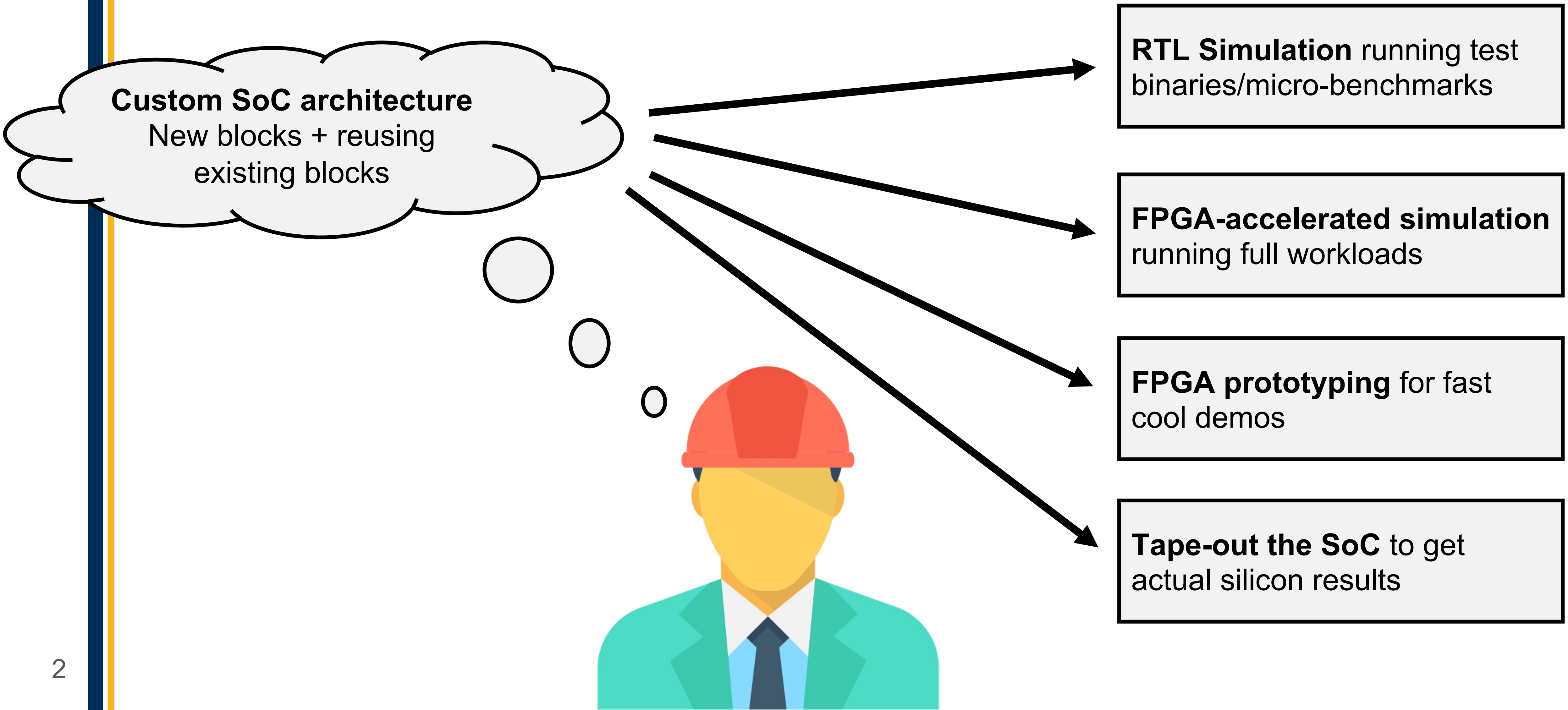
CHIPYARD

Architecture and Components

Use Cases



Custom SoC architecture
New blocks + reusing existing blocks

RTL Simulation running test binaries/micro-benchmarks

FPGA-accelerated simulation running full workloads

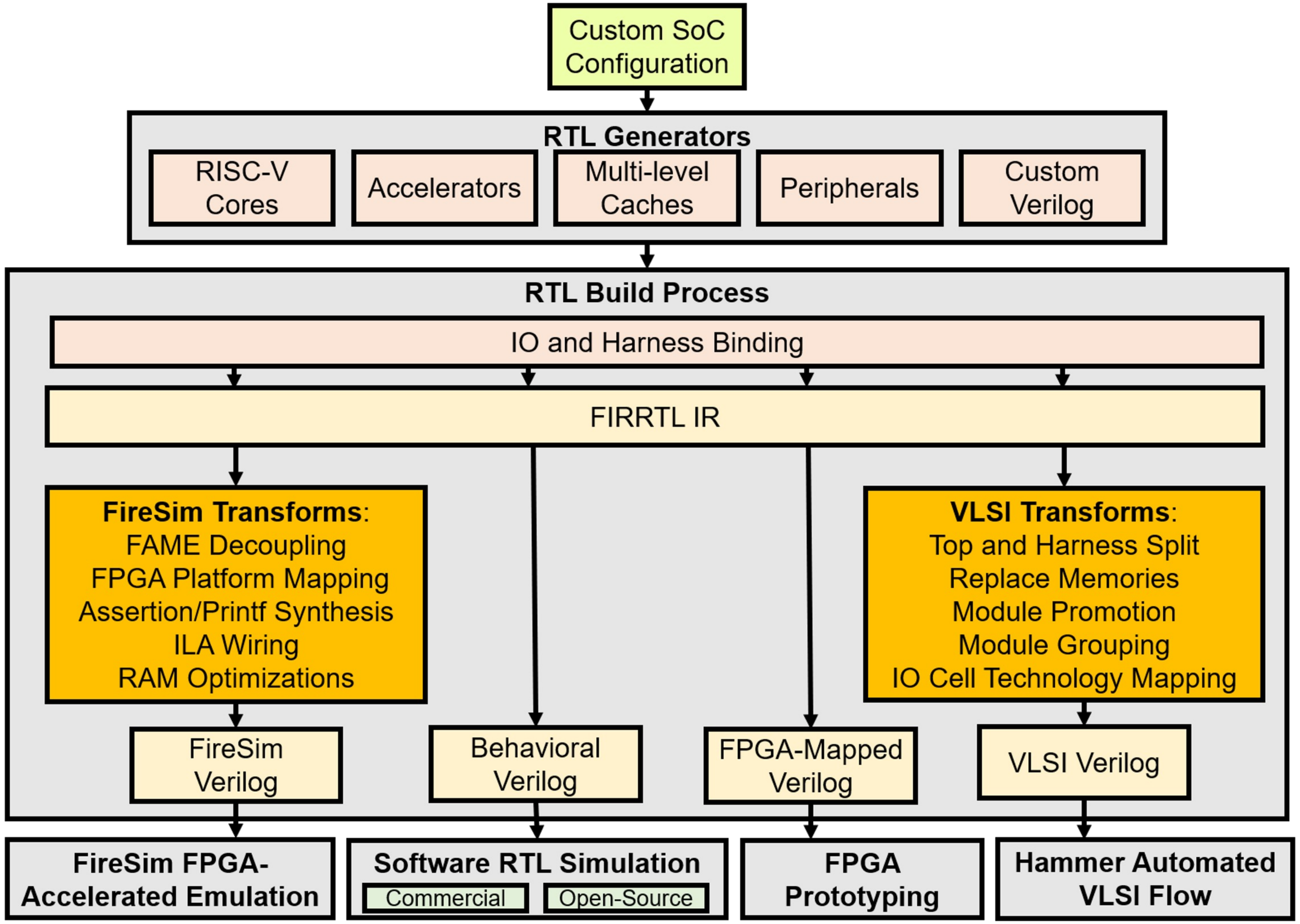
FPGA prototyping for fast cool demos

Tape-out the SoC to get actual silicon results

CHIPYARD Organization

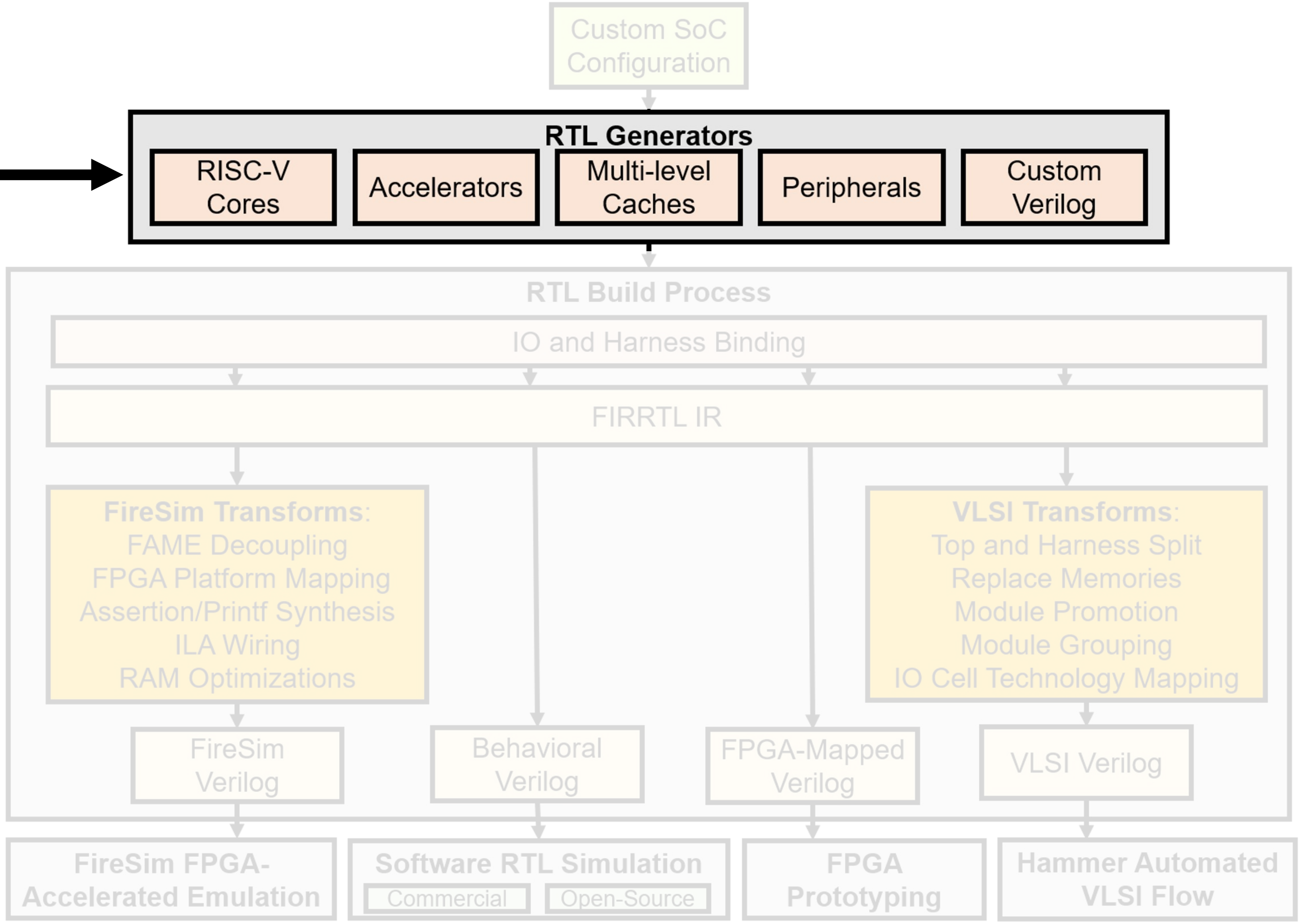
What is Chipyard?

- An organized **framework** for various SoC design tools
- A **curated IP library** of open-source RISC-V SoC components
- A **methodology** for agile SoC architecture design, exploration, and evaluation

