+Ariane Tutorial

HiPEAC 2019, Valencia

21.01.2019

Michael Schaffner, PULP Team [1]

Jonathan Balkind, OpenPiton Team [2]

ETH zürich

[1]*Integrated Systems Laboratory*

PRINCETON UNIVERSITY

[2]*Princeton Parallel Group*

http://pulp-platform.org

http://openpiton.org/

OpenPiton

What's on the menu

- 10:00 – 11:00 Frank K. Gürkaynak – Introduction
  - 11:00 – 11:30 Coffee Break
- 11:30 – 12:00 Andreas Kurth – Software Development Kit
- 12:00 – 13:00 Fabian Schuiki – PULP and CAPI (Hands on demo)
  - 13:00 – 14:00 Lunch
- 14:00 – 15:00 Andreas Kurth – Hero (live demo)
- 15:00 – 15:30 Michael Schaffner – Ariane + OpenPiton (intro)
  - 15:30 – 16:00 Break
- 16:00 – 16:30 Michael Schaffner – Ariane + OpenPiton (live demo)
- 16:30 – 17:30 Q&A, demos

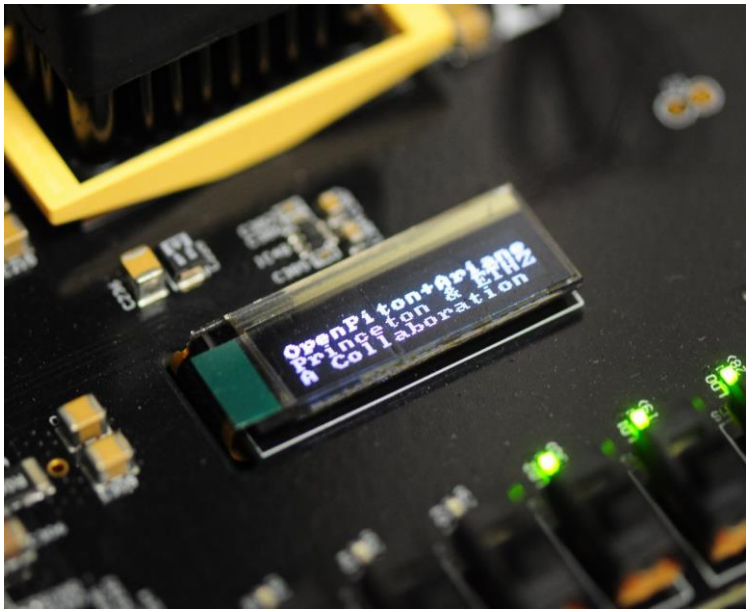
What's on the menu

- **15:00 – 15:30 Ariane + OpenPiton Part 1**
  - Introduction
  - Ariane overview
  - OpenPiton overview
  - OpenPiton+Ariane system
- **16:00 – 16:30 Ariane + OpenPiton Part 2**
  - Walkthrough: from scratch to simulation
  - FPGA Demo (if time permits)

Part 1 – Introduction and Overview

- Collaboration between ETH Zürich and Princeton University
- Goal is to develop a permissively licensed, Linux capable many-core research platform based on RISC-V

- Ariane
  - RV64IMAFDCX Core
  - Linux capable
- OpenPiton
  - Research manycore system
  - OpenSPARC T1 based
  - Coherent NoC, distributed cache



RISC-V cores under development at IIS

32 bit

Low Cost Core

- **Zero-riscy**
  - RV32-ICM
- **Micro-riscy**
  - RV32-CE

Core with DSP enhancements

- **RI5CY**
  - RV32-ICMX
    - SIMD
    - HW loops
    - Bit manipulation
    - Fixed point

Floating-point capable Core

- **RI5CY + FPU**
  - RV32-ICMFX

64 bit

Linux capable Core

- **Ariane**
  - RV64-IMAFDCX
  - Full privileged specification
- **“OS Core”**

OpenPiton

ETH

A new perspective: 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, …)

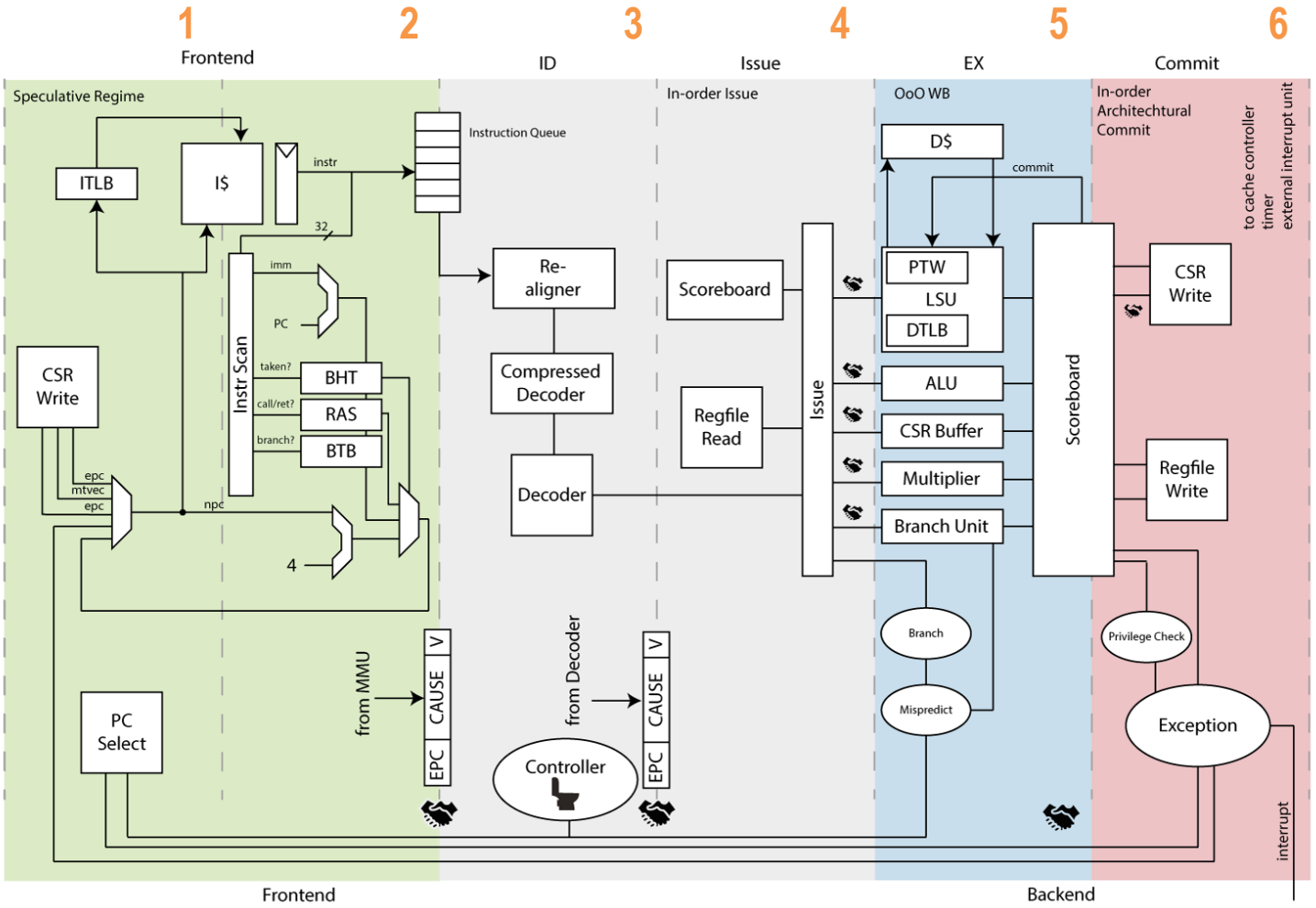
- **Larger address space** (64-bit)
- Requires more hardware support
  - MMU (TLBs, PTW)
  - Privilege Levels
  - More Exceptions (page fault, illegal access)

→ Ariane an application class processor

ARIANE: Linux capable 64-bit core

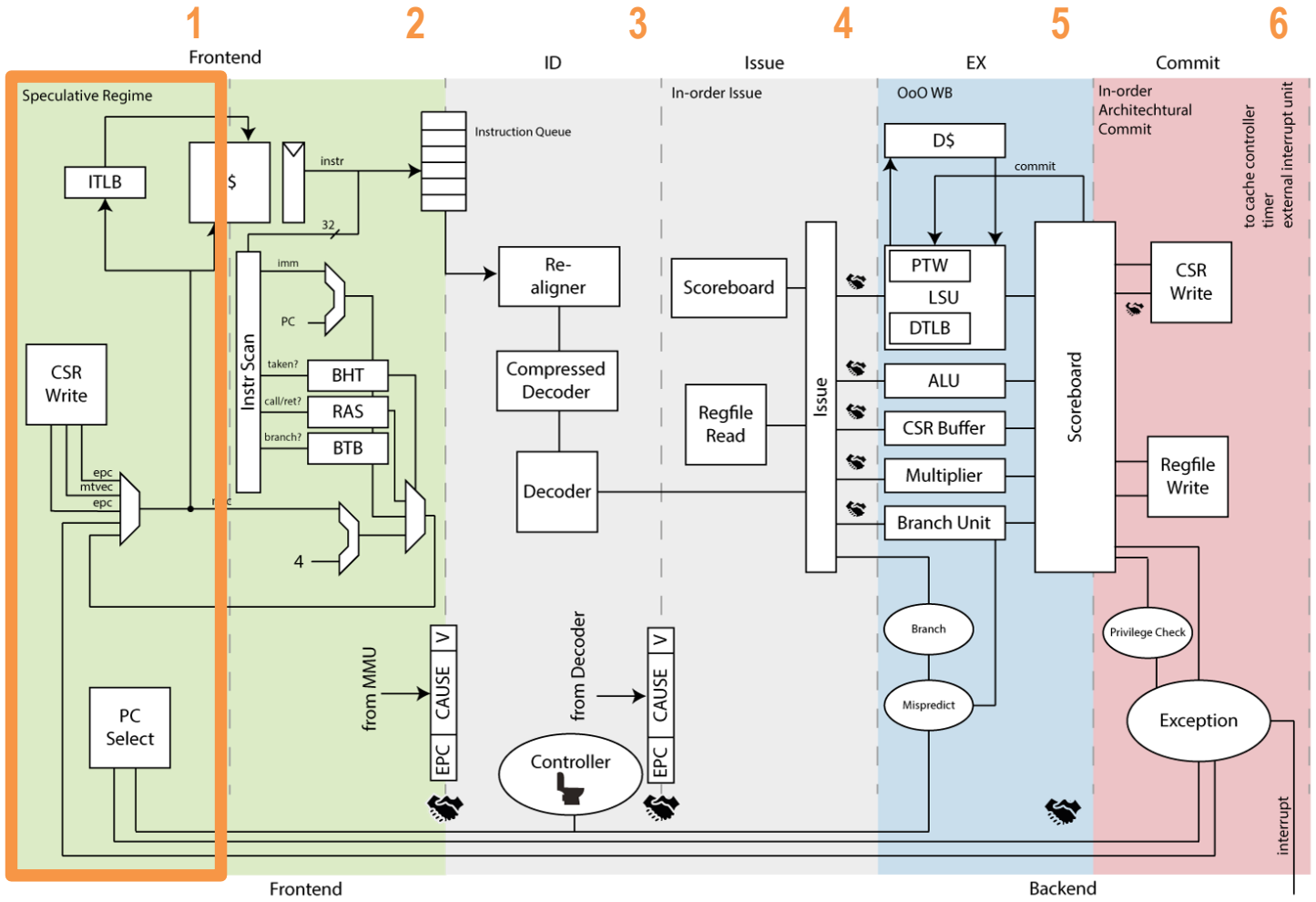
- Application class processor
- Linux Capable
  - Tightly integrated D$ and I$
  - M, S and U privilege modes
  - TLB, SV39
  - Hardware PTW
- Optimized for performance
  - Frequency: 1.5 GHz (22 FDX)
  - Area: ~ 175 kGE
  - Critical path: ~ 25 logic levels
- 6-stage pipeline
  - In-order issue
  - Out-of-order write-back
  - In-order commit
- Scoreboarding
- Designed for extendibility
- Branch-prediction
  - RAS
  - Branch Target Buffer
  - Branch History Table