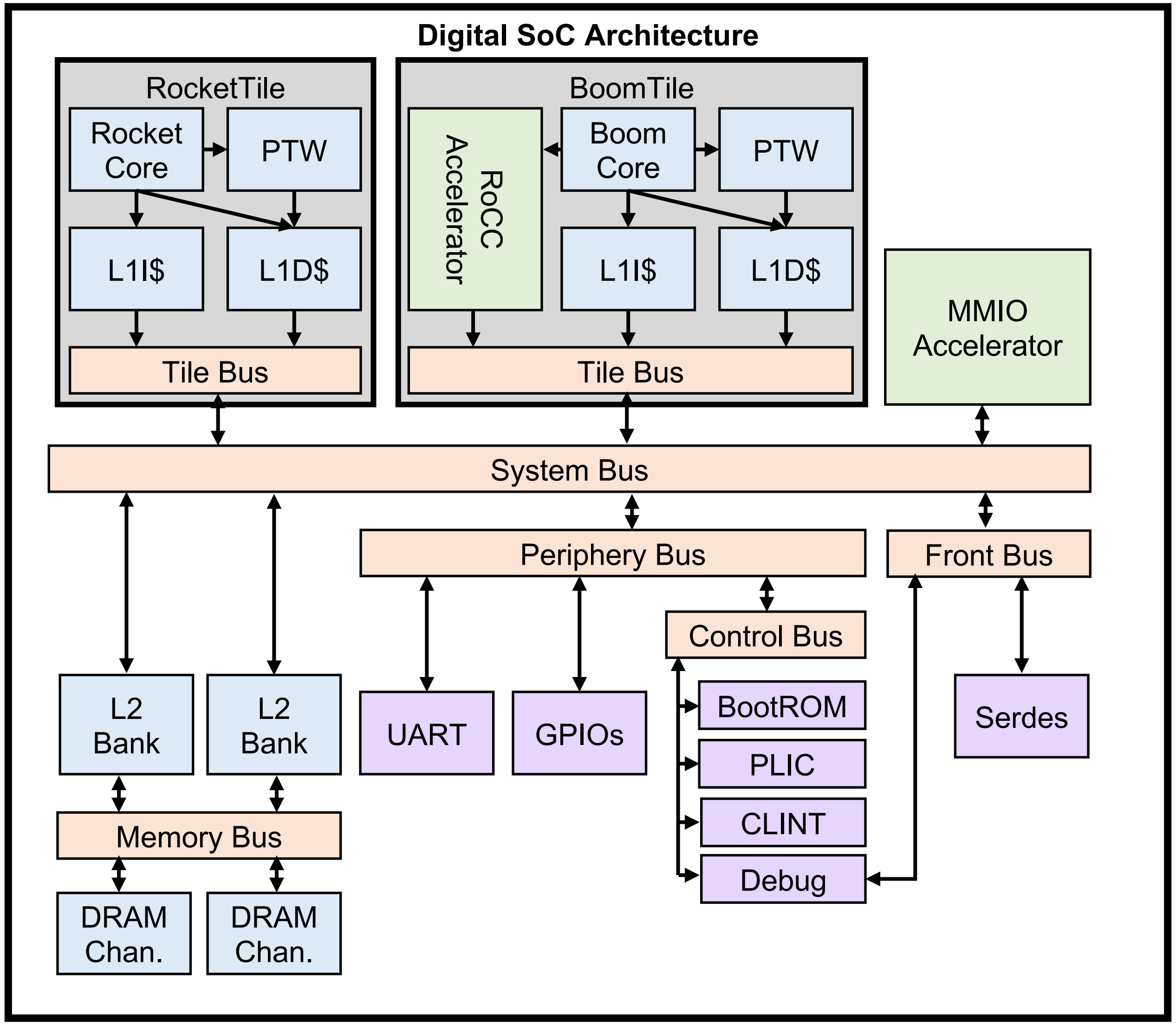


CHIPYARD Organization

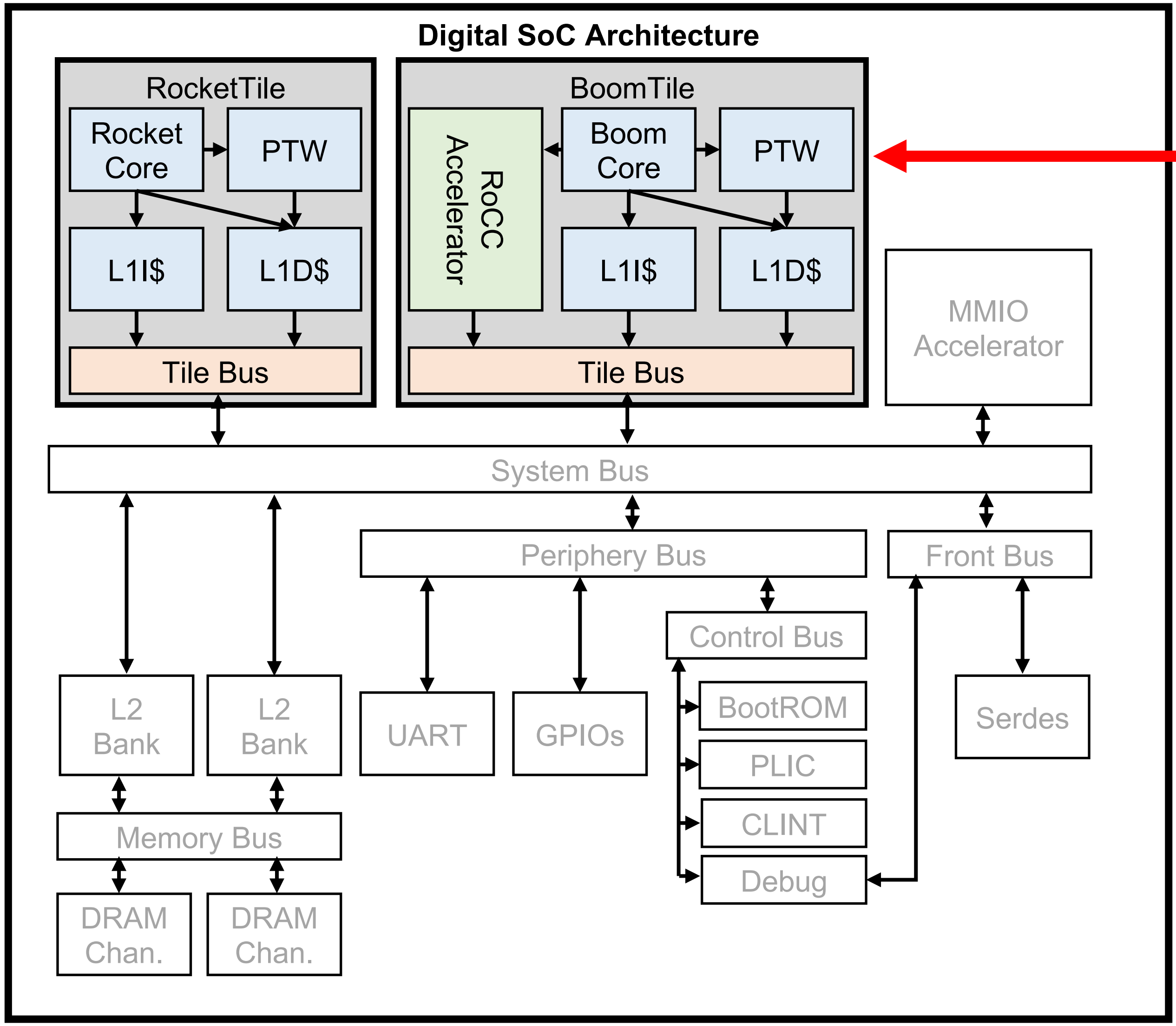
SoC architecture and generators



SoC Architecture



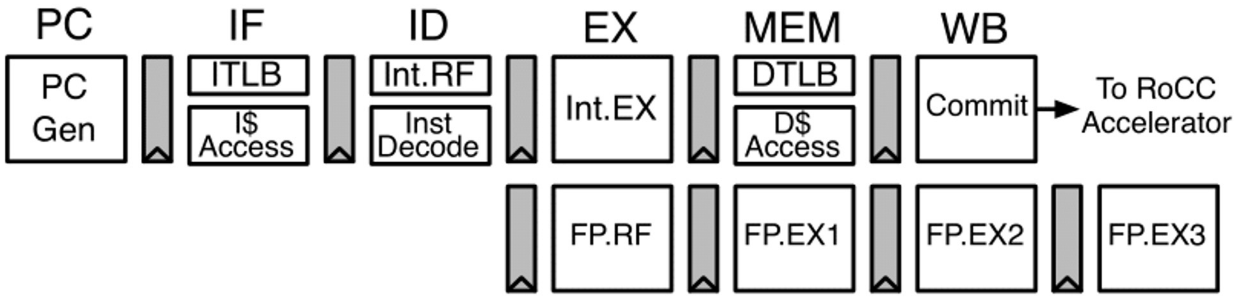
Tiles and Cores



Tiles:

- Each Tile contains a RISC-V core and private caches
- Several varieties of Cores supported
- Interface supports integrating your own RISC-V core implementation

Rocket and BOOM

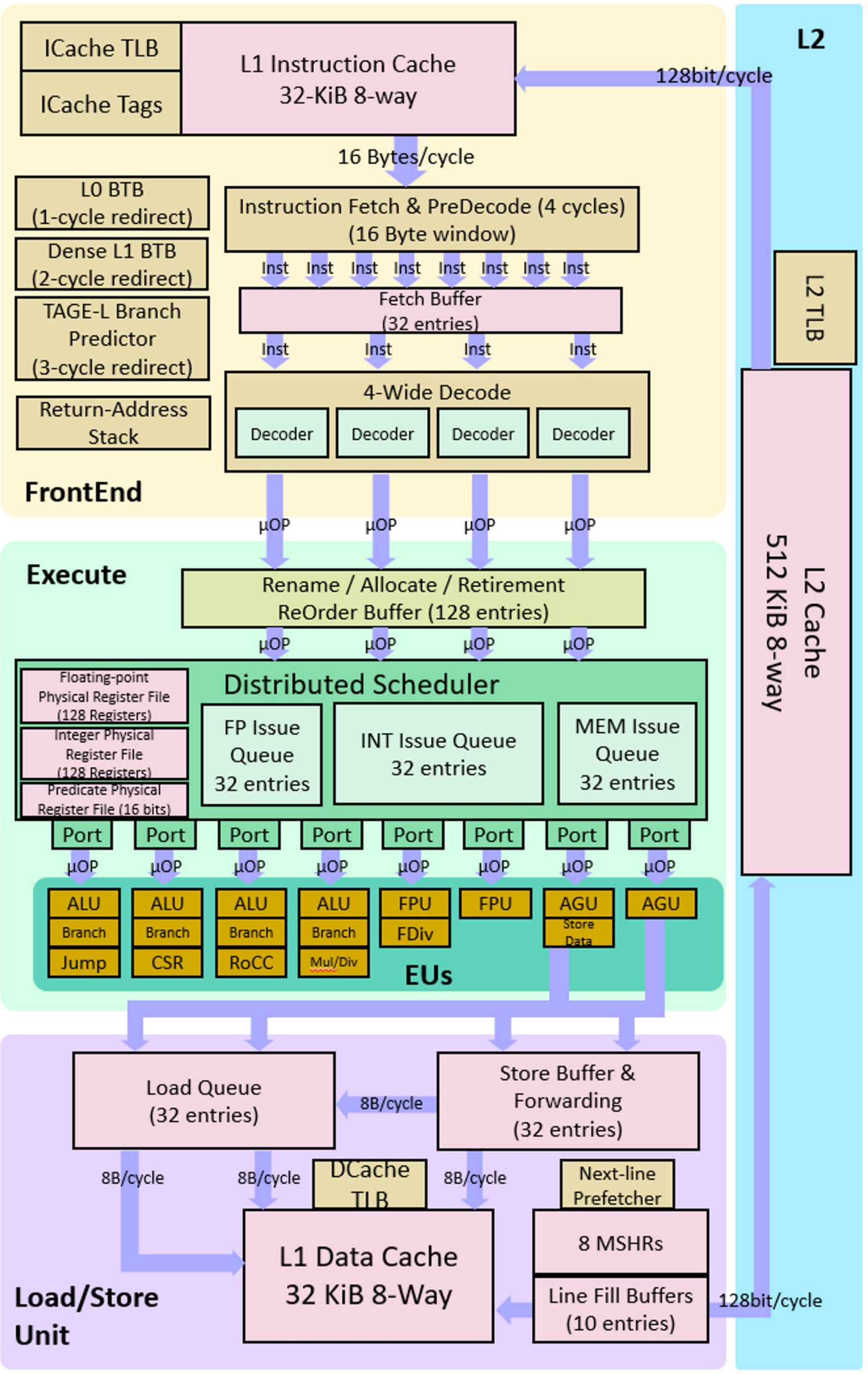


Rocket:

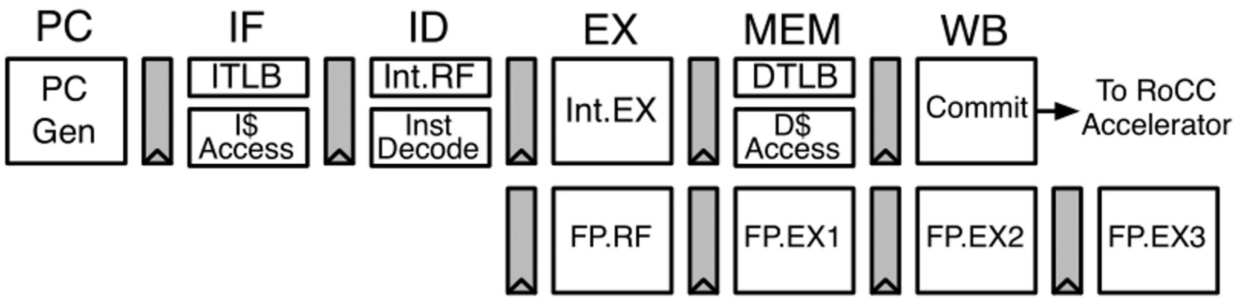
- First open-source RISC-V CPU
- In-order, single-issue RV64GC core
- Efficient design point for low-power devices

SonicBOOM:

- Superscalar out-of-order RISC-V CPU
- Advanced microarchitectural features to maximize IPC
- TAGE branch prediction, OOO load-store-unit, register renaming
- High-performance design point for general-purpose