ARIANE: Linux capable 64-bit core



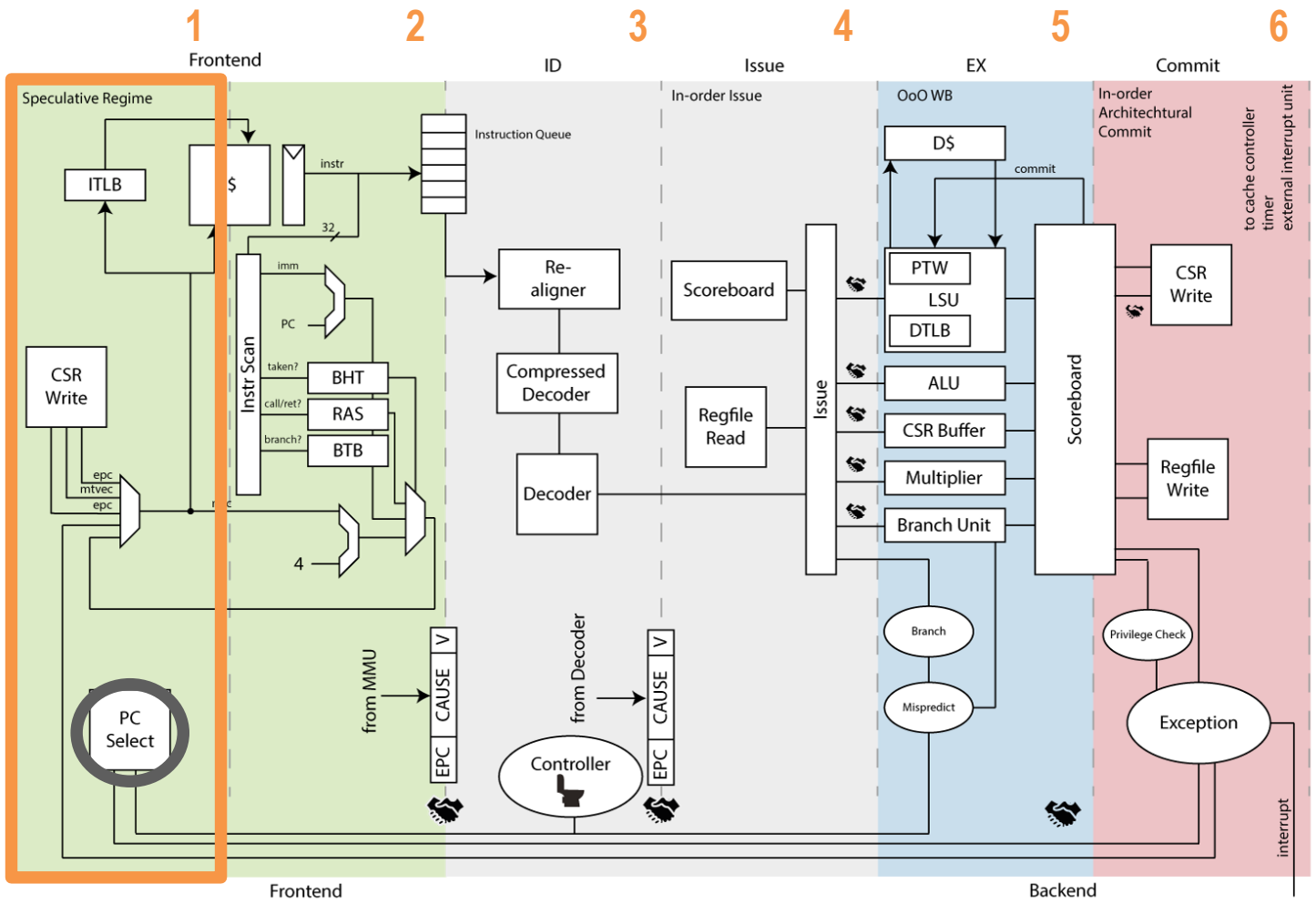
ARIANE: Linux capable 64-bit core

1. PC Gen



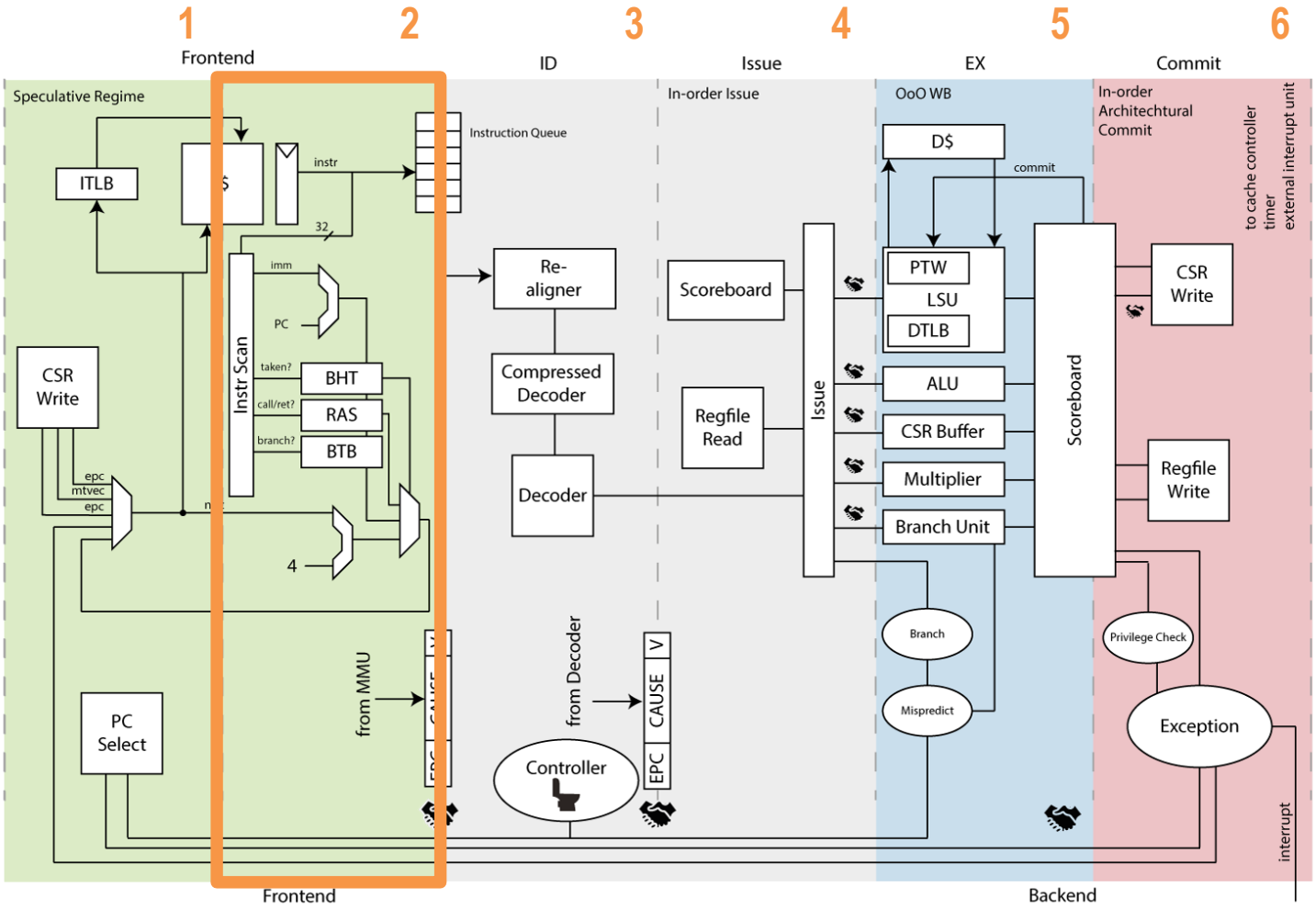
ARIANE: Linux capable 64-bit core

1. **PC Gen**
   - Select PC



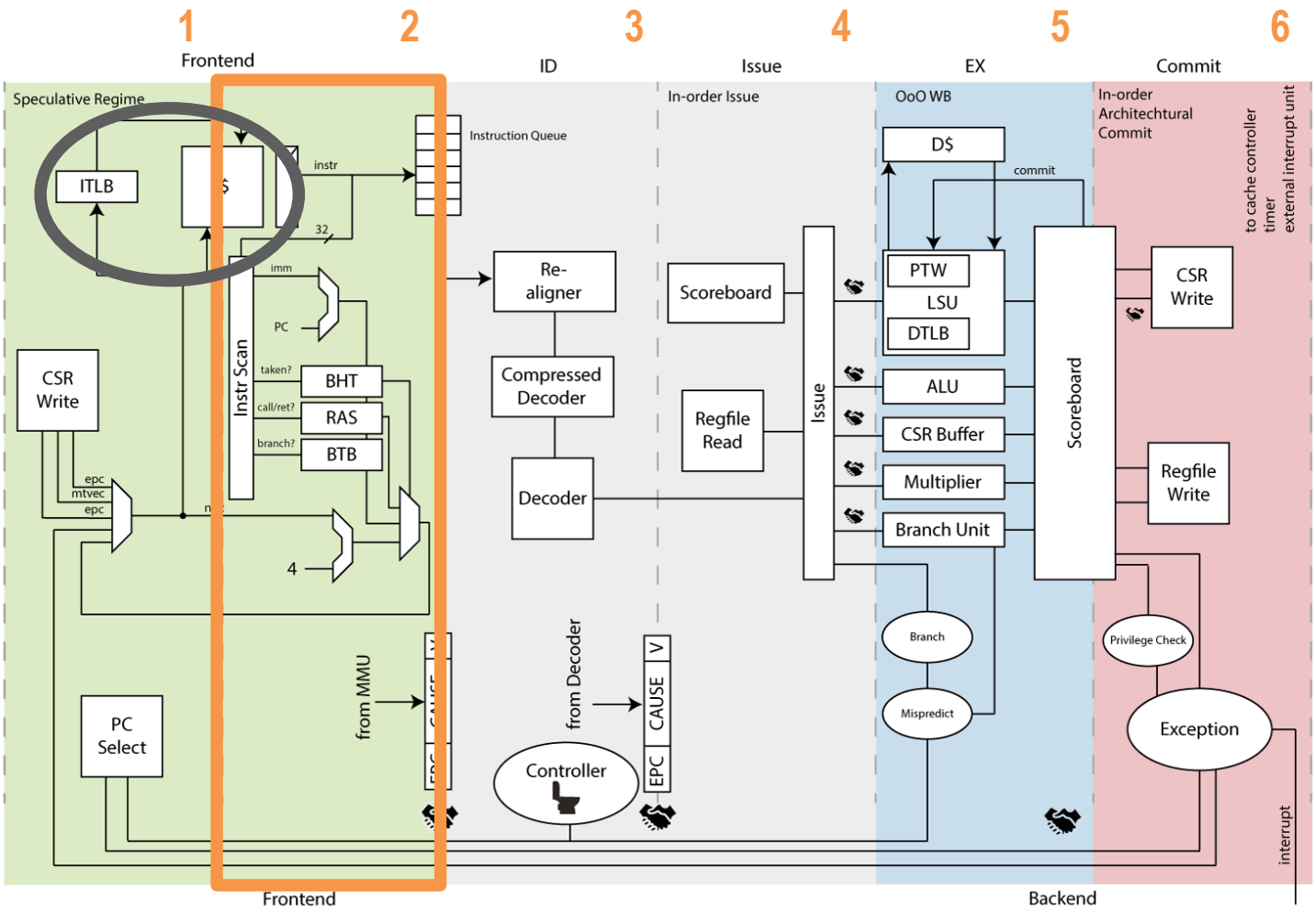
ARIANE: Linux capable 64-bit core

1. PC Gen
   - Select PC
2. Instr. Fetch



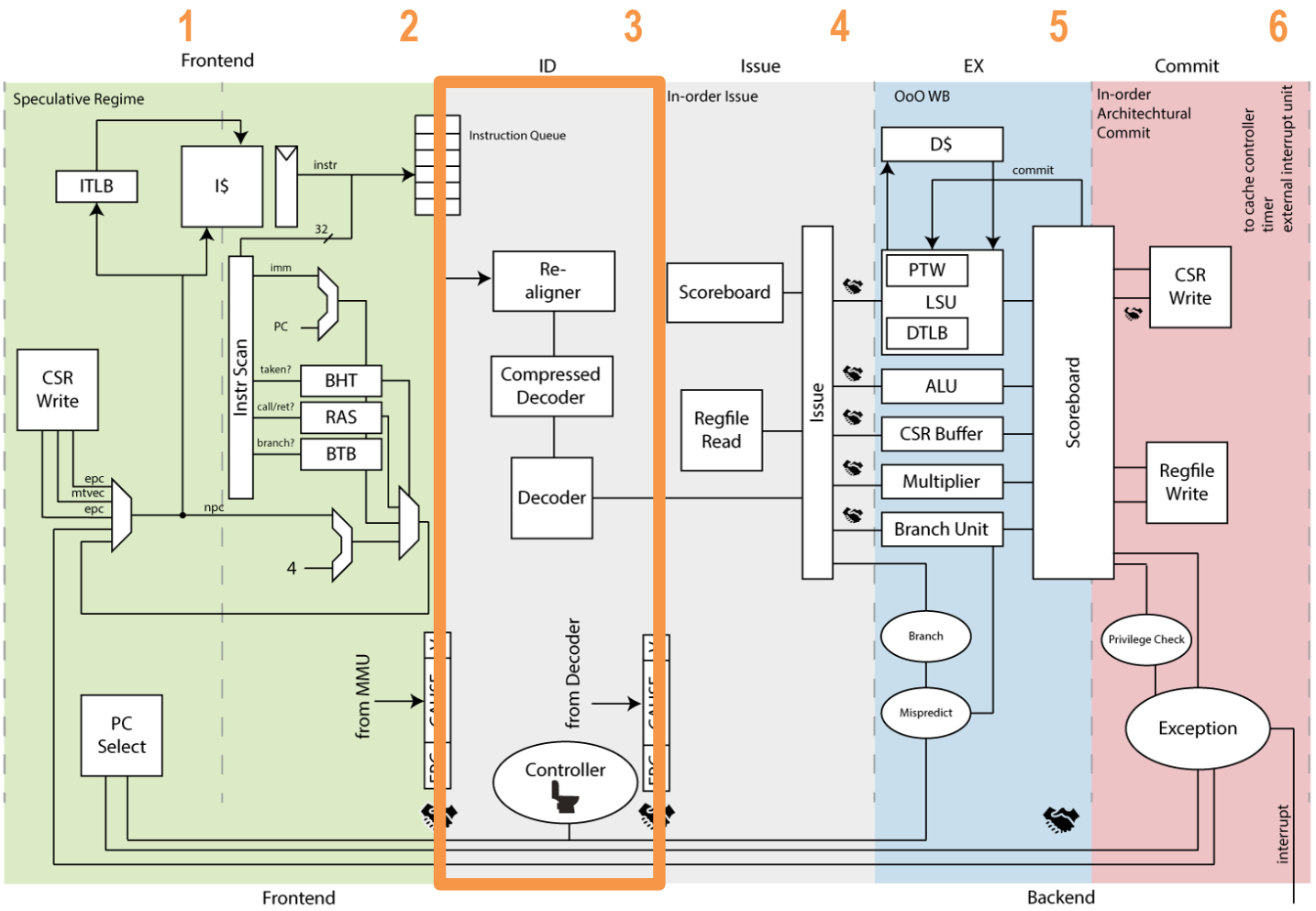
ARIANE: Linux capable 64-bit core

1. PC Gen
   - Select PC
2. Instr. Fetch
   - TLB
   - Query I$



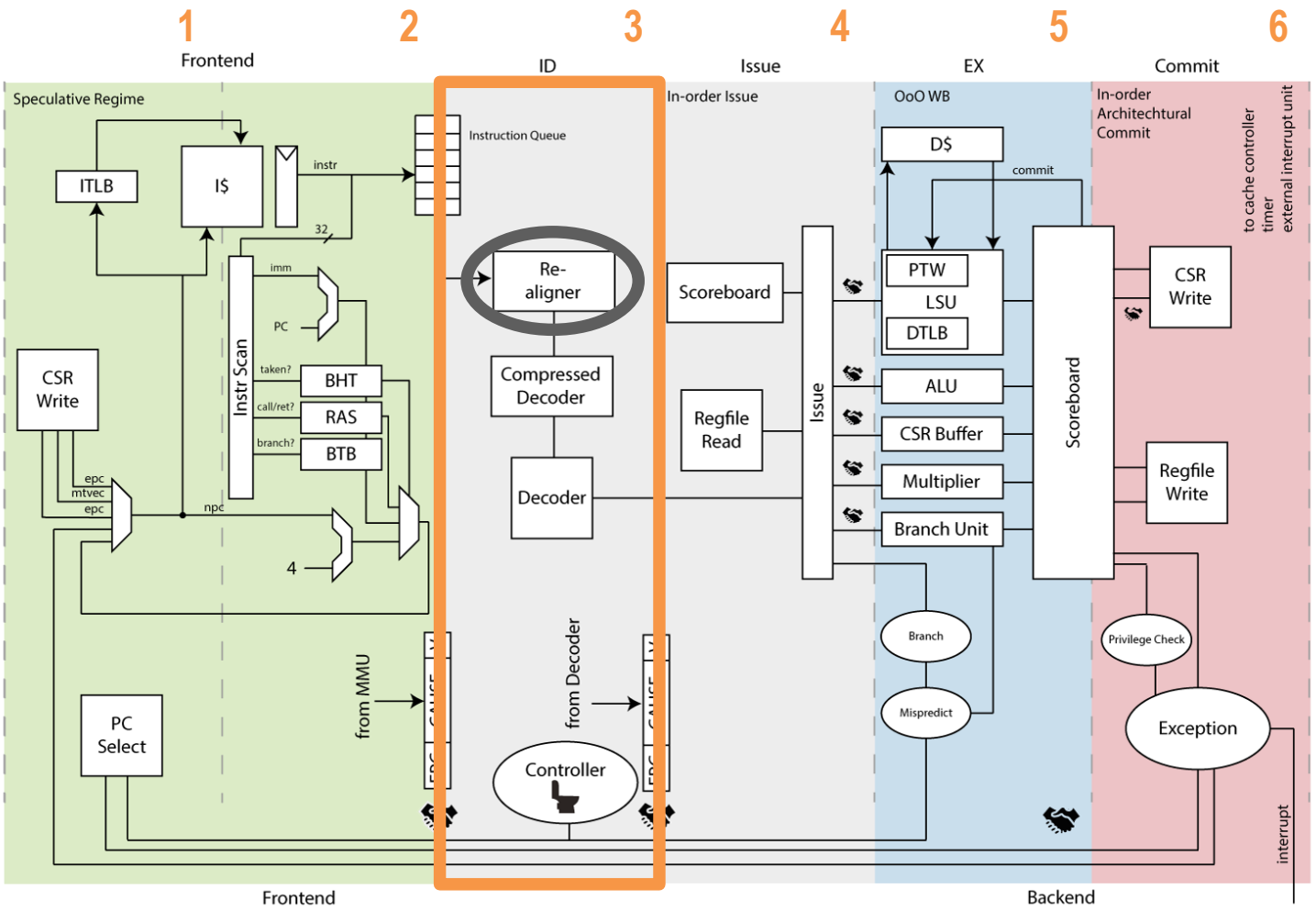
ARIANE: Linux capable 64-bit core

1. PC Gen
   - Select PC
2. Instr. Fetch
   - TLB
   - Query I$
3. Instr. Decode



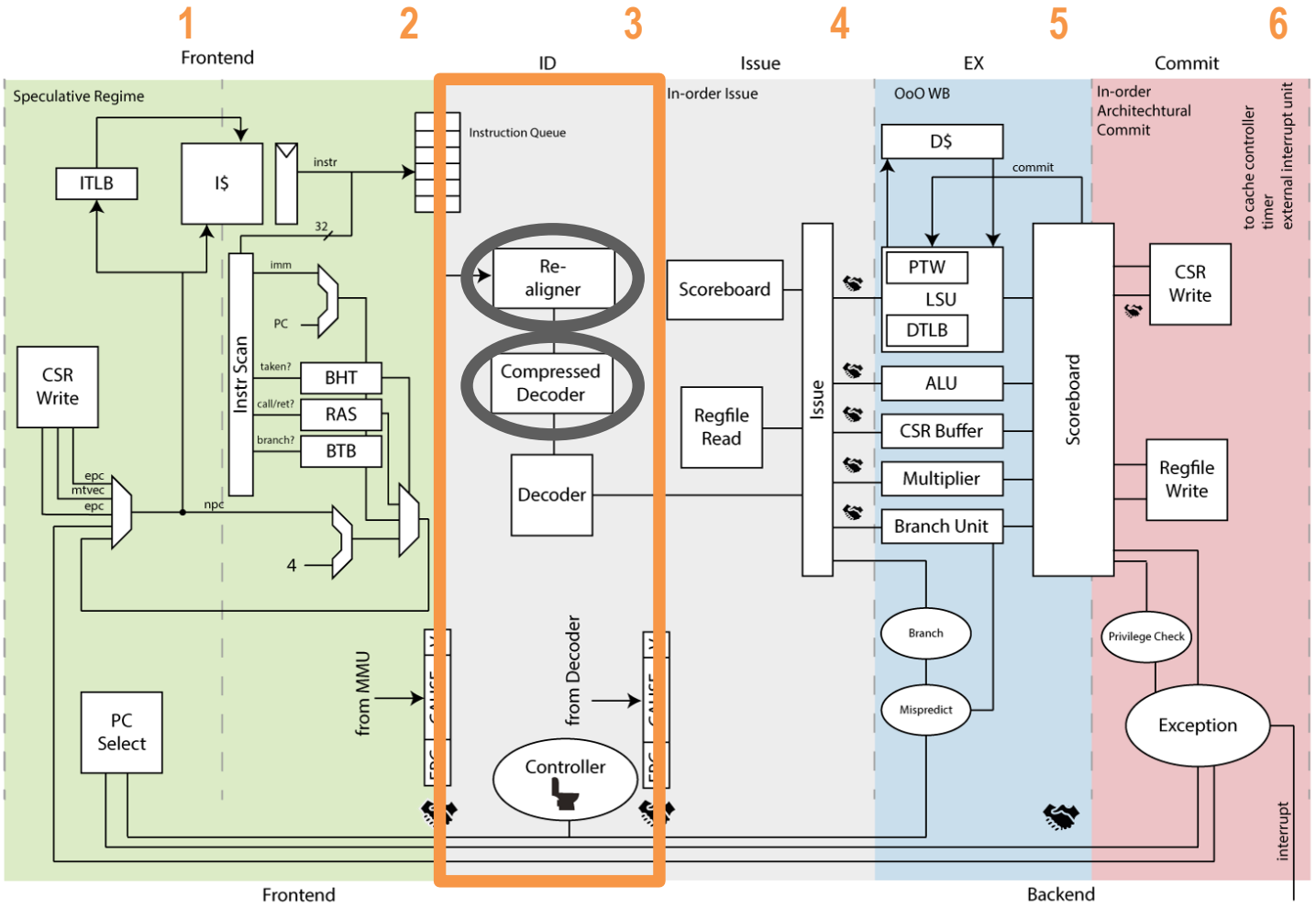
ARIANE: Linux capable 64-bit core

1. **PC Gen**
   - Select PC
2. **Instr. Fetch**
   - TLB
   - Query I$
3. **Instr. Decode**
   - Re-align



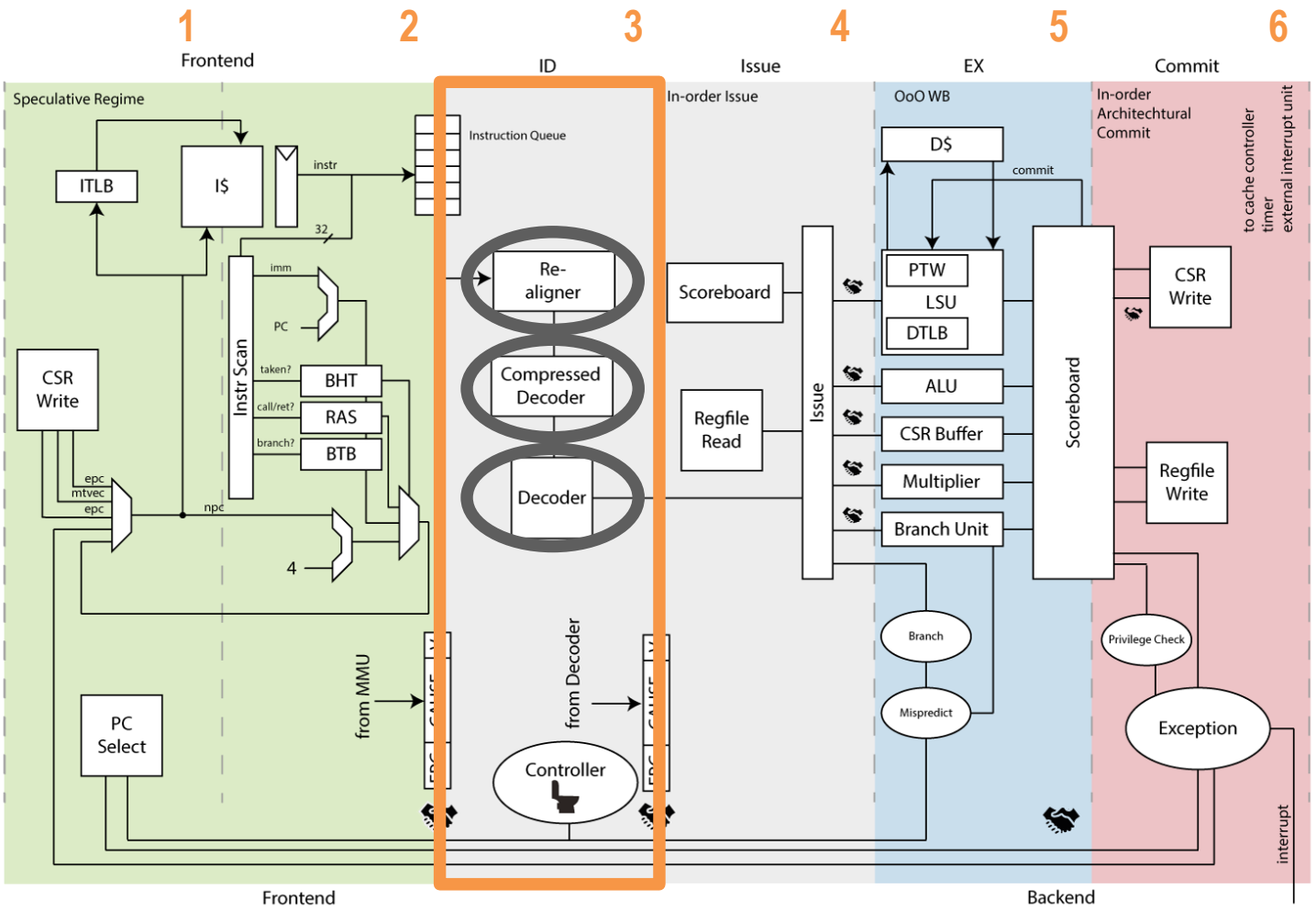
ARIANE: Linux capable 64-bit core

1. **PC Gen**
   - Select PC
2. **Instr. Fetch**
   - TLB
   - Query I$
3. **Instr. Decode**
   - Re-align
   - De-compress