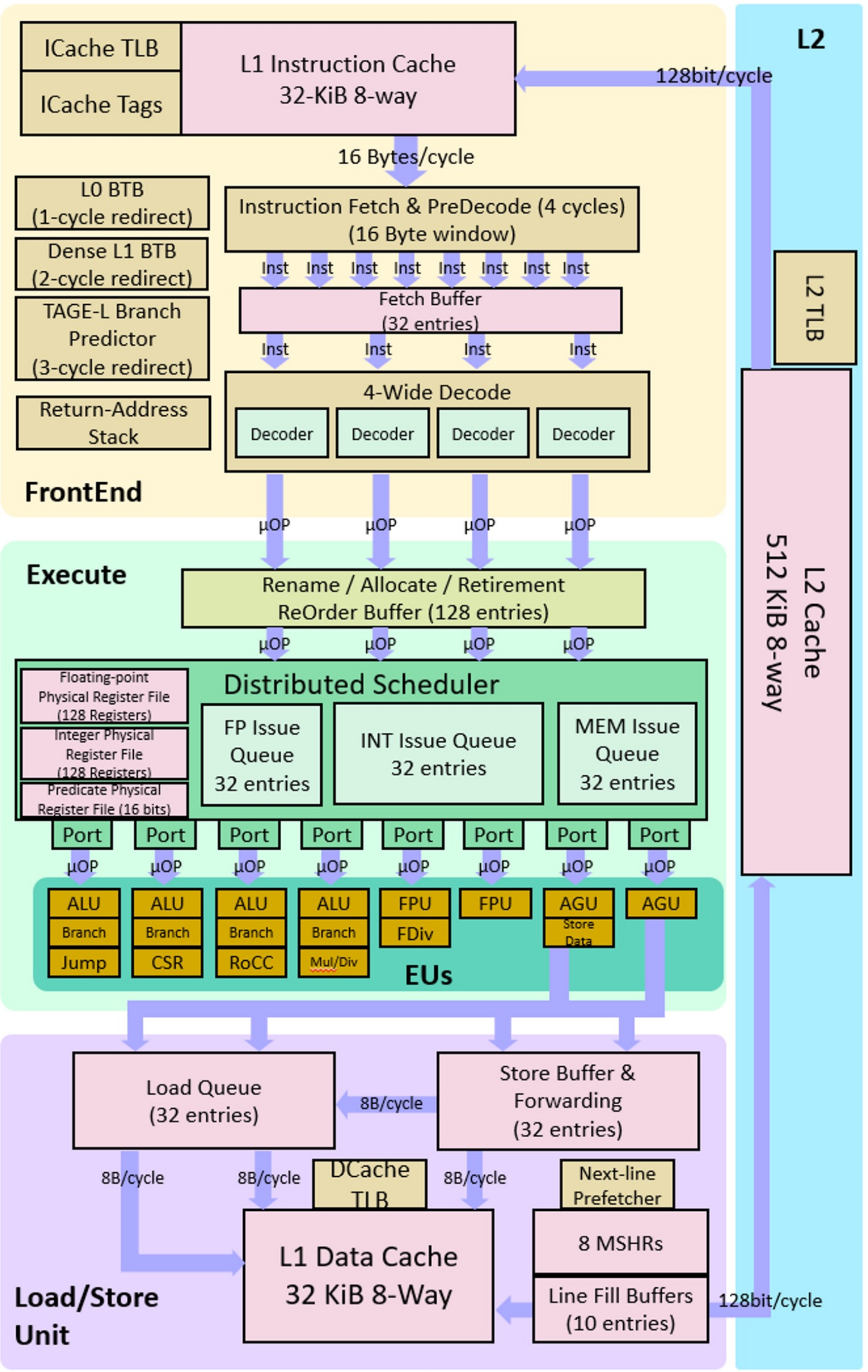

ICache TLB
ICache Tags
L1 Instruction Cache 32-KiB 8-way
128bit/cycle
L2
16 Bytes/cycle
L0 BTB (1-cycle redirect)
Dense L1 BTB (2-cycle redirect)
TAGE-L Branch Predictor (3-cycle redirect)
Return-Address Stack
Instruction Fetch & PreDecode (4 cycles) (16 Byte window)
Inst
Fetch Buffer (32 entries)
4-Wide Decode
Decoder
FrontEnd
μOP
L2 TLB
L2 Cache 512 KiB 8-way
Execute
Rename / Allocate / Retirement ReOrder Buffer (128 entries)
Distributed Scheduler
Floating-point Physical Register File (128 Registers)
Integer Physical Register File (128 Registers)
Predicate Physical Register File (16 bits)
FP Issue Queue 32 entries
INT Issue Queue 32 entries
MEM Issue Queue 32 entries
Port
ALU
Branch
Jump
CSR
RoCC
Mul/Div
FPU
FDiv
AGU
Store Data
EUs
Load Queue (32 entries)
8B/cycle
Store Buffer & Forwarding (32 entries)
DCache TLB
Next-line Prefetcher
L1 Data Cache 32 KiB 8-Way
8 MSHRs
Line Fill Buffers (10 entries)
128bit/cycle
Load/Store Unit

Rocket and BOOM



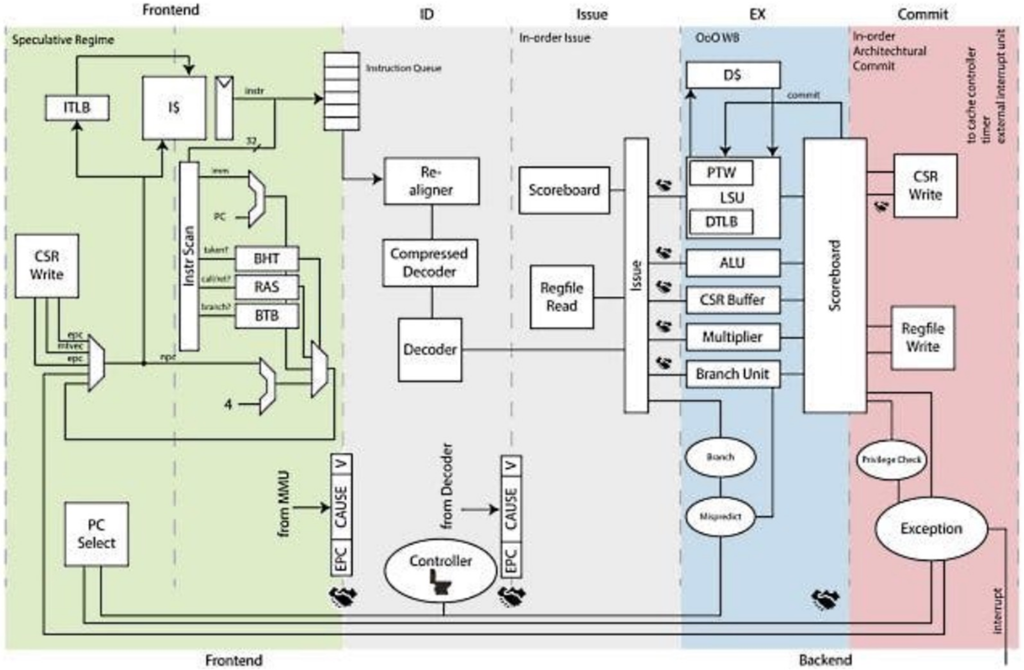
Rocket and SonicBOOM:

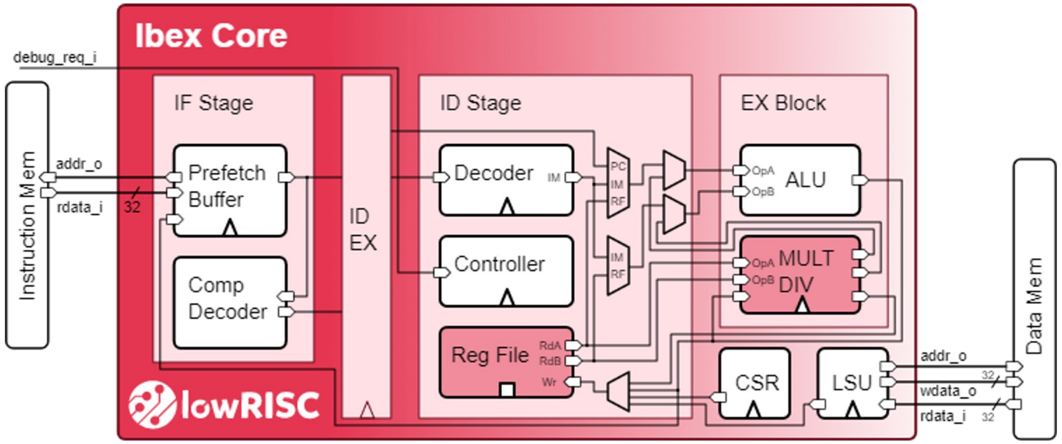
- Support RV64GC ISA profile
- Boots off-the-shelf RISC-V Linux distros (buildroot, Fedora, etc.)
- Fully synthesizable, tapeout-proven
- Described in Chisel
- Fully open-sourced



L2
ICache TLB
ICache Tags
L1 Instruction Cache 32-KiB 8-way
128bit/cycle
16 Bytes/cycle
L0 BTB (1-cycle redirect)
Dense L1 BTB (2-cycle redirect)
TAGE-L Branch Predictor (3-cycle redirect)
Return-Address Stack
Instruction Fetch & PreDecode (4 cycles) (16 Byte window)
Inst
Fetch Buffer (32 entries)
4-Wide Decode
Decoder
FrontEnd
µOP
L2 TLB
L2 Cache 512 KiB 8-way
Execute
Rename / Allocate / Retirement ReOrder Buffer (128 entries)
Floating-point Physical Register File (128 Registers)
Integer Physical Register File (128 Registers)
Predicate Physical Register File (16 bits)
Distributed Scheduler
FP Issue Queue 32 entries
INT Issue Queue 32 entries
MEM Issue Queue 32 entries
Port
ALU
Branch
Jump
CSR
RoCC
Mul/Div
FPU
FDiv
AGU
Store Data
EUs
Load Queue (32 entries)
8B/cycle
Store Buffer & Forwarding (32 entries)
DCache TLB
Next-line Prefetcher
L1 Data Cache 32 KiB 8-Way
8 MSHRs
Line Fill Buffers (10 entries)
128bit/cycle
Load/Store Unit