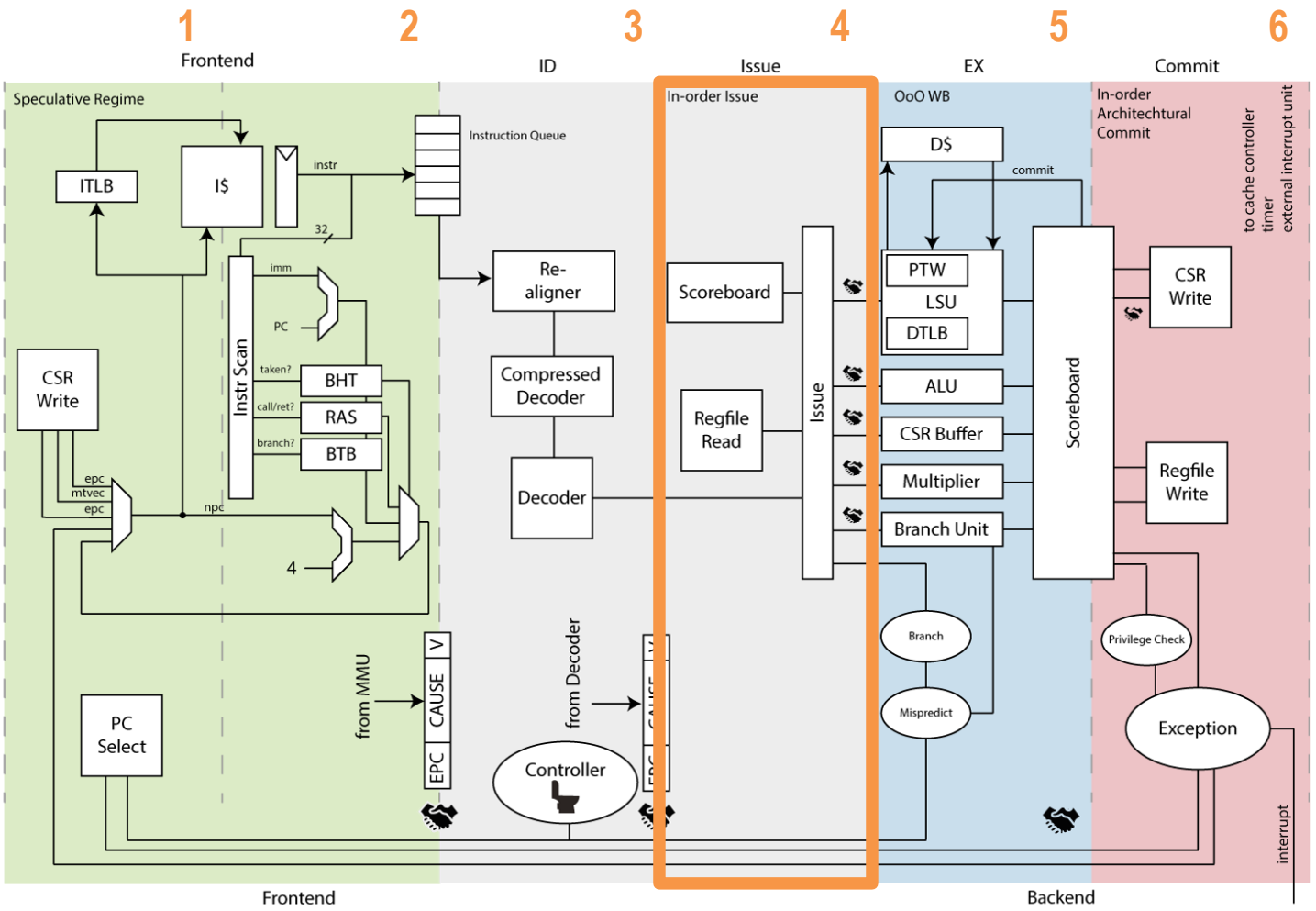
ARIANE: Linux capable 64-bit core

1. **PC Gen**
   - Select PC
2. **Instr. Fetch**
   - TLB
   - Query I$
3. **Instr. Decode**
   - Re-align
   - De-compress
   - Decode



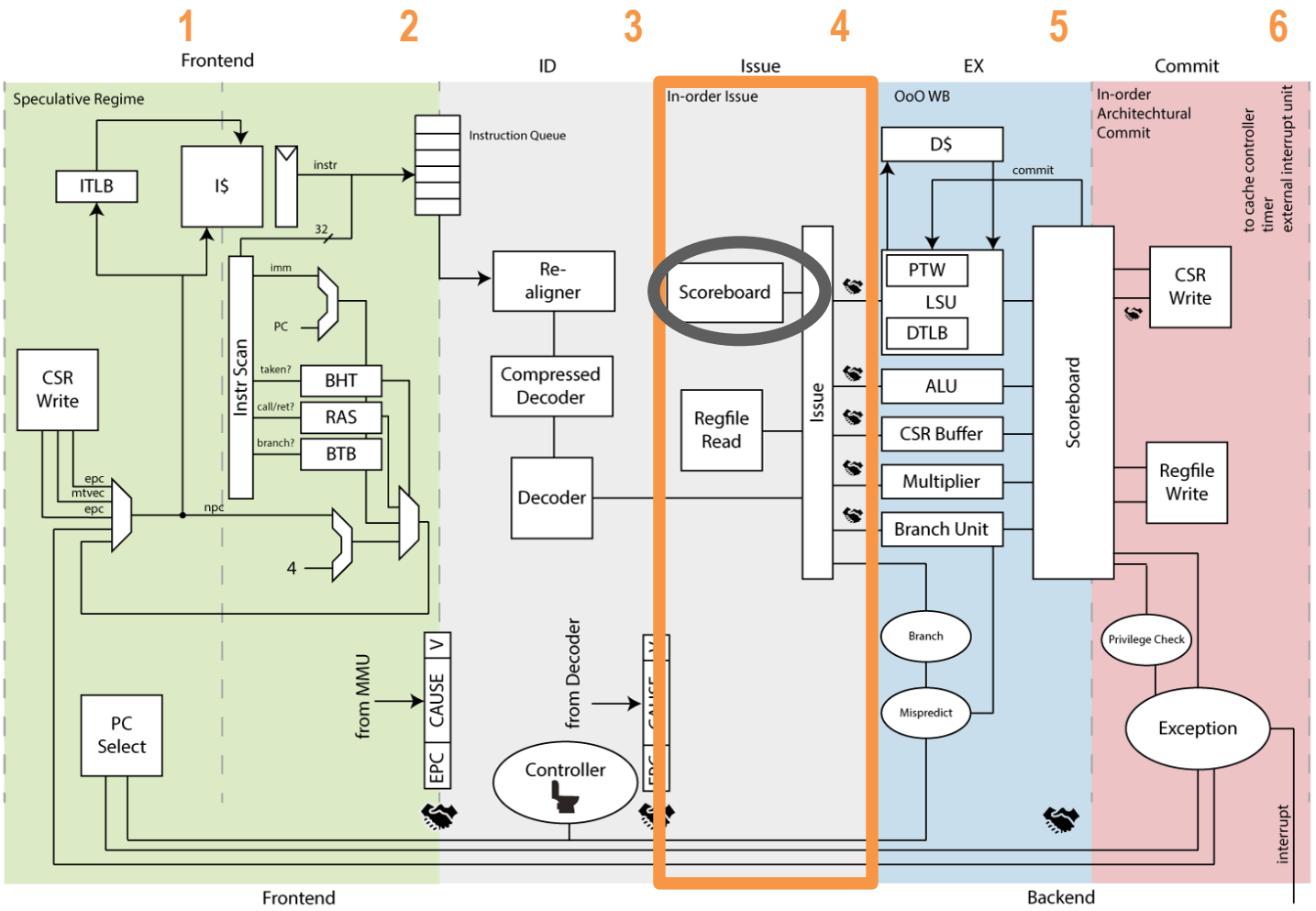
ARIANE: Linux capable 64-bit core

1. PC Gen
   - Select PC
2. Instr. Fetch
   - TLB
   - Query I$
3. Instr. Decode
   - Re-align
   - De-compress
   - Decode
4. Issue



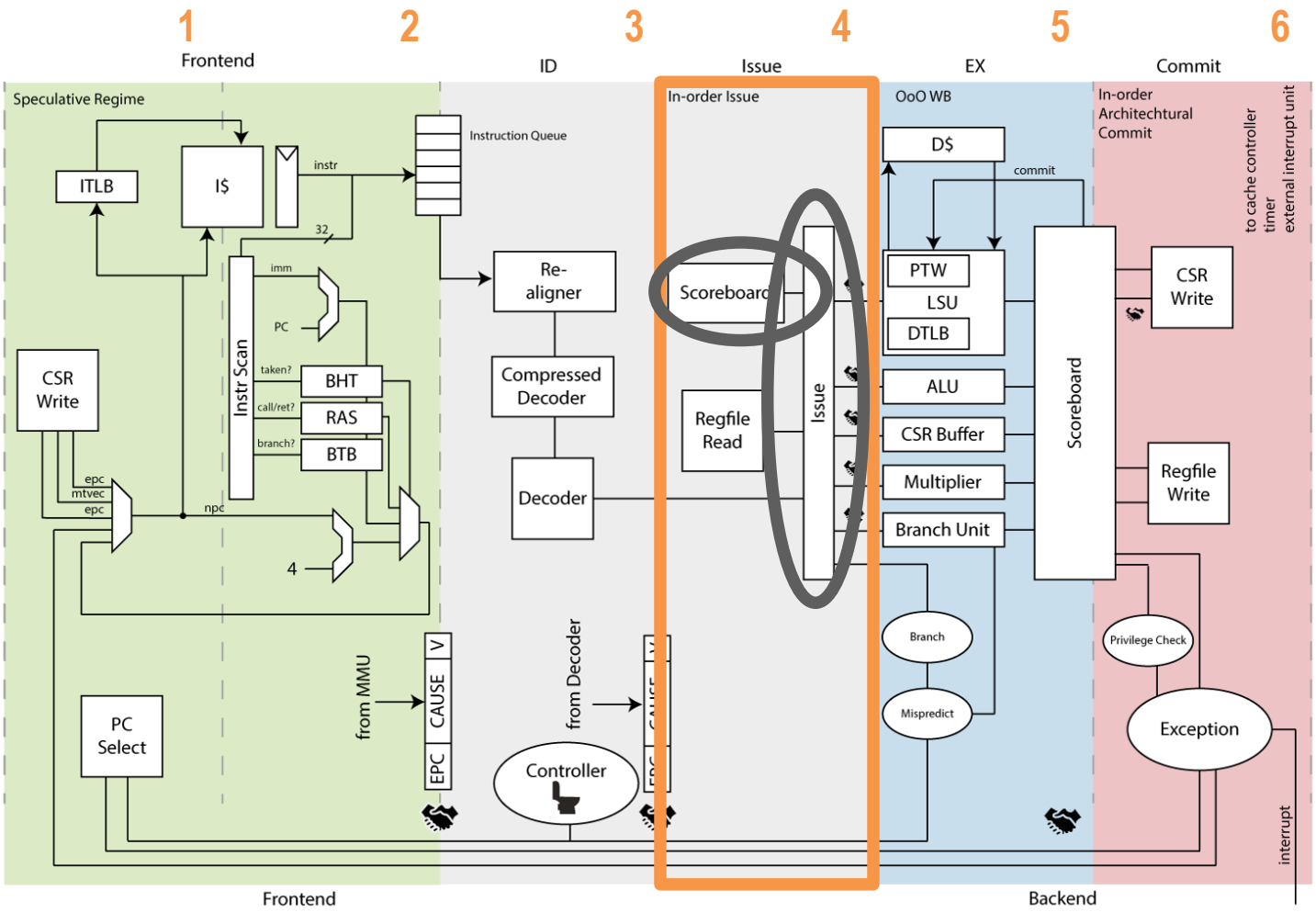
ARIANE: Linux capable 64-bit core

1. **PC Gen**
   - Select PC
2. **Instr. Fetch**
   - TLB
   - Query I$
3. **Instr. Decode**
   - Re-align
   - De-compress
   - Decode
4. **Issue**
   - Select FU



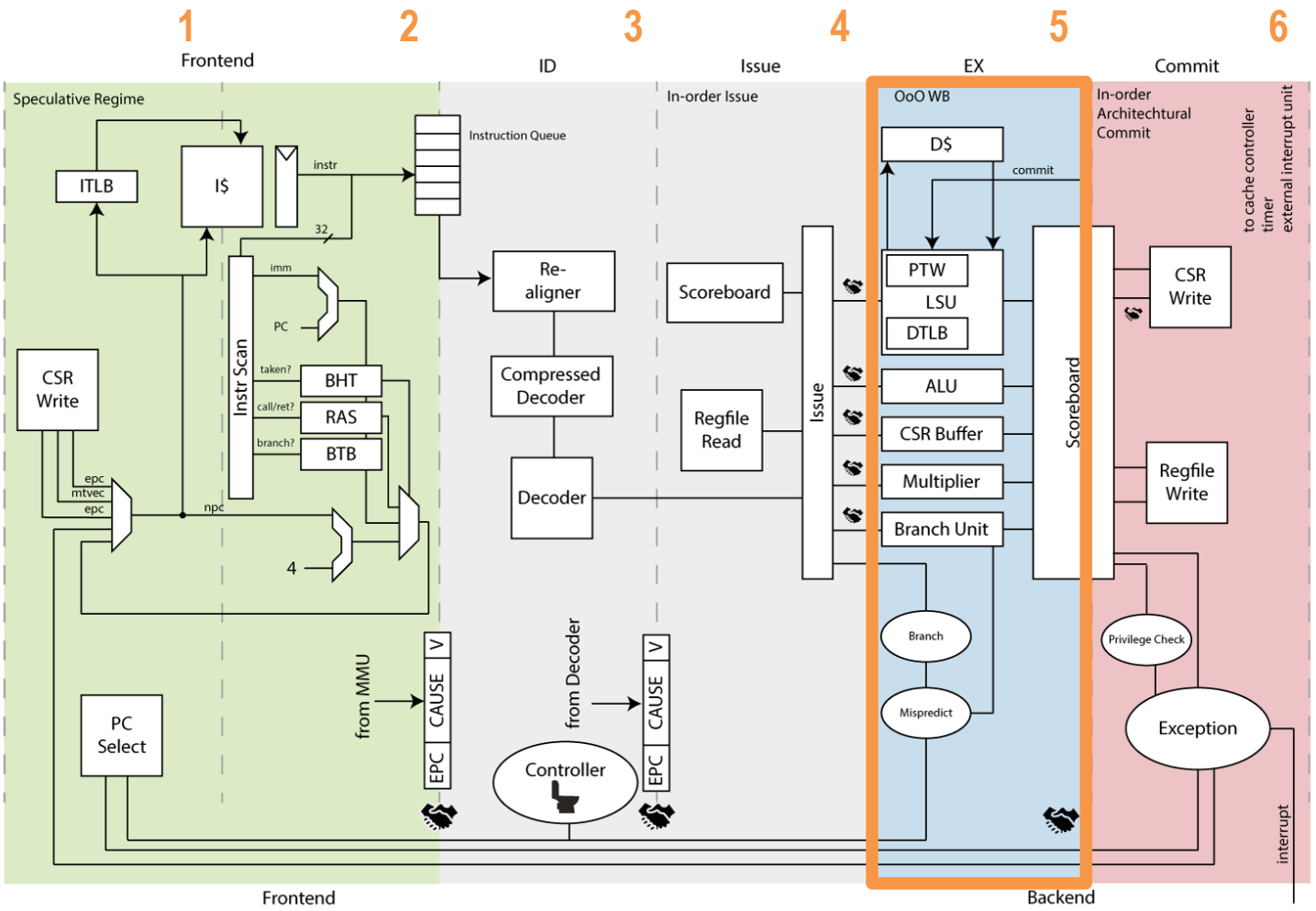
ARIANE: Linux capable 64-bit core

1. **PC Gen**
   - Select PC
2. **Instr. Fetch**
   - TLB
   - Query I$
3. **Instr. Decode**
   - Re-align
   - De-compress
   - Decode
4. **Issue**
   - Select FU
   - Issue



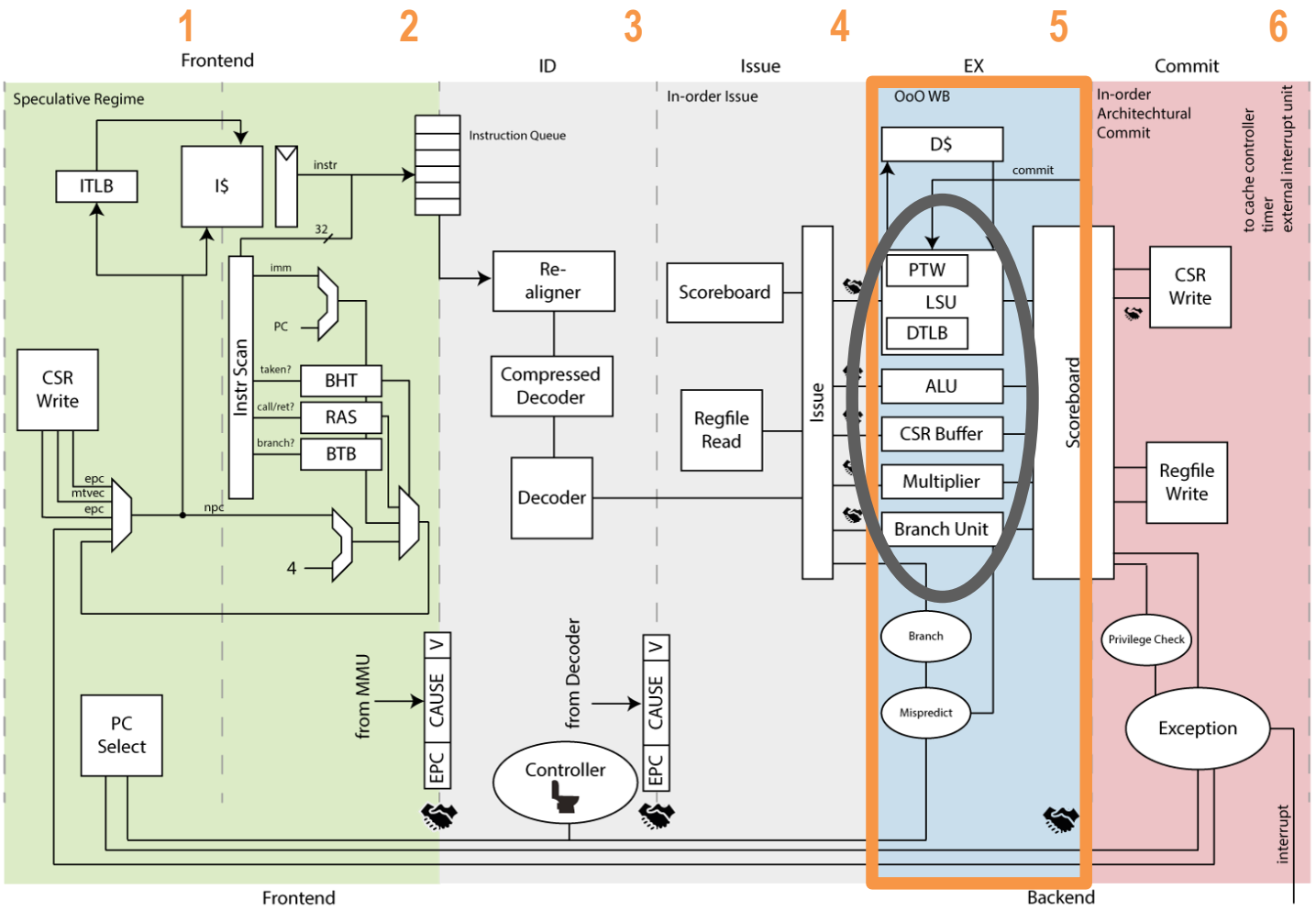
ARIANE: Linux capable 64-bit core

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



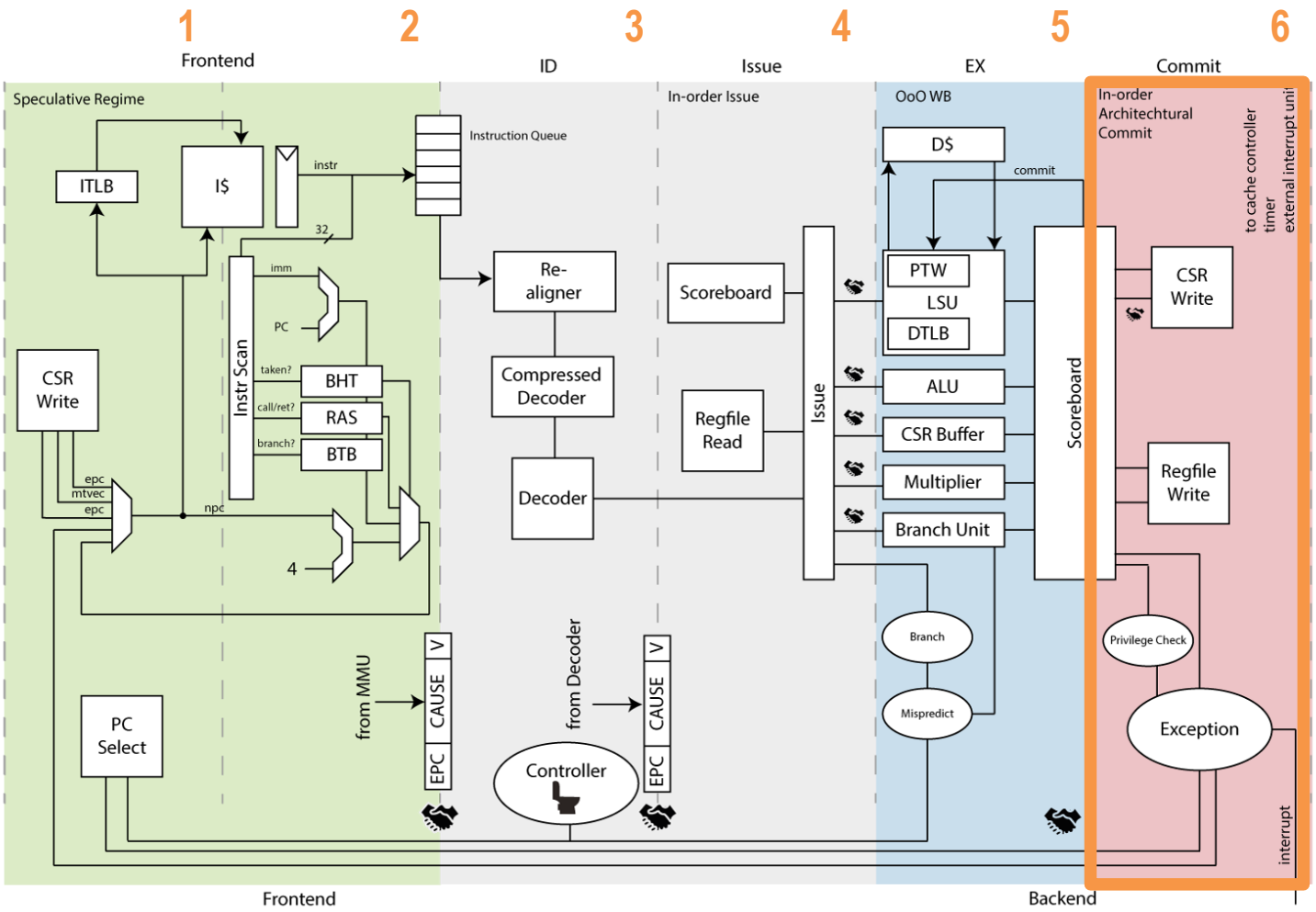
ARIANE: Linux capable 64-bit core

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**
   - FUs



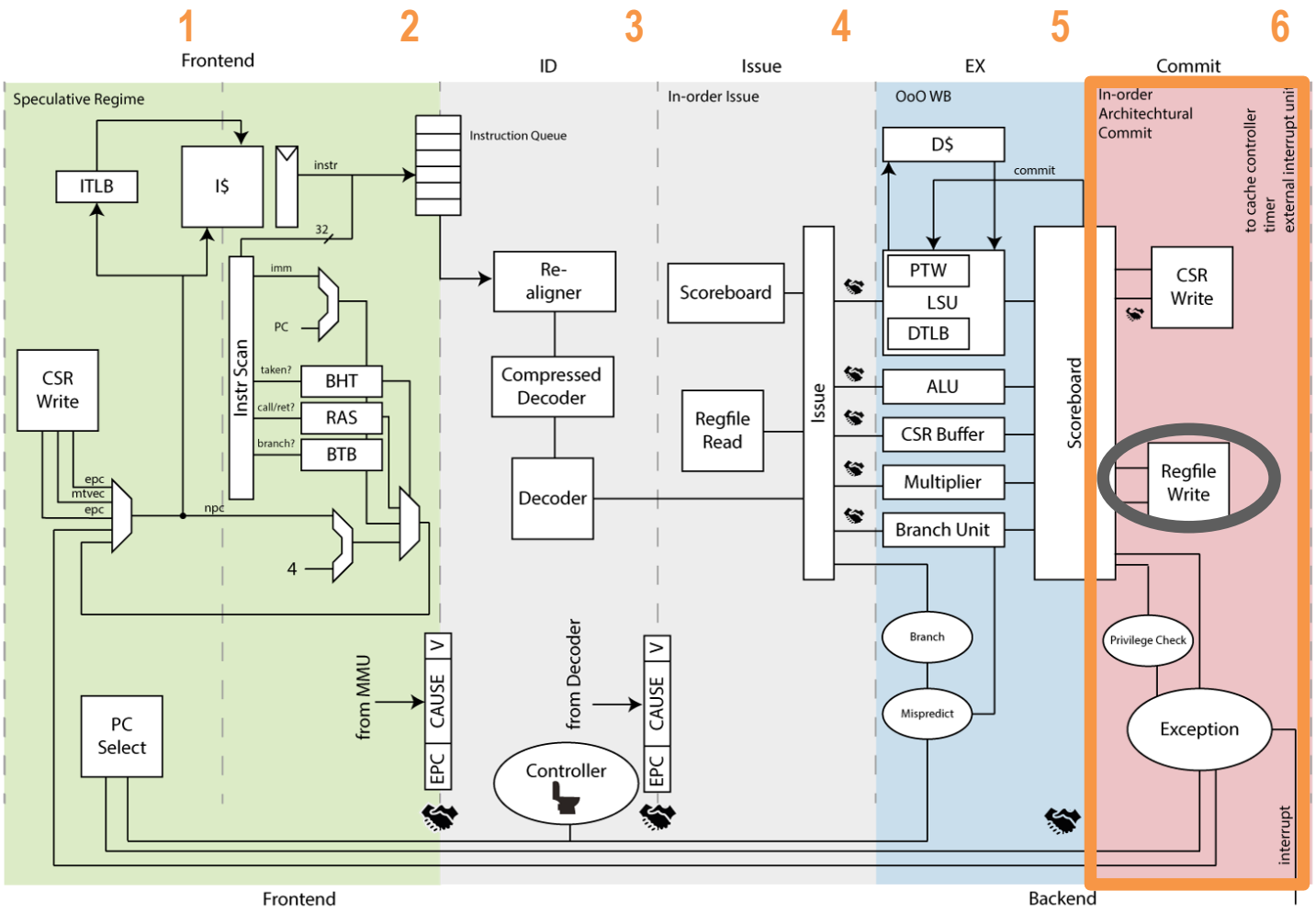
ARIANE: Linux capable 64-bit core

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
   - FUs
6. Commit



ARIANE: Linux capable 64-bit core

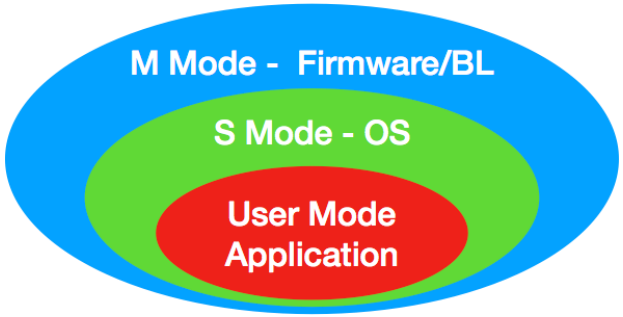
1. **PC Gen**
   - Select PC
2. **Instr. Fetch**
   - TLB
   - Query I$
3. **Instr. Decode**
   - Re-align
   - De-compress
   - Decode
4. **Issue**
   - Select FU
   - Issue
5. **Execute**
   - FUs
6. **Commit**
   - Write state



RISC-V Privilege Modes

- 3 (4) Layer approach
  - Machine Mode (Ring 0)
    - M-Mode CSRS, Interrupts
  - (Hypervisor Mode)
    - Still in draft, efficient support for Type-1 and Type-2 Hypervisors
  - Supervisor Mode
    - MPU/MMU
    - S-Mode CSRs, Interrupts
  - User Mode (U Extension)
    - (Optional) U-Mode CSRs, Interrupts

- Virtual Memory
  - Page-based support
  - 2-4 levels of page table
    - Early-out mechanism enables bigger pages (MB, GB)
  - ASID (application specific identifier) for more efficient TLB flushing
  - Soft-TLB possible but no support yet



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

RISC-V Debug

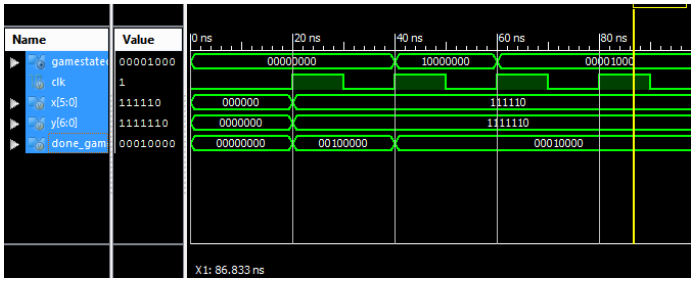
- Draft specification 0.13
  - More or less frozen
- Defines debug registers for
  - run/halt/single-step
  - reading/writing GPR, FPR and CSRs
  - Querying hart status
- JTAG interface
- OpenOCD support
- SiFive influenced

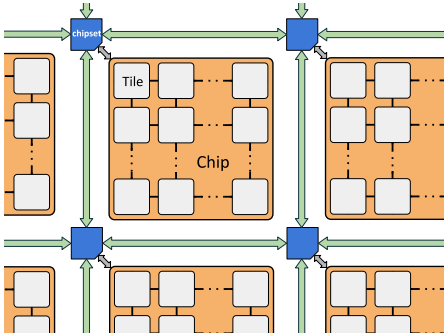
- RI5CY/Ariane contain performance counters
  - SoC performance monitoring not part of RISC-V spec
- Trace task group working on PC tracing
  - UltraSoC leading efforts
  - PULP effectively engaging
  - Working on implementation for PULPissimo

What is OpenPiton ?

- Developed by Princeton Parallel Group (David Wentzlaff)
- Open source (GPL core, BSD uncore) manycore
- Scalable, coherent cache system, based on NoC
- Written in Verilog RTL
- Scales to ½ billion cores
- Configurable core (OpenSPARC T1), uncore
- Includes synthesis and back-end flow
- ASIC & FPGA verified
- ASIC power and energy fully characterized [HPCA 2018]
- Runs full stack multi-user Debian Linux
- Great for Architecture, Programming Language, Compilers, Operating Systems, Security, EDA research