PULP Cores in CHIPYARD





CVA6 (Formerly Ariane):

- RV64IMAC 6-stage single-issue in-order core
- Open-source
- Implemented in SystemVerilog
- Developed at ETH Zurich as part of PULP,
- Now maintained by OpenHWGroup

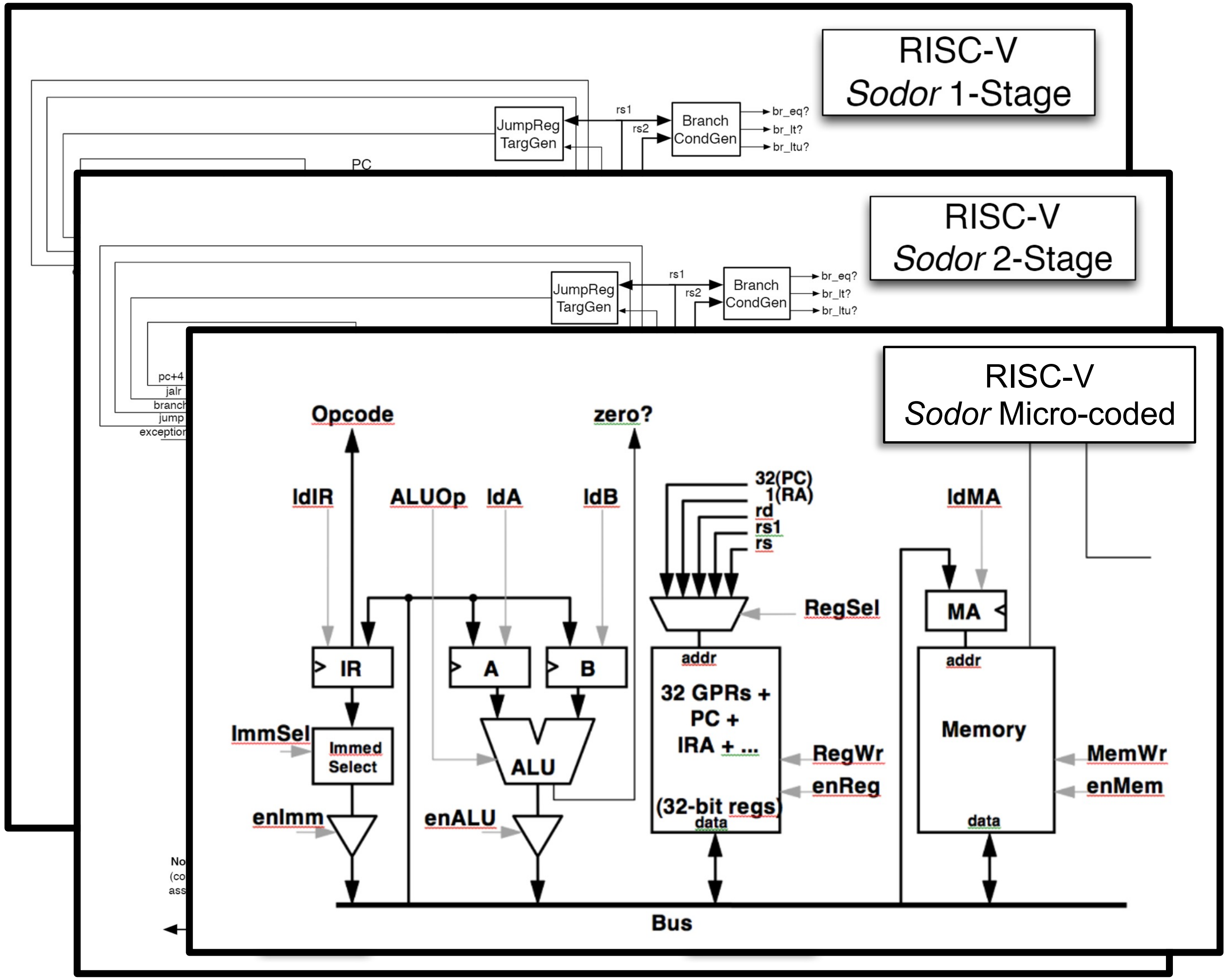
Ibex (Formerly Zero-RISCY):

- RV64IMC 2-stage single-issue in-order core
- Open-source
- Implemented in SystemVerilog
- Developed at ETH Zurich as part of PULP
- Now maintained by lowRISC

Sodor Education Cores

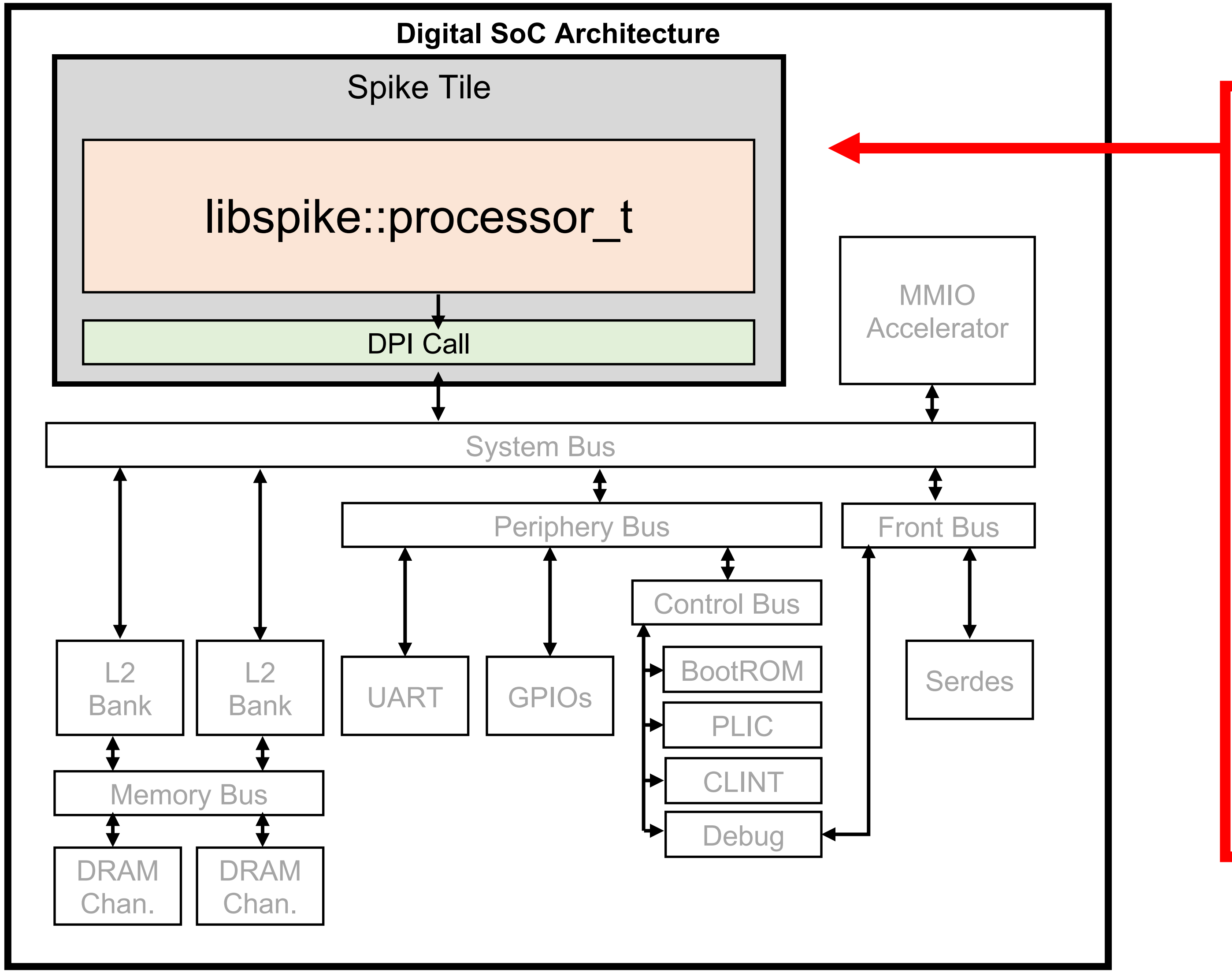
Sodor Core Collection

- Collection of RV32IM cores for teaching and education
- 1-stage, 2-stage, 3-stage, 5-stage implementations
- Micro-coded “bus-based” implementation
- Used in introductory computer architecture courses at Berkeley