GPL BSD



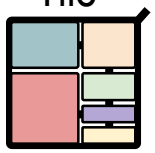


debian

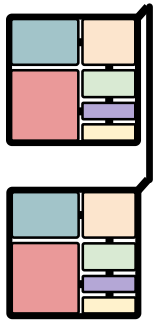
XILINX®

OpenPiton System Overview

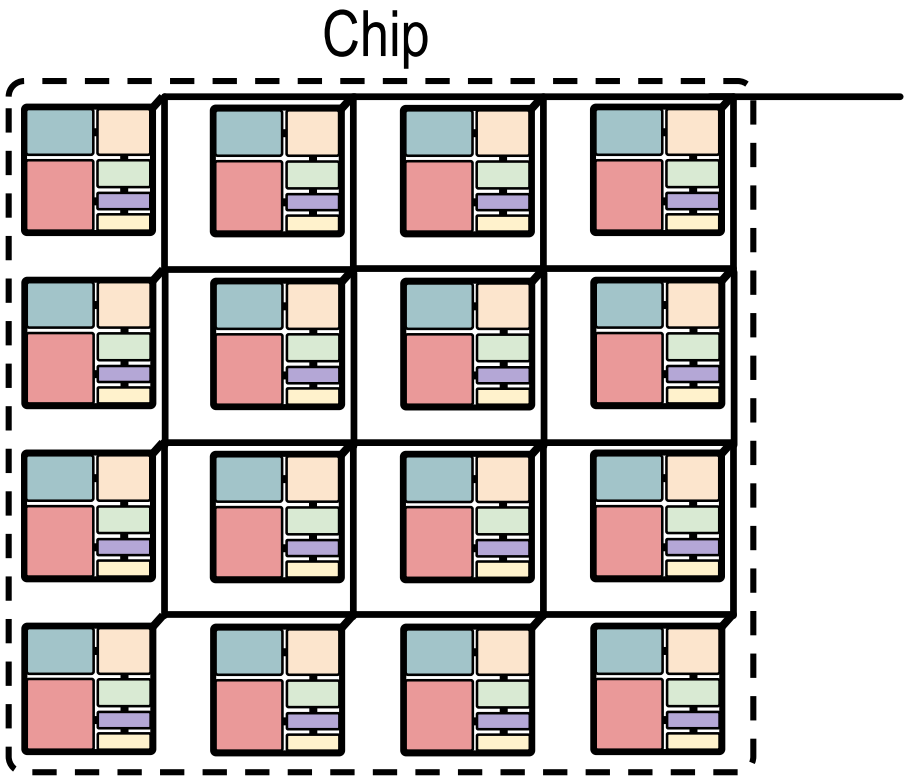
Tile



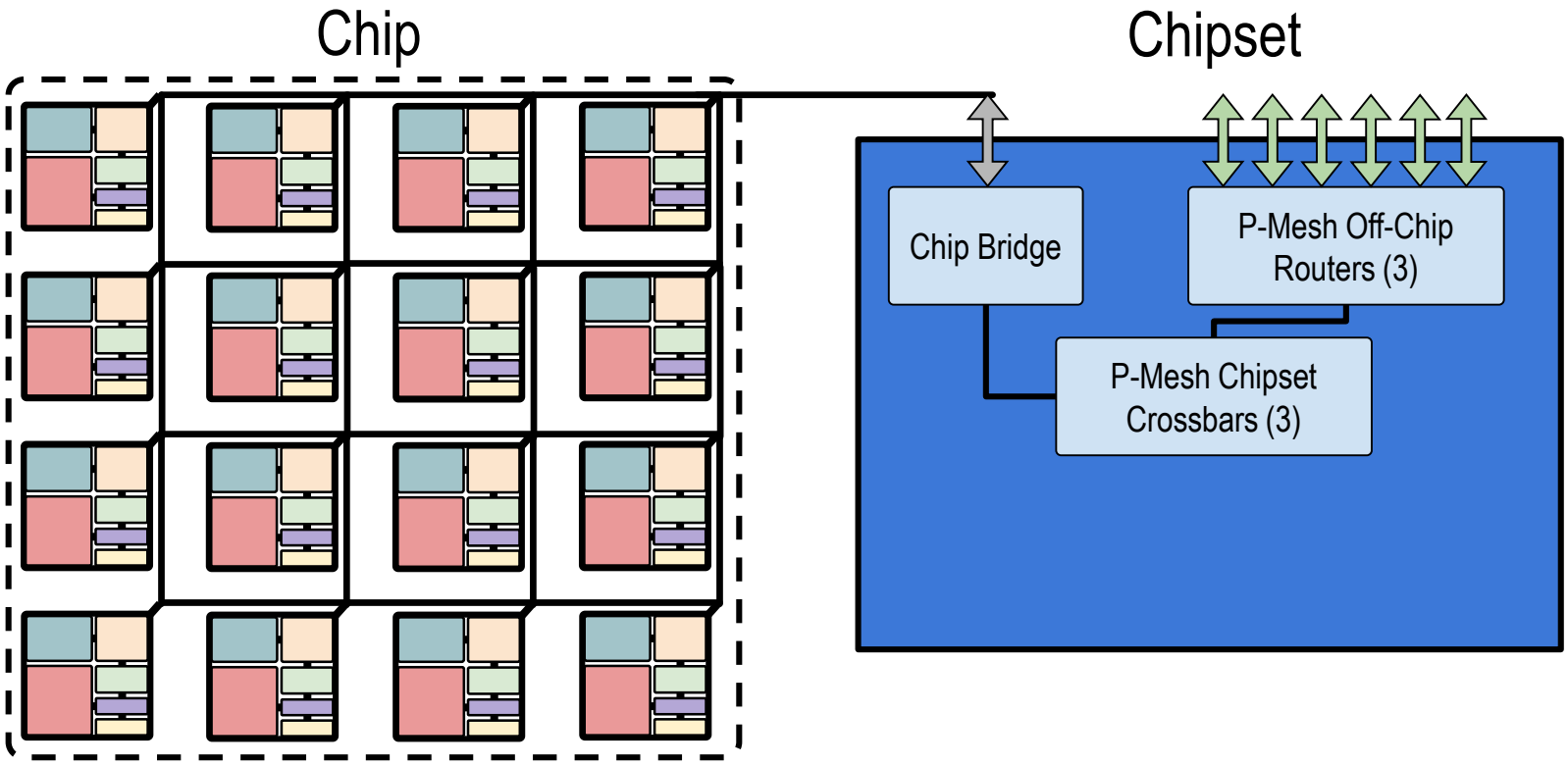
OpenPiton System Overview



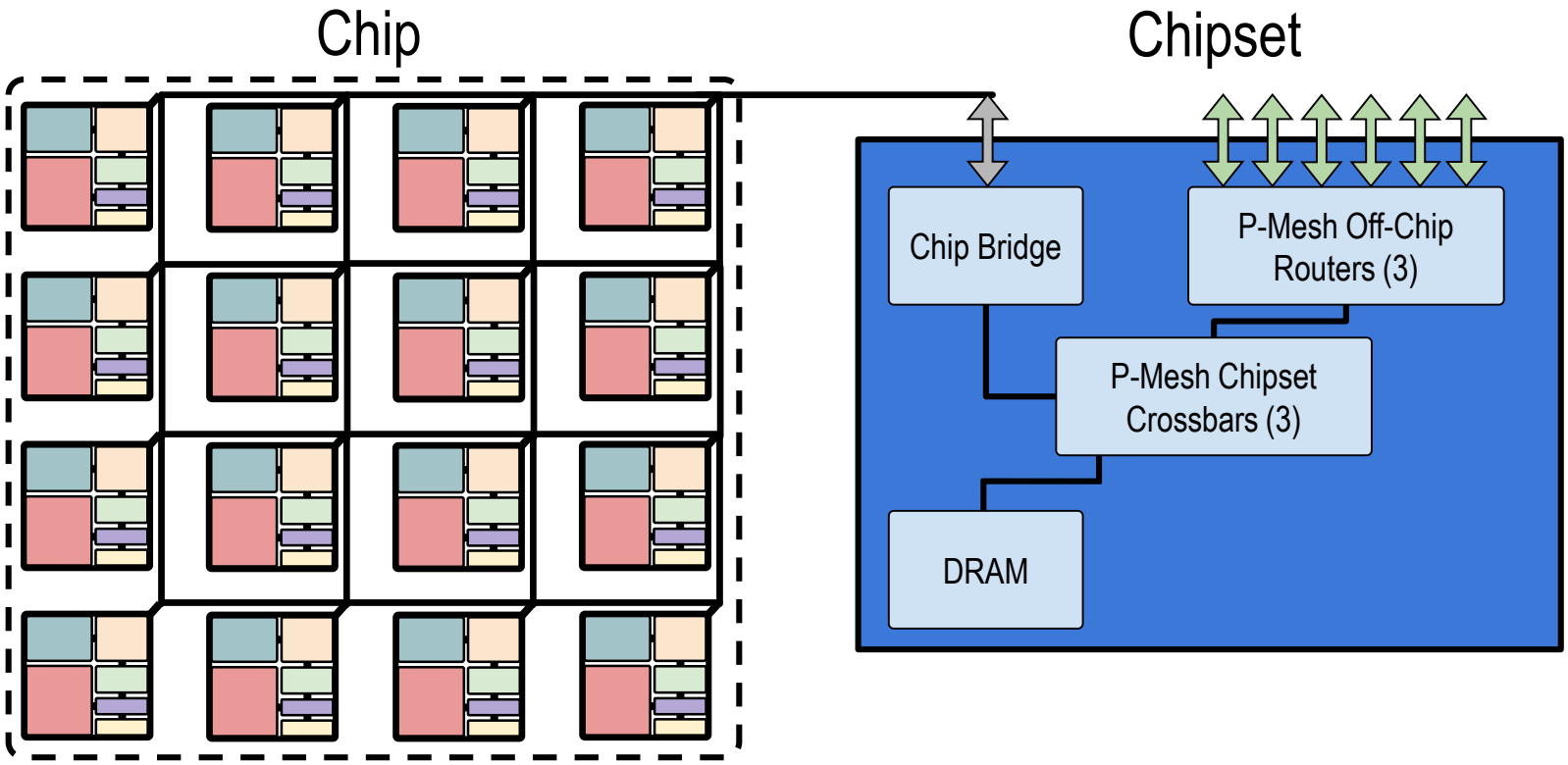
OpenPiton System Overview



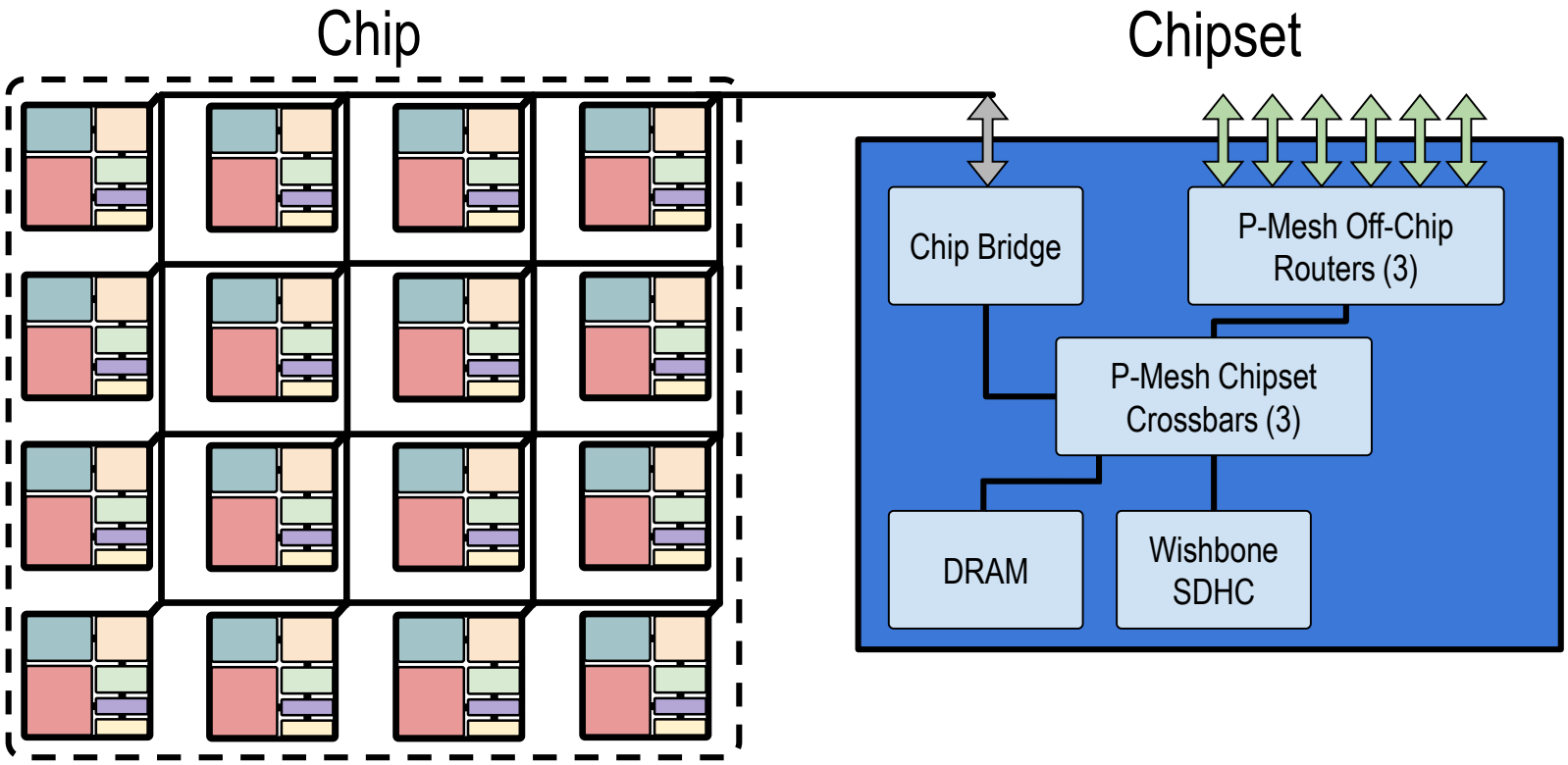
OpenPiton System Overview



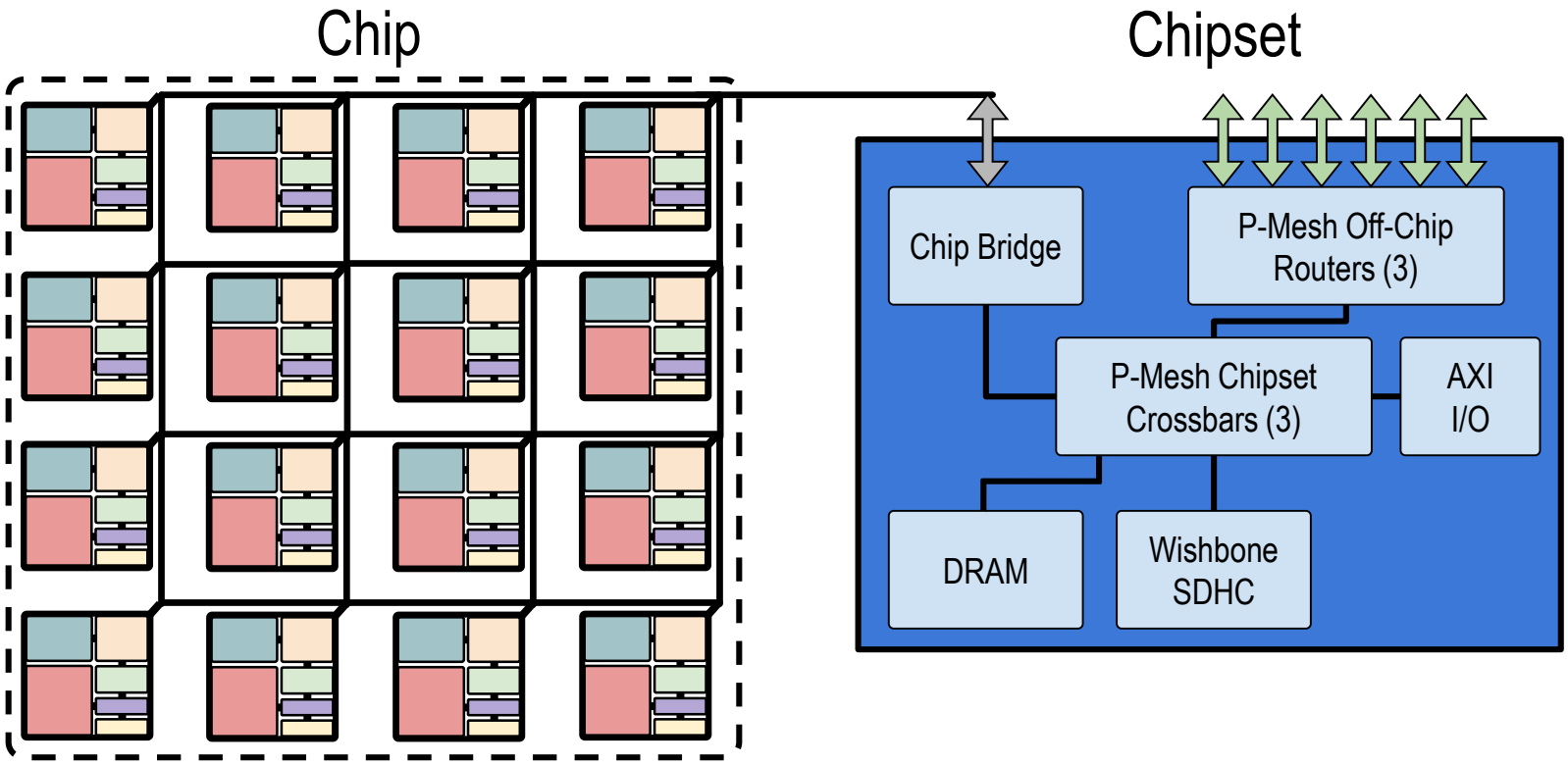
OpenPiton System Overview



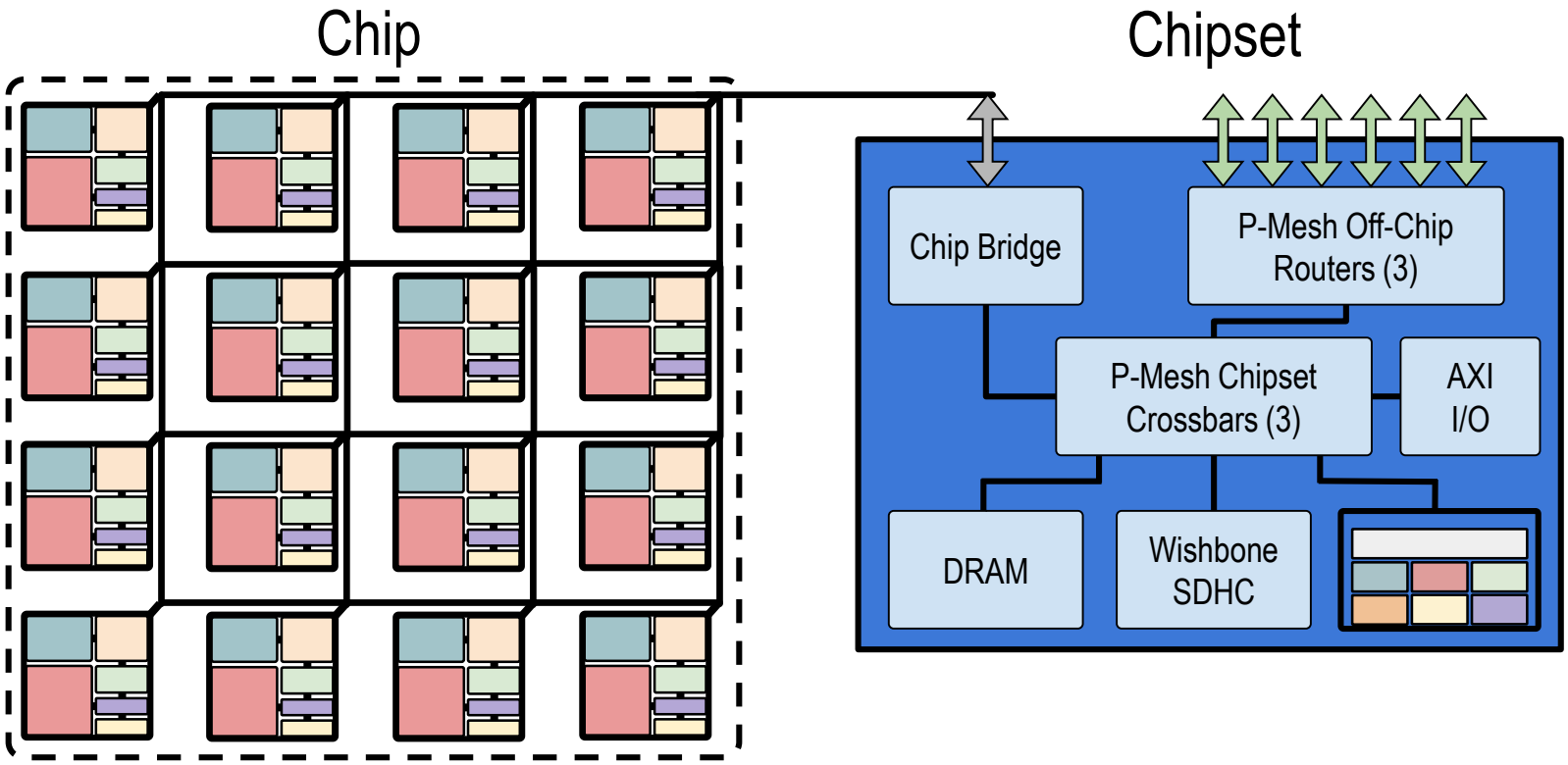
OpenPiton System Overview



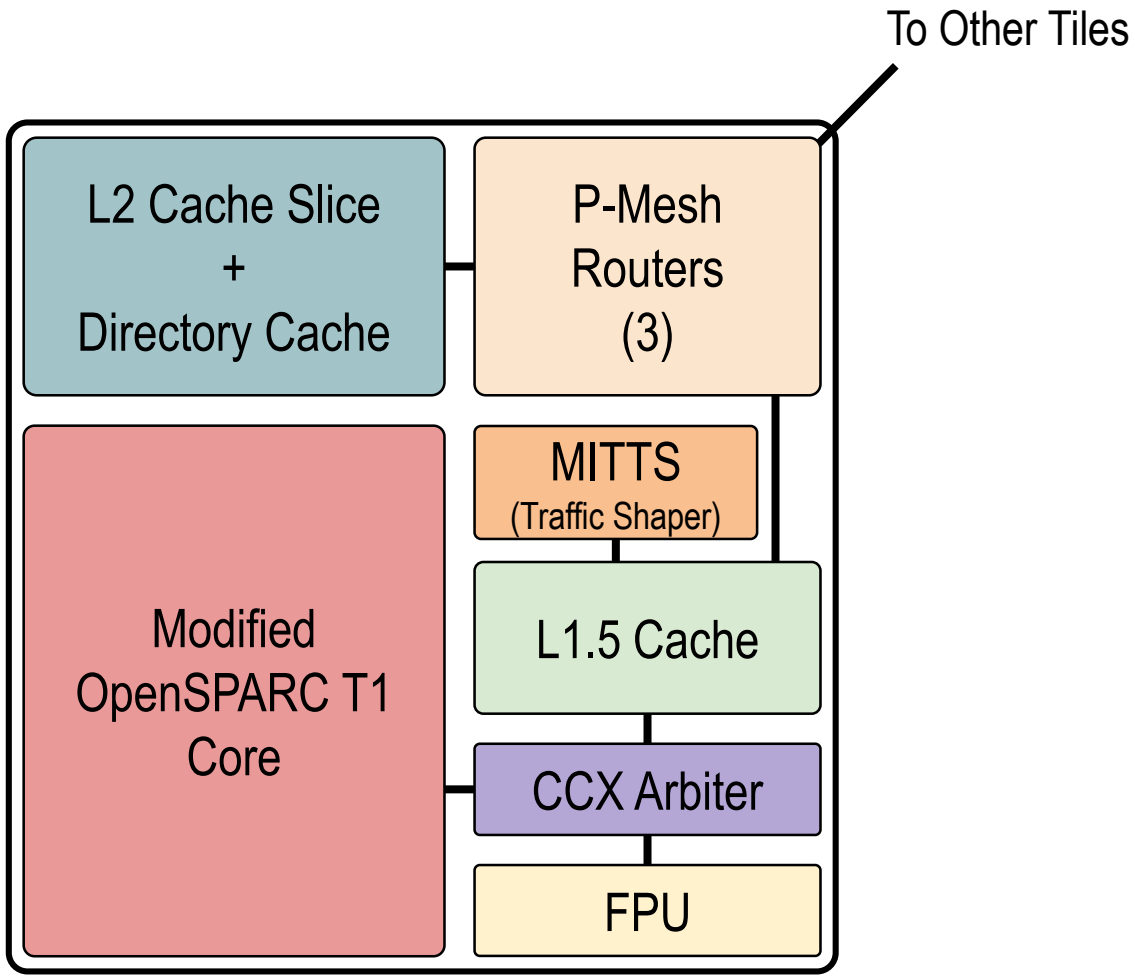
OpenPiton System Overview



OpenPiton System Overview



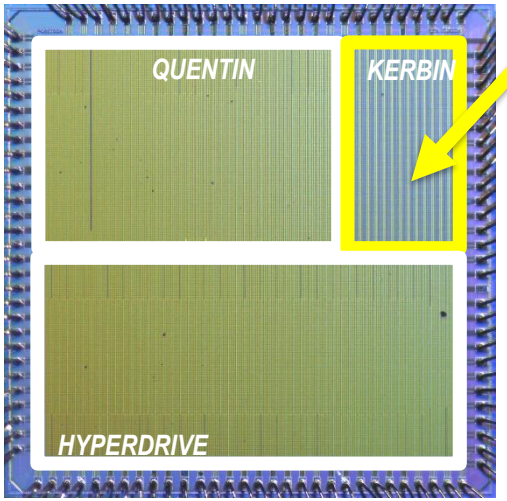
Tile Overview



Silicon Proven Designs: Ariane

- Ariane has been taped-out Globalfoundries 22nm FDX in 2017 and 2018
- The system features 16 kByte of instruction and 32 kByte of data cache.
- Poseidon:
  - Area: 0.23 mm2 – 175 kGE
  - 0.2 - 1.7 GHz (0.5 V – 1.15 V)
- Kosmodrom:
  - RV64GCXsmallFloat
  - Transprecision / Vector FPU
  - Ariane HP
    - 8T library, 0.8V, 1.3 GHz
    - 55 mW @ 1 GHz
  - Ariane LP
    - 7.5T ULP library, 0.5V, 250 MHz
    - 5 mW @ 200 MHz

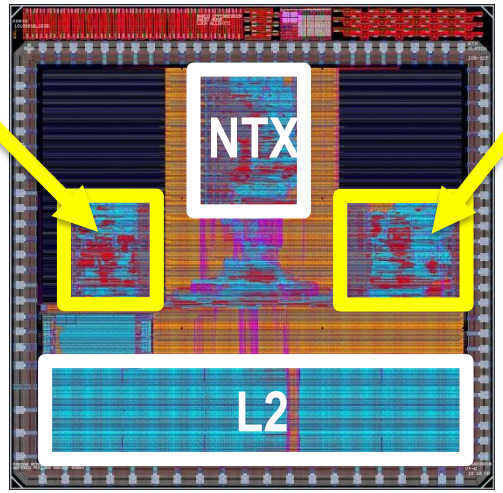
Poseidon layout



Ariane

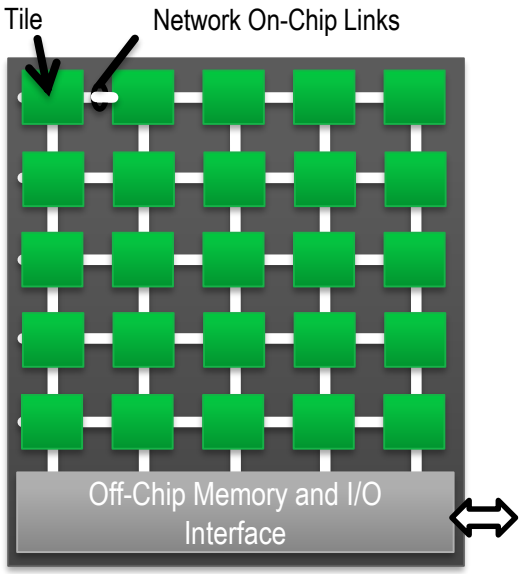
Kosmodrom layout

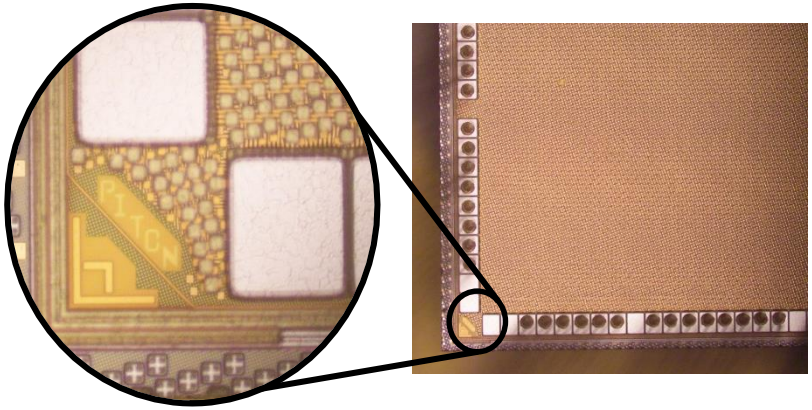
Ariane HP



Ariane LP

Silicon Proven Designs: Piton Chip



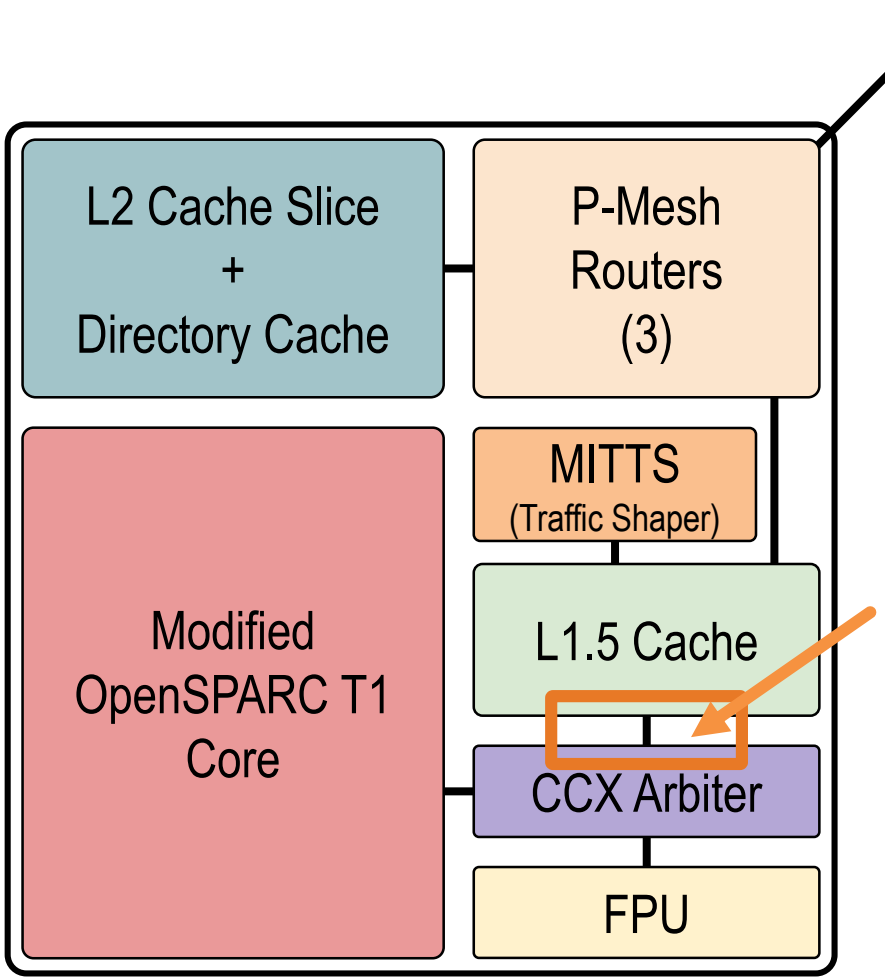


- 25-core
  - Modified OpenSPARC T1 Core
  - 64bit, 2 Threads per core
- 3 NoCs
  - 64-bit, 2D Mesh
  - Extend off-chip enabling multichip systems
- Directory-Based Cache System
  - 64KB L2 Cache per core (Shared)
  - 8KB L1.5 Data Cache
  - 8KB L1 Data Cache
  - 16KB L1 Instruction Cache
- IBM 32nm SOI Process
  - 6mm x 6mm
  - 460 Million Transistors
- Target: 1GHz Clock @ 900mV
- Among largest chips ever built in academia
- Received silicon and runs full-stack Debian in lab

OpenPiton

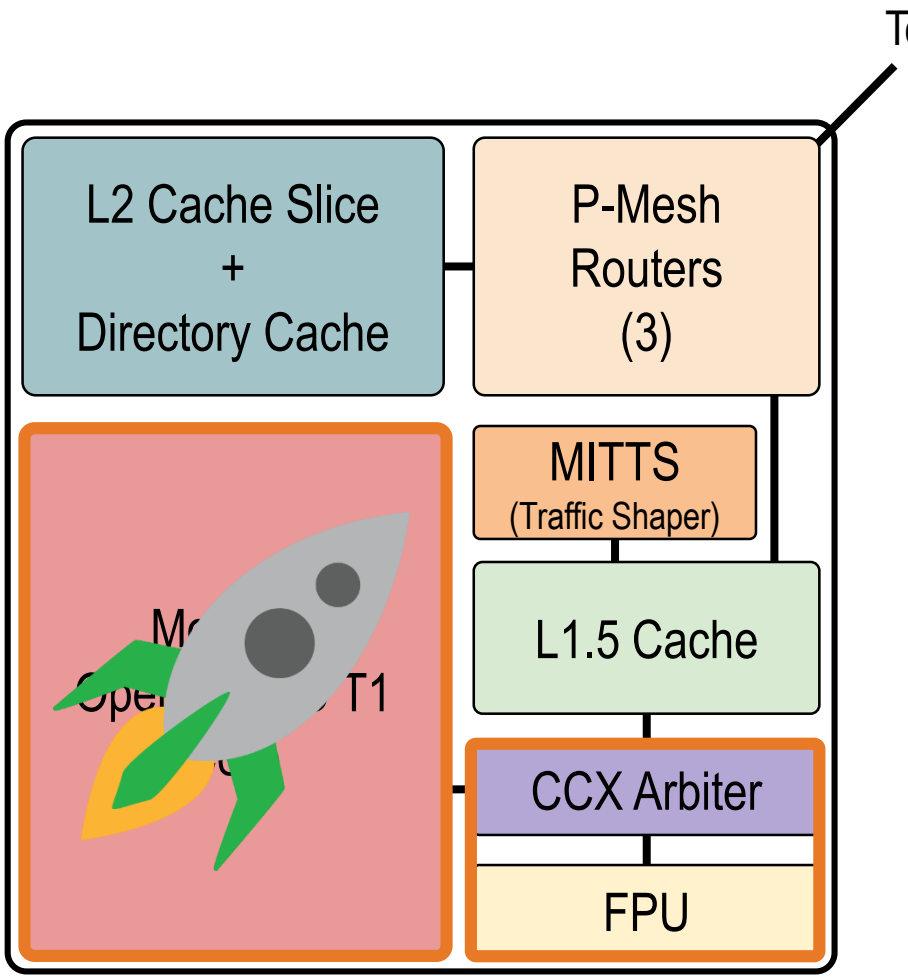
ETH

Putting it all together



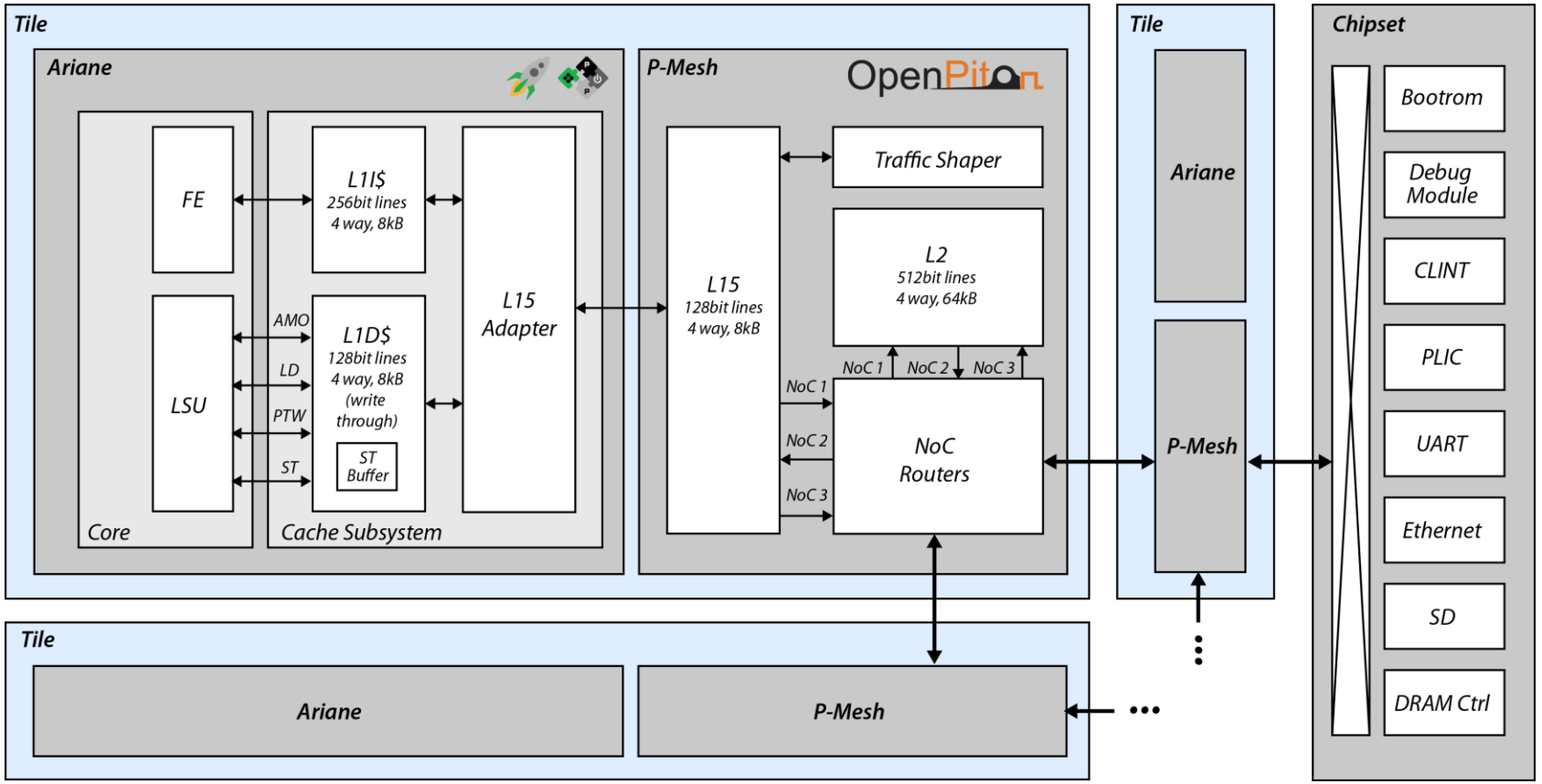
- Native L1.5 interface is the ideal point to attach a new core
- Well defined interface similar to CCX from OpenSPARC
- Write-through cache protocol
- Coherency mechanism: only need to support invalidation messages

Putting it all together

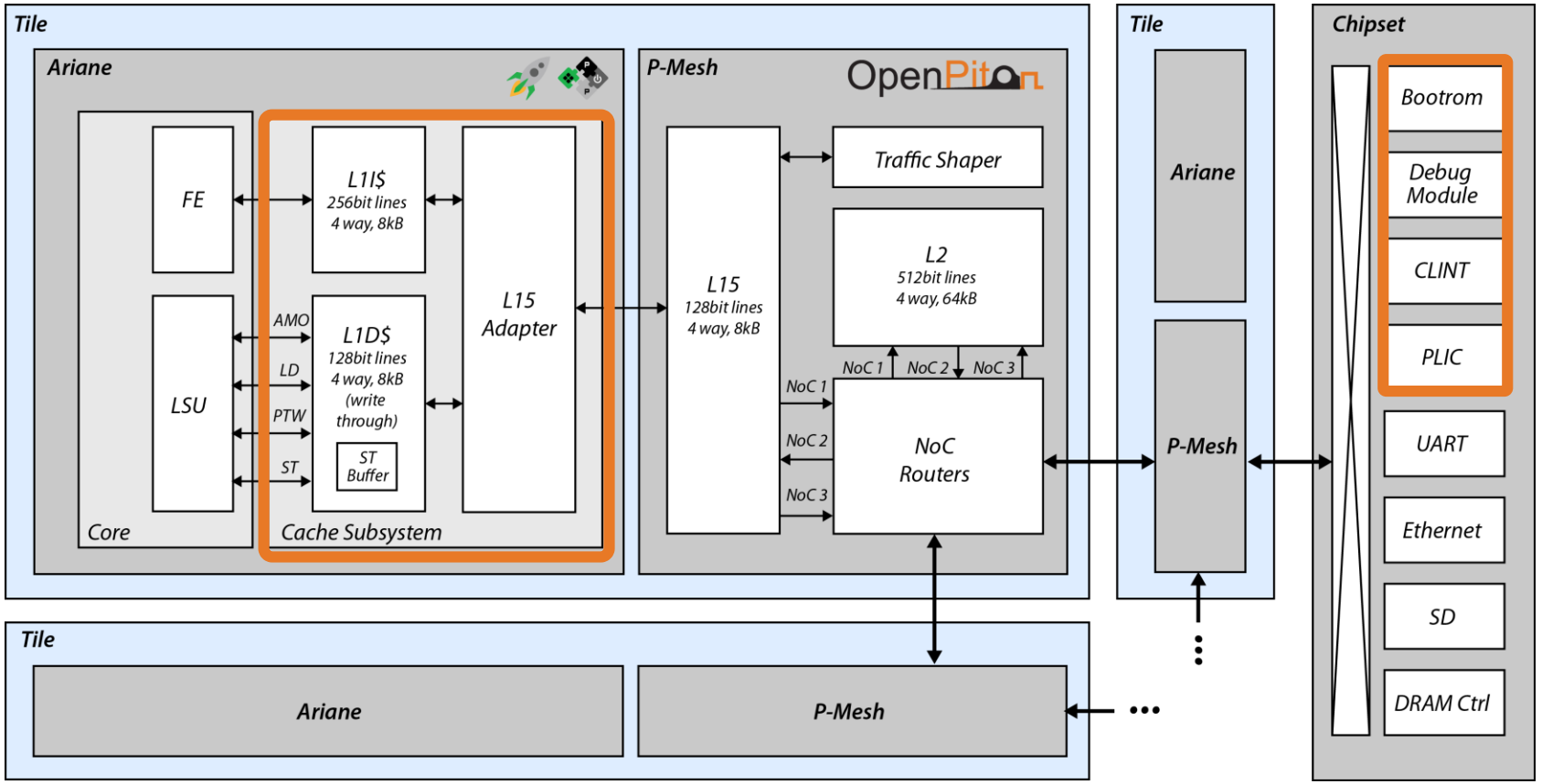


- Native L1.5 interface is the ideal point to attach a new core
- Well defined interface similar to CCX from OpenSPARC
- Write-through cache protocol
- Coherency mechanism: only need to support invalidation messages