New in Chipyard 1.9.0

Spike –as –a- Tile



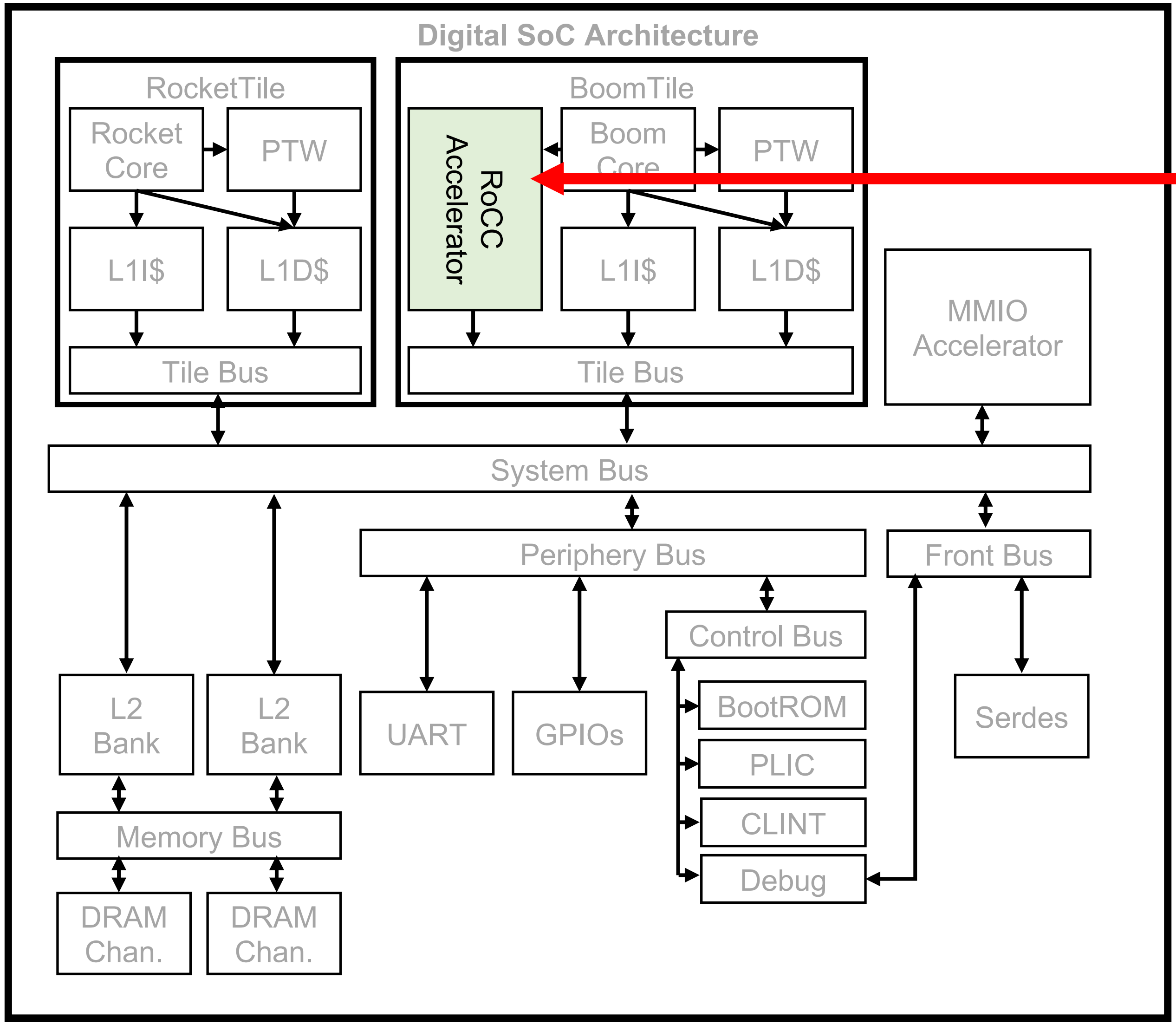
Spike:

- Open-source RISC-V ISA simulator
- Fast, extensible, C++ functional model

Spike-as-a-Tile:

- DPI interface between RTL SoC simulation and SoC functional model
- Spike “virtual platform”
- Enables testing of complex device software in RTL simulation

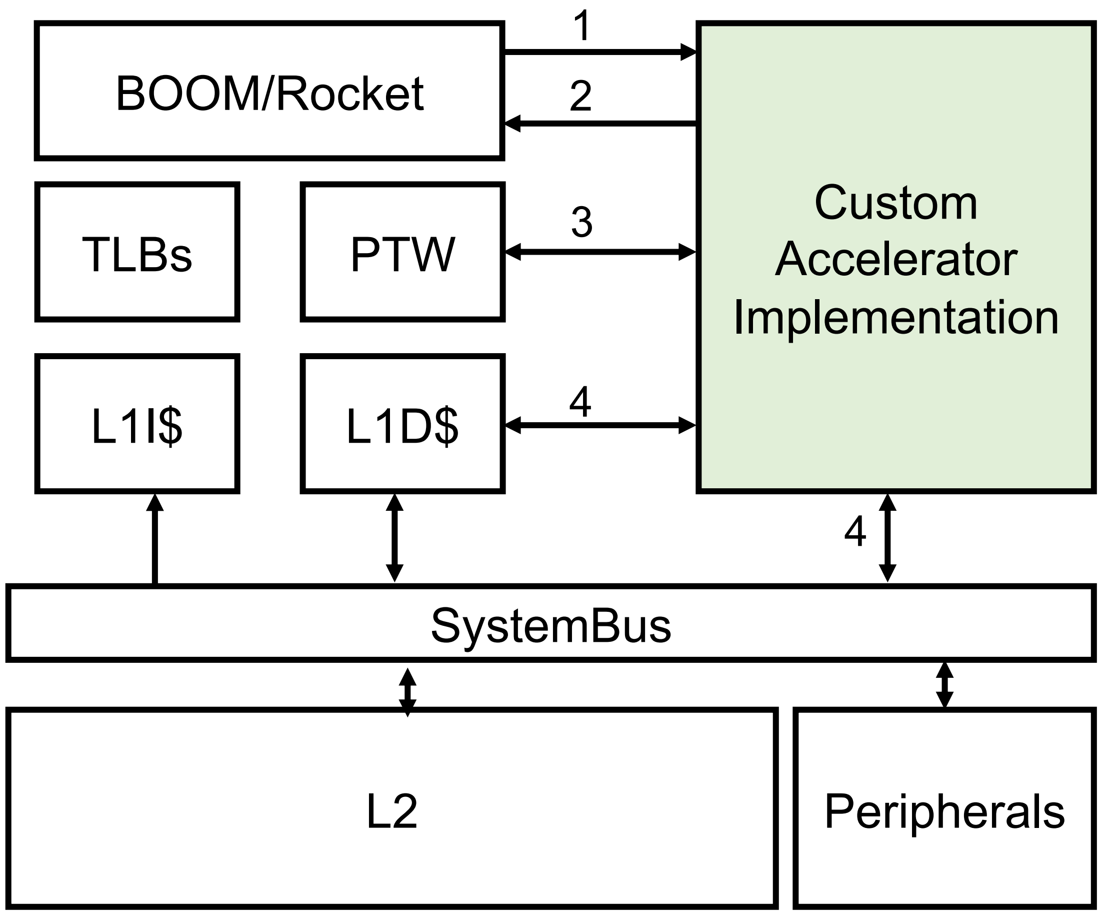
RoCC Accelerators



RoCC Accelerators:

- Tightly-coupled accelerator interface
- Attach custom accelerators to Rocket or BOOM cores