OpenPiton + Ariane



OpenPiton + Ariane



- New write-through cache subsystem with support for invalidations and the L1.5 interface
- RISC-V peripherals

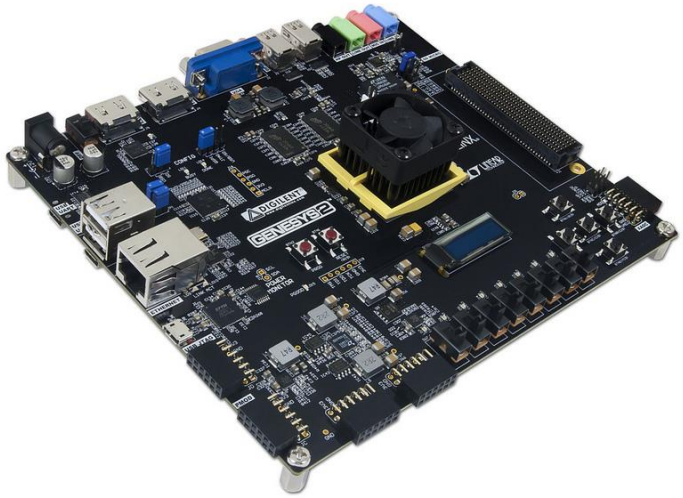
Configurability Options

Component	Configurability Options	
Cores (per chip)	Up to 65,536	
Cores (per system)	Up to 500 million	
Core Type	OpenSPARC T1	Ariane 64bit RISCV
Threads per Core	1/2/4	1
Floating-Point Unit	FP64, FP32	FP64, FP32, FP16, FP8, BFLOAT16
Stream-Processing Unit	Present/Absent	-
TLBs	8/16/32/64 entries	16 entries (configurable)
L1 I-Cache	8*/16/32KB	
L1 D-Cache	4*/8/16KB	
L1.5 Cache	Number of Sets, Ways	
L2 Cache	Number of Sets, Ways	
Intra-chip Topologies	2D Mesh, Crossbar	
Inter-chip Topologies	2D Mesh, 3D Mesh, Crossbar, Butterfly Network	
Bootloading	SD/SDHC Card, UART	

*L1 cache goes to 2-ways at smallest size

FPGA Mapping

- Digilent Genesys2
  - $999 Board ($600 academic)
  - Kintex-7 XC7K325T
  - 1 core config:
    - 8kB L1D, 8kB L1.5, 16kB L1I, 64kB L2
    - 66 MHz
    - 85k LUT (42%)
    - 67 BRAM (15%)



- Xilinx VCU118 (available soon)
  - $7000 Board
  - XCVU9P for large manycore configurations
  - >100MHz feasible

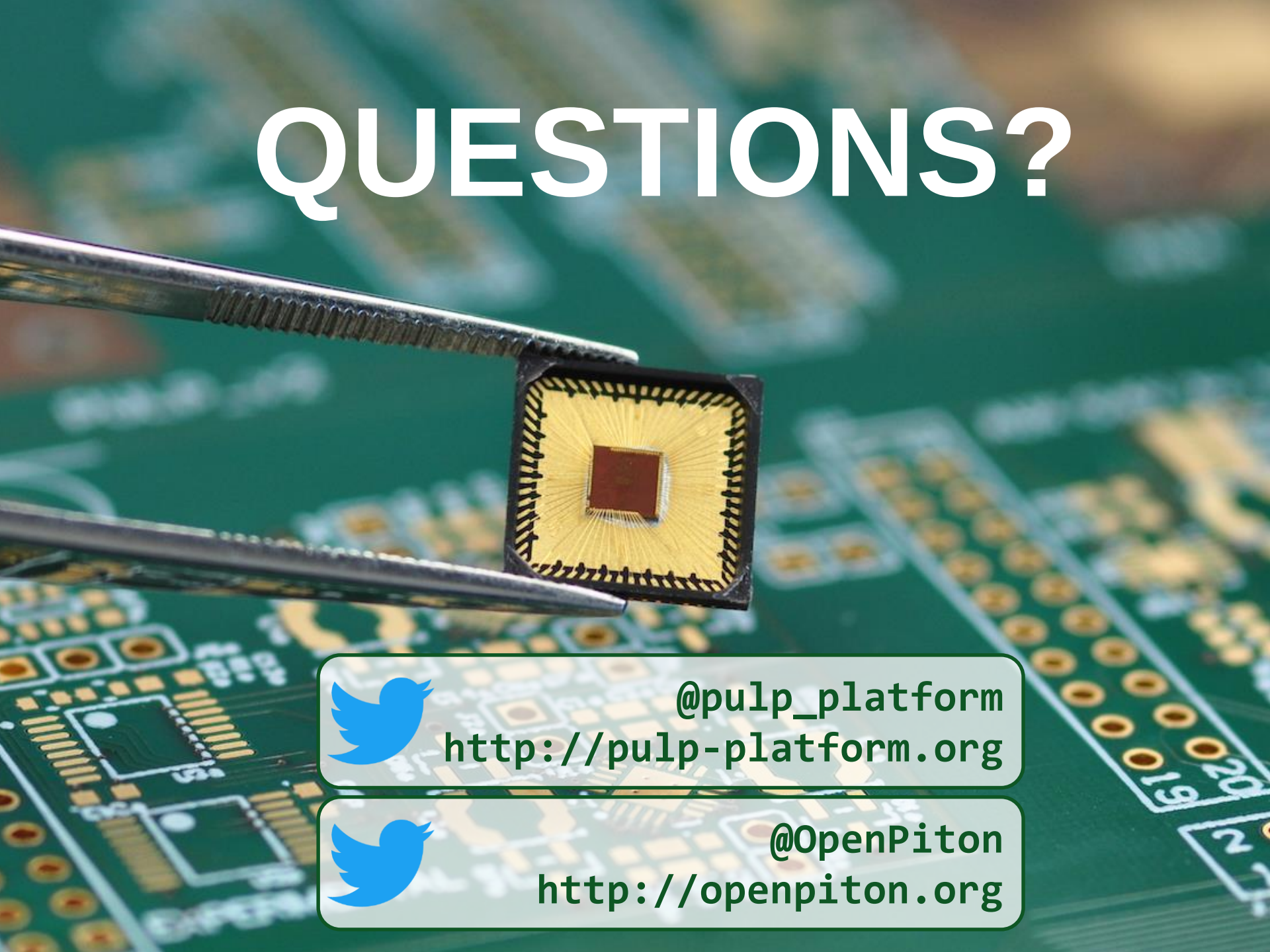
WIP - Planned Improvements

- **First Version Released in December 2018**
  - Bare metal on 1 or several tiles.
  - Genesys2 FPGA mapping.

- **Upcoming Improvements in Q1 2018**

  - Floating point support.
  - RISC-V FESVR support in simulation.
  - Thorough validation of cache coherence.
  - Synthesis flow for large FPGAs (e.g., VCU118 with VUP9).
  - RISC-V Compliant Interrupt Controllers. The CLINT and PLIC have been included, but are not fully tested yet.
  - Support for simulation with Synopsys VCS.
  - Performance enhancements (cache re-parameterization, write-buffer throughput).
  - Single-core and SMP Linux support.

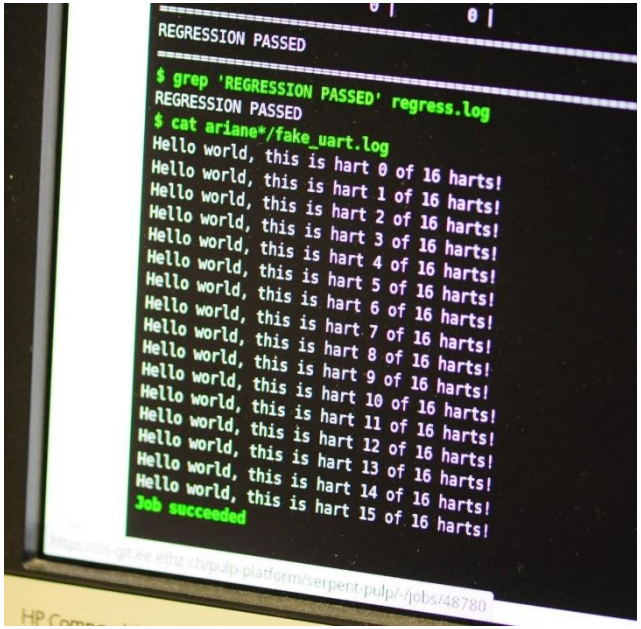
QUESTIONS?

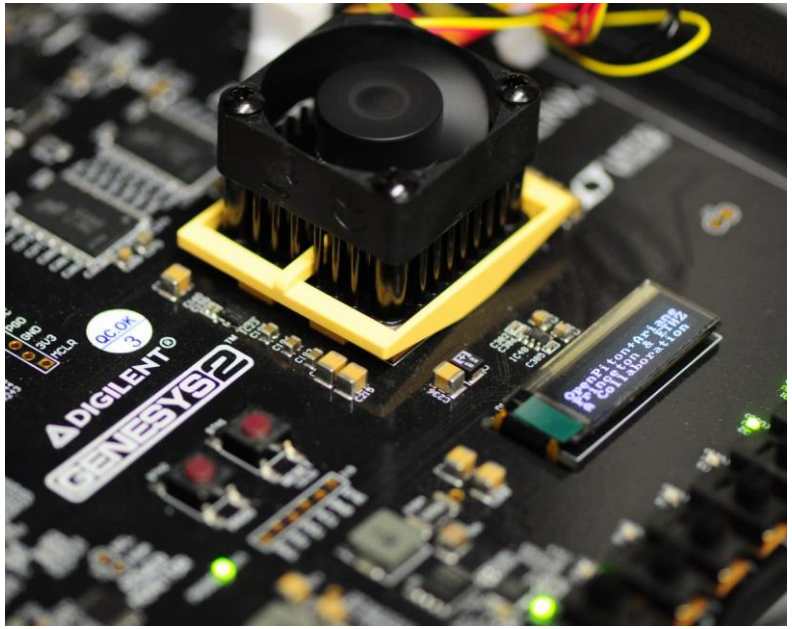


@pulp_platform
http://pulp-platform.org

@OpenPiton
http://openpiton.org

Part 2 – Walkthrough and Demo





- From scratch to simulation
- FPGA demo

Relevant Resources

- RTL, scripts, documentation, cross-compiler, FPGA disk images & FPGA bit files:
  - Grab the latest from http://openpiton.org
  - Clone/Fork our GitHub repository https://github.com/PrincetonUniversity/openpiton
  - Ariane related documentation in GitHub Readme (examples of this walkthrough can be found there)
- Ariane repository is a submodule, so no need to check out manually (https://github.com/pulp-platform/ariane)

Environment Setup – Tools and Paths

- Requirements:
  - Ubuntu 16.04 or Springdale (Red Hat) 6.6
  - Vivado: 2018.2 and newer
  - Mentor QuestaSim 10.6a
- Install the dependencies listed in `piton/ariane_setup.sh`
- Adapt `piton/ariane_setup.sh` and `piton/piton_settings.bash` to reflect your local setup
  - Set tool versions correctly (QuestaSim, Vivado)
  - Make sure that $RISCV points to your local installation of the RISC-V tools (it is recommended to build them using the scripts provided, see next slide)

Environment Setup – Getting Started

1. Source required tool scripts
   - QuestaSim, Vivado, etc

2. Source `piton/ariane_setup.sh`
   - Source this from the root of the repo
   - `$PITON_ROOT` then points to the root directory

3. When running for the first time, build the RISC-V tools with `piton/ariane_build_tools.sh`
   - This includes the RISC-V compiler, the RISC-V assembly tests and benchmarks and the RISC-V FESVR

Where is everything? - `$PITON_ROOT`

- `piton/`
  - Aliased to `$DV_ROOT`
  - Home to RTL, tools, assembly tests
- `build/`
  - Aliased to `$MODEL_DIR`
  - Temporary build files, files from FPGA flow
- `docs/`
  - Documentation as seen on `http://openpiton.org`

Where is everything? - `$PITON_ROOT/piton/`

- `design/`
  - Top level of the RTL module tree
  - Structure follows Verilog module hierarchy
  - The Ariane subrepository is located under `design/chip/tile/ariane`
- `tools/`
  - Home to all simulation, synthesis, FPGA tools
- `verif/`
  - Location for all verification-related files

Useful Paths

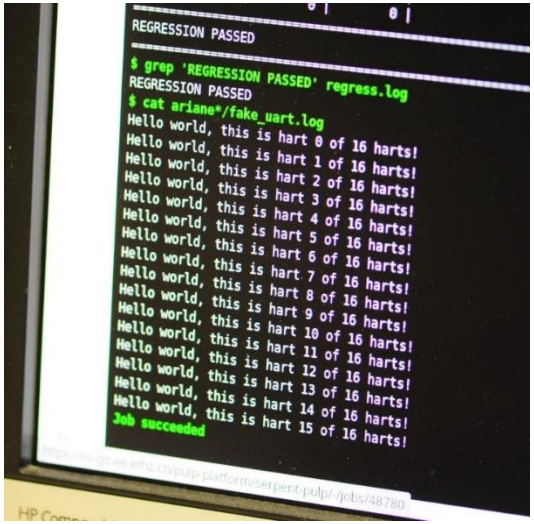
- Where's the RTL?
  - `piton/design/*/rtl/`
- Where are the assembly test cases?
  - `piton/verif/diag/assembly/`
- Where are the module-agnostic backend scripts?
  - FPGA: `piton/tools/src/proto/`
  - ASIC: `piton/tools/synopsys/`
- Where are the module-specific backend scripts?
  - FPGA: `piton/design/*/xilinx/`
  - ASIC: `piton/design/*/synopsys/script/`

What can I do with OpenPiton+Ariane?

- Simulation
- ASIC Synthesis & Backend (not supported yet)
- FPGA Synthesis & Backend
- Validation
- Configuration
- OS/Hypervisor Development (not supported yet)

Anatomy of a Simulation

- Simulation model
  - Design under test (DUT) RTL
  - Top-level test bench
  - Simulator compiler arguments
    - Verilog macros, include directories, monitor params, etc.



- Test stimuli
  - Assembly tests
  - C tests
  - Source/sink bit vectors
    - Based on infrastructure from Christopher Batten's group at Cornell

OpenPiton Simulation Models

Table 3: OpenPiton Simulation Models

Name	Type
manycore	C/Assembly
chip_fpga_bridge	Unit Test
dmbr	Unit Test
dmbr_test	Unit Test
fpga_chip_bridge	Unit Test
fpga_fpga_hpc_bridge	Unit Test
fpga_fpga_lpc_bridge	Unit Test
ifu_esl	Unit Test
ifu_esl_counter	Unit Test
ifu_esl_fsm	Unit Test
ifu_esl_htsm	Unit Test
ifu_esl_lfsr	Unit Test
ifu_esl_rtsm	Unit Test
ifu_esl_shiftreg	Unit Test
ifu_esl_stsm	Unit Test
jtag_testbench	Unit Test

Address Map

- OpenPiton has a 40bit physical address space
- Simulation configuration of manycore model:
  - `piton/verif/env/manycore/devices_ariane.xml`
- FPGA build configuration of the full system for the Genesys2 board:
  - `piton/design/xilinx/genesys2/devices_ariane.xml`
- DRAM starts at 0x00_8000_0000 (aligned with other RISC-V 64bit systems), cached range is currently set to 1GB
- I/O region with peripherals is above 0x80_0000_0000
- Ariane cores start fetching from a boot rom and jump to the DRAM base (each core has its own ID)

Simulation Scripts/Tools

- `sims`
  - `piton/tools/src/sims/sims,2.0`
  - Adapted from OpenSPARC T1
  - Build individual simulation models and run test stimuli
  - Regressions (single simulation model)
  - Calls RISC-V GCC compiler, bare-metal environment at the moment
  - OpenPiton+Ariane configuration currently only runs with QuestaSim
    - VCS support will be available soon

- Required `sims` arguments

```
-sys=<simulation model>
-msm_build
```

- Other useful arguments

```
-msm_build_args=<MSM arguments>
-build_id=<name>
```
  - Default is `rel-0.1`

```
-ariane
-config_rtl=MINIMAL_MONITORING
```

Example: The `manycore` Model with Ariane

- `cd build`
  - Tool output creates many files…
- `sims -sys=manycore -msm_build -ariane`
  - `sims.log`: check for build errors
  - `build/manycore/rel-0.1/`
- Default: 1x1 tile configuration
- Can be parameterized, e.g., to 2x2 tiles
  - `sims -sys=manycore -x_tiles=2 -y_tiles=2 -msm_build -ariane`

Running a Simulation

- Required `sims` arguments

  `-sys=<simulation model>`

  `-msm_run`

  `<test stimuli>`

  - Varies by simulation model type (assembly file, source/sink prefix)

- Other useful arguments

  `-build_id=<name>`

  `-gui`

  `-ariane`

  `-rtl_timeout <clock cycles>`

  `-finish_mask <hexvalue>`

  `-precompiled`

Simulation Outputs

- Depends on test type
- `stdout and sims.log`
  - `PASS (HIT GOOD TRAP)`
  - `FAIL (HIT BAD TRAP)`
  - `Or timeout…`
- Test binary (`diag.exe`)
- Memory image (`mem.image`)
- Symbol table (`symbol.tbl`)
- Performance log (`perf.log`)
- Status log (`status.log`)
- “Fake UART” (`fake_uart.log`)

Example: Hello World

- 1x1 tile configuration:
  - `sims -sys=manycore -msm_build -ariane`
  - `sims -sys=manycore -msm_run -ariane hello_world.c`
  - Check `fake_uart.log`
- 2x2 tile configuration:
  - `sims -sys=manycore -msm_build -ariane -x_tiles=2 -y_tiles=2 -config_rtl=MINIMAL_MONITORING`
  - `sims -sys=manycore -msm_run -ariane -x_tiles=2 -y_tiles=2 -rtl_timeout 1000000 -finish_mask 0x1111 hello_world_many.c`

Example: RISC-V Assembly Tests and Benchmarks

- Only supported in 1x1 tile configuration:
  - `sims -sys=manycore -msm_build –ariane -config_rtl=MINIMAL_MONITORING`
- Assembly tests (e.g., `rv64ui-p-addi`):
  - `sims -sys=manycore -msm_run -ariane rv64ui-p-addi.S -precompiled`
- Benchmarks (e.g., `dhrystone.riscv`):
  - `sims -sys=manycore -msm_run -ariane dhrystone.riscv –precompiled`

Batch Regressions

- Diaglists (`piton/verif/diag/*.diaglist`)
  - Groups of assembly tests and assembly test declarations
  - Assembly test declaration

    ```
    label testfile.s <sims arguments>
    ```

  - Assembly group definitions

    ```
    <groupname sys=mymodel sims args>
            test1 test1.s
    </groupname>
    ```

  - Groups can be nested

- See: `piton/verif/diag/master_diaglist_princeton`

- For a summary, step into the build subfolder and call

  ```
  regreport . -summary
  ```

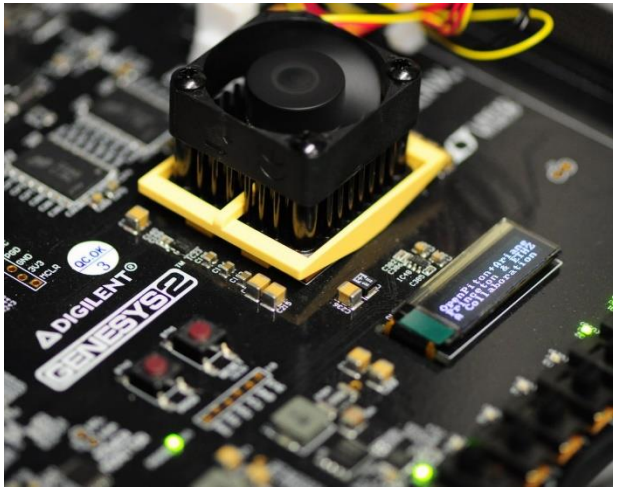
Example: Batch Regressions

- Some examples for batch regressions:

```
sims -group=ariane_tile1_asm_tests_p -sim_type=msm
sims -group=ariane_tile1_asm_tests_v -sim_type=msm
sims -group=ariane_tile1_amo_tests_p -sim_type=msm
sims -group=ariane_tile1_amo_tests_v -sim_type=msm
sims -group=ariane_tile1_benchmarks -sim_type=msm
sims -group=ariane_tile1_simple -sim_type=msm
sims -group=ariane_tile16_simple -sim_type=msm
```

FPGA Mapping

- Currently supported board: Genesys2
  - Build time ~ 1h on a recent desktop machine



- Uses `protosyn` build script
  - See also FPGA manual from OpenPiton
  - Tested with Vivado 2018.2

- Run and debug bare-metal programs in-system using
  - The `pitonstream` utility (UART-based)
  - A RISC-V debug-spec compliant debug environment
    - e.g., Olimex JTAG adapter + OpenOCD + GDB
    - Not covered in this tutorial, see OpenPiton readme for an example

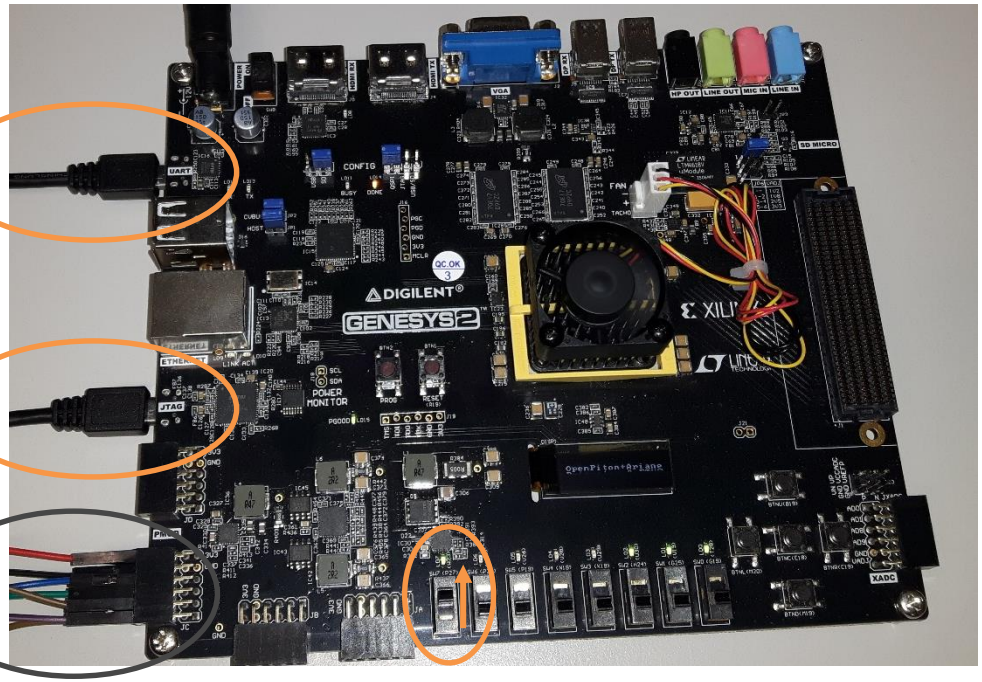
Building with `protosyn`

- ```
  protosyn -b <board_type>
  [-d <design>]
  [--bram-test <test_name>]
  [--from <FPGA flow step>]
  [--to <FPGA flow step>]
  [--no-ddr]
  [--uart-dmw ddr]
  [--eth]
  [--oled <string>]
  [--core [sparc|pico|pico_het|ariane]
  [… more options to override system default config]
  ```

- To build a 1x1 tile configuration for Genesys2 with `pitonstream` support:
  - `protosyn -b genesys2 -d system --core=ariane --uart-dmw ddr`
  - Output under `build/genesys2`
  ```

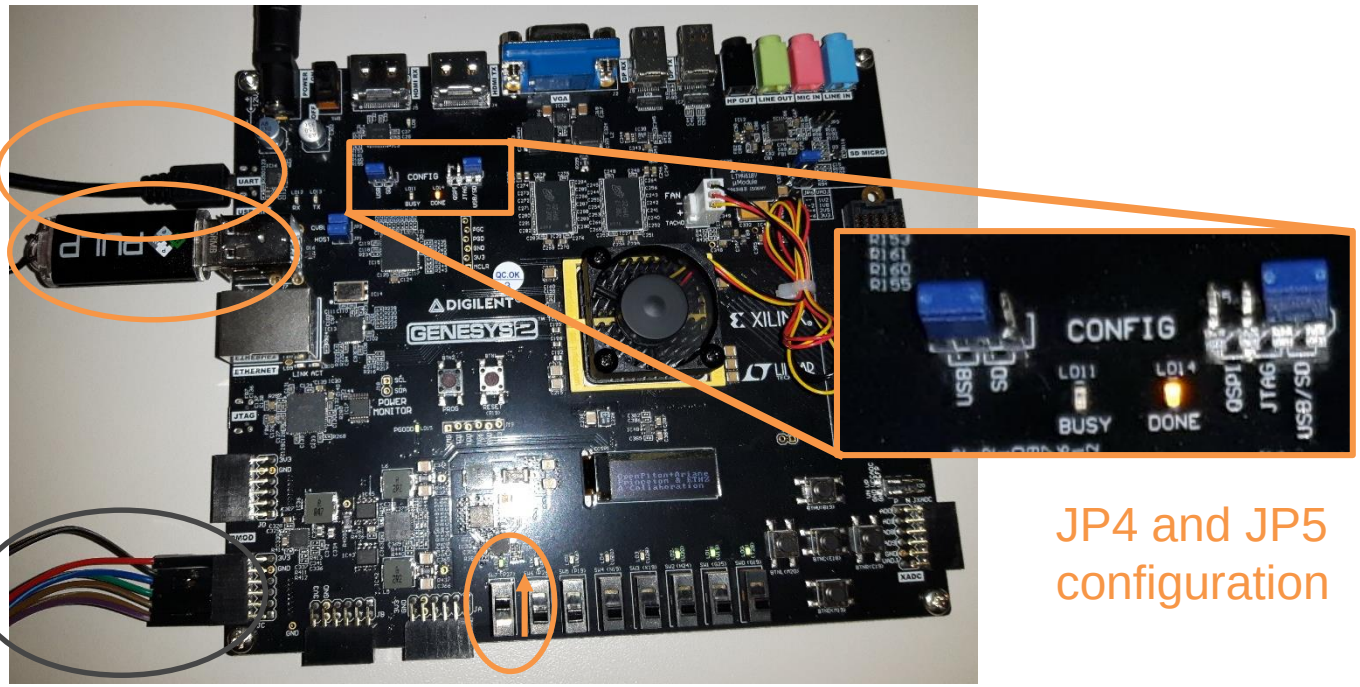
FPGA Board Setup for `pitonstream` (V1)

- Connect the JTAG and UART USB ports to your system first (2 micro-USB cables)
- Open Vivado -> Open Hardware Manager -> Open Target -> Auto Connect -> Program Device
- Select the bitfile that has been build with `protosyn`, and program the device
- Flip the slider switch 7 up on the Genesys2 board to enable UART loading



FPGA Board Setup for `pitonstream` (V2)

- Connect UART USB port to your system first (1 micro-USB cable)
- Put one single *.bit file onto a FAT-32 formatted USB stick
- Put the USB drive into the upper USB port of the board, configure JP4 and JP5 as shown below
- Flip the slider switch 7 up on the Genesys2 board to enable UART loading



Running a Bare-metal Program using `pitonstream`

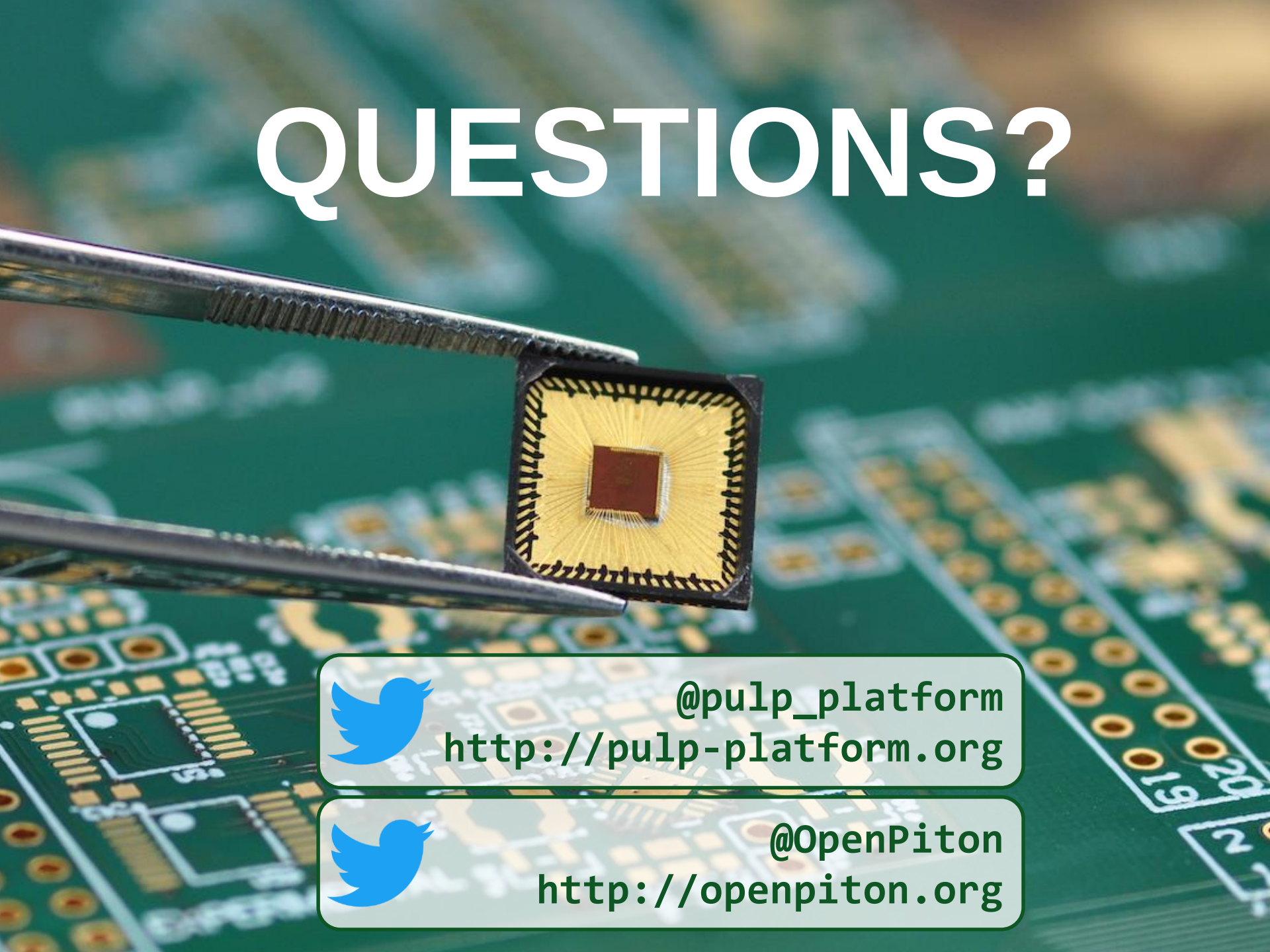
- `pitonstream -b <board_type> -d <design>`
  `-f <file listing the tests to run>]`
  `--core [sparc|ariane]`
  `--precompiled`

- `pitonstream` calls `sims` to compile, loads and runs the binary, and checks for good/bad traps (these are encoded in the binary as memory loads to "magic" addresses).

- To run hello_world.c:
  - Make a new file called `tests.list` and write `hello_world.c` on the first line, save and close it
  - `pitonstream -b genesys2 -d system --core=ariane`
    `-f ./tests.list`

- In order to run the RISC-V benchmarks:
  - `pitonstream -b genesys2 -d system --core=ariane --precompiled`
    `-f ./piton/design/chip/tile/ariane/ci/riscv-benchmarks.list`

QUESTIONS?



@pulp_platform
http://pulp-platform.org

@OpenPiton
http://openpiton.org

Open-source RISC-V Cores

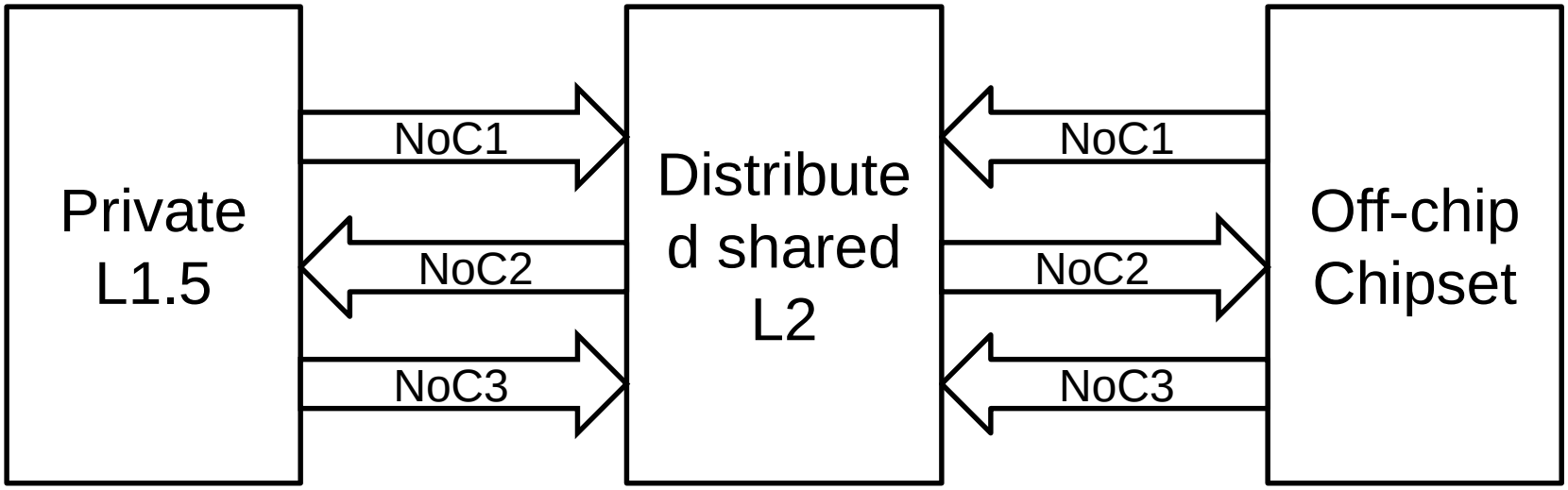
Name	Description	Language	Maintainer	Note	Licence
High Performance					
Rocket	RV64GC with full privileged spec	Chisel	SiFive, UC BAR	It comes with its own SoC	BSD
BOOM	Out-Of-Order RV64G with full privileged spec	Chisel	Esperanto	It comes with its own SoC	BSD
Ariane	RV64GC with full privileged spec	SV	ETH Zurich		Solderpad
Embedded Systems					
Riscy	RV32IM[F]CXpulp with limited privileged spec	SV	ETH Zurich, University of Bologna	DSP, bit manipulation, packet-SIMD support	Solderpad
Zero-riscy	RV32{E,I}[M]C	SV	ETH Zurich, University of Bologna	Ultra-low-Area	Solderpad
ORCA	RV32IM	VHDL	VectorBlox	Optimized for FPGA	BSD
VexRiscv	RV32I[M] with option MMU	Scala	SpinalHDL	Optimized for FPGA	MIT Licence
RV12	RV{32,64}I	Verilog	RoaLogic		Non Comm Licence

Cache Coherence Protocol

Directory-based MESI coherence Protocol

- Four-hop message communication (no direct communication between private L1.5 caches)
- Uses 3 physical NoCs with point-to-point ordering to avoid deadlock
- The directory and L2 are co-located but state information are maintained separately
- Silent eviction in E and S states
- No need for acknowledgement upon write-back of dirty lines from L1.5 to L2, but writeback guard needed in some cases.

Memory Hierarchy Datapath



NoC Messages

In order to avoid deadlock, NoC3 messages will never be blocked