RoCC Accelerators



1. Core automatically decodes + sends custom instructions to accelerator

2. Accelerator can write back into core registers

3. Accelerator can support virtual-addressing by sharing core PTW/TLB

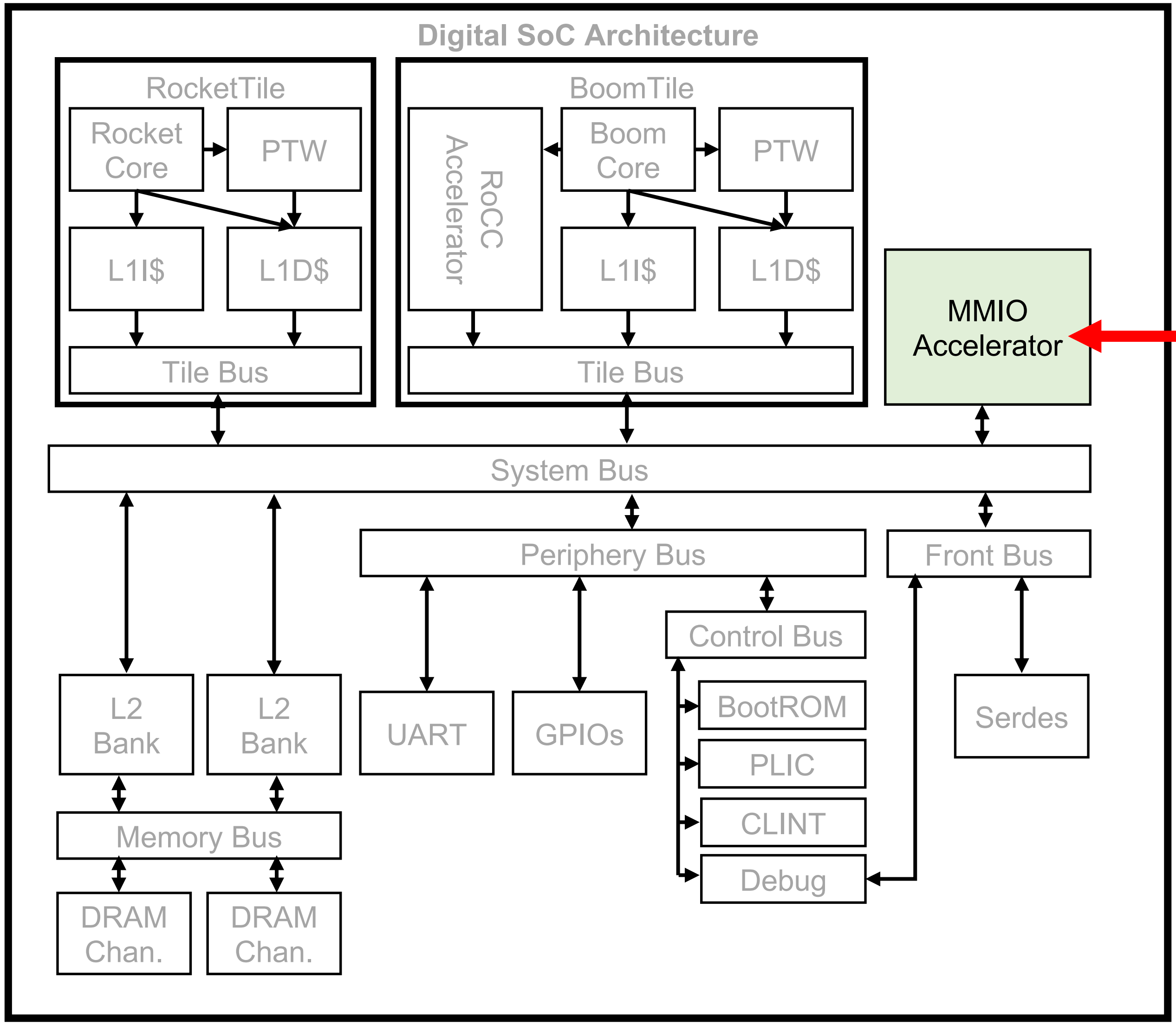
4. Accelerator can fetch-from/write-to coherent L1 data cache or outer-memory

Flexible interface supports a variety of accelerator designs

Included in Chipyard:

- Gemmini ML accelerator
- Hwacha vector accelerator
- SHA3 accelerator

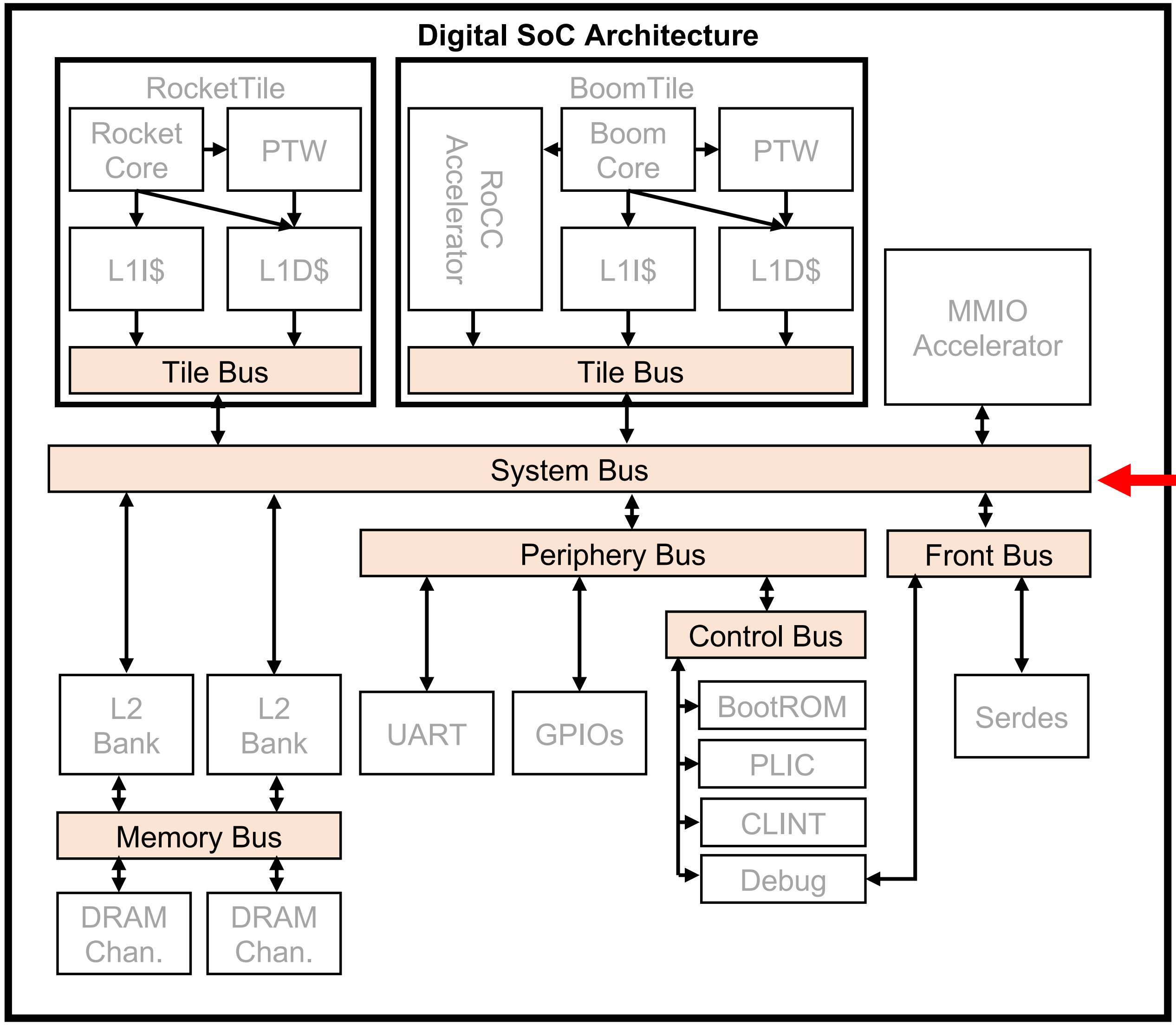
MMIO Accelerators



MMIO Accelerators:

- Controlled by MMIO-mapped registers
- Supports DMA to memory system
- Examples:
  - Nvidia NVDLA accelerator
  - FFT accelerator generator

Coherent Interconnect



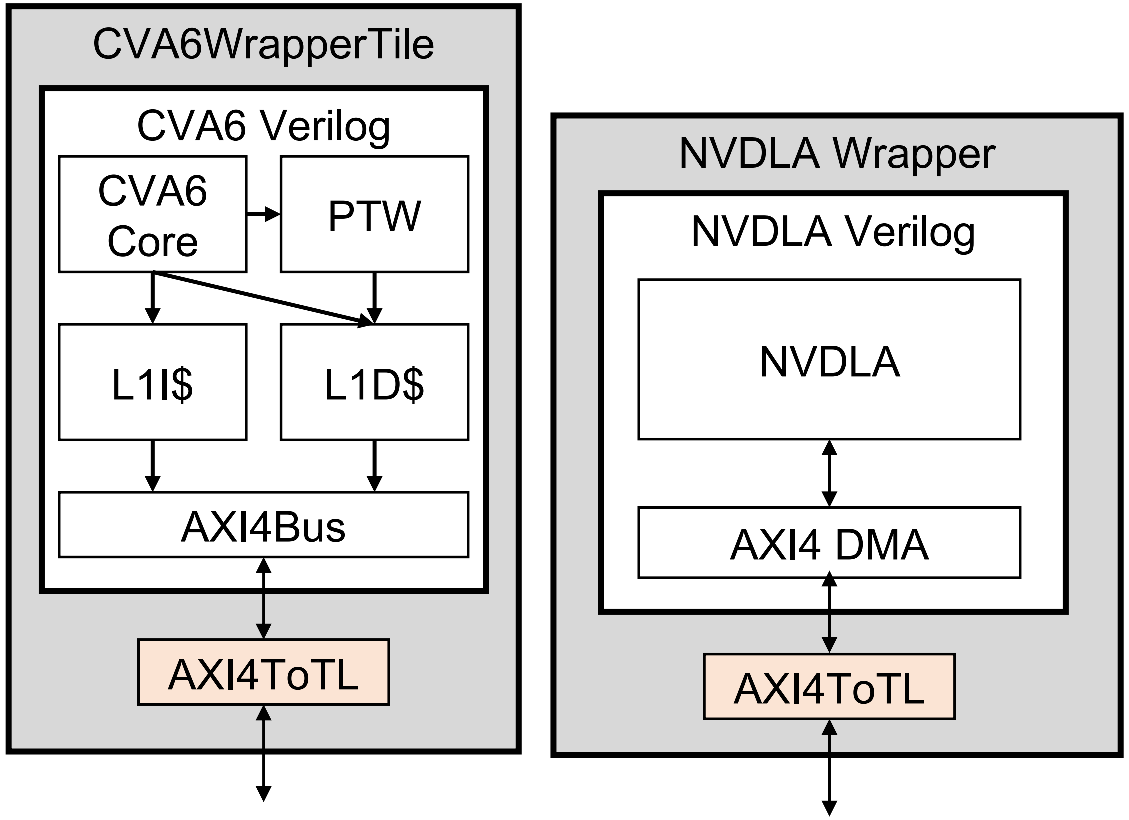
TileLink Standard:

- TileLink is open-source chip-scale interconnect standard
- Comparable to AXI/ACE
- Supports multi-core, accelerators, peripherals, DMA, etc

Interconnect IP:

- Library of TileLink RTL generators provided in RocketChip
- RTL generators for crossbar-based buses
- Width-adapters, clock-crossings, etc.
- Adapters to AXI4, APB

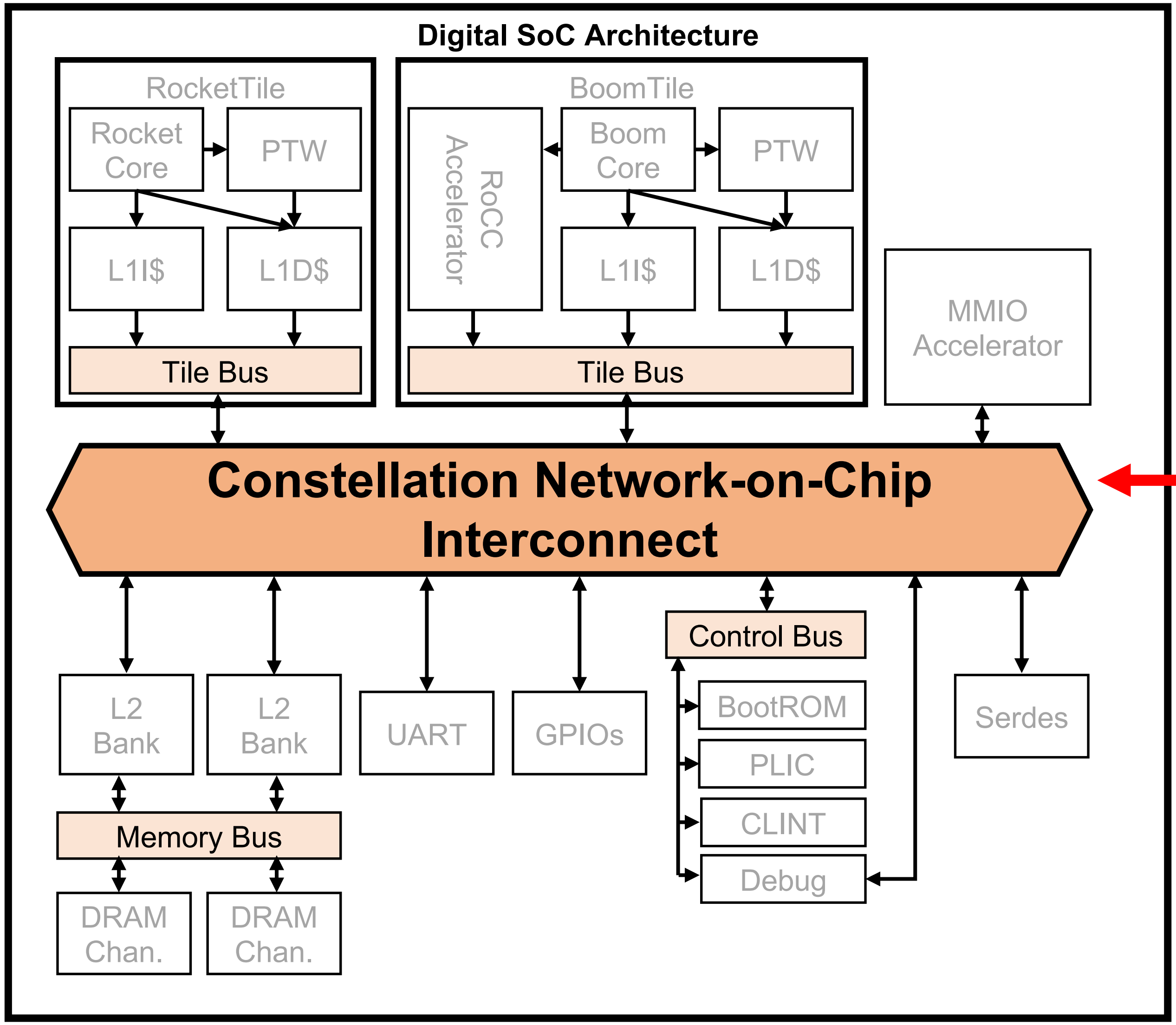
Protocol Shims



AMBA-to-TileLink shims enable easy integration with existing IP

- Works for cores/peripherals/accelerators
- Drop-in Verilog integration of CVA-6, NVDLA

NoC Interconnect



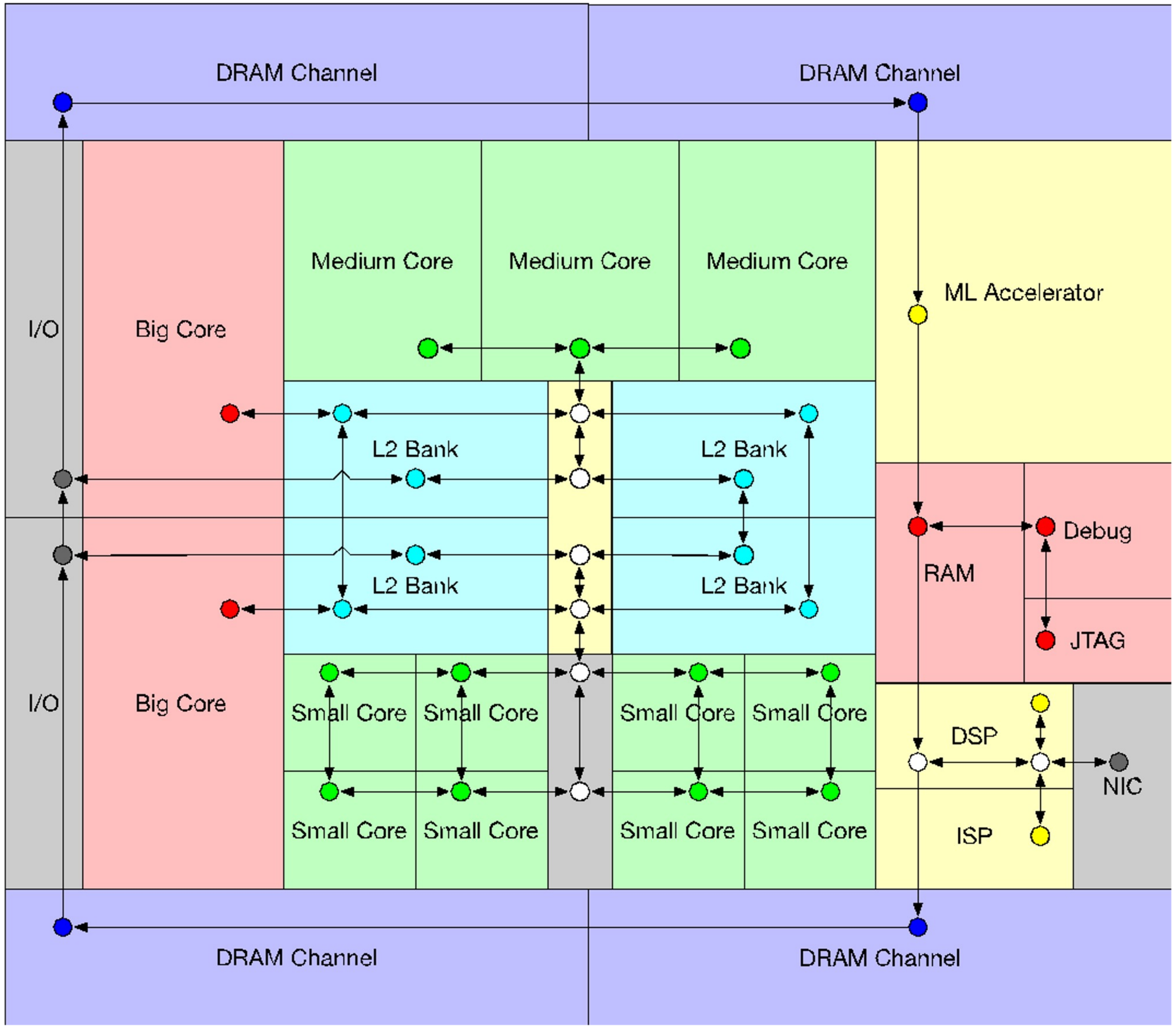
Constellation

- Drop-in replacement for TileLink crossbar buses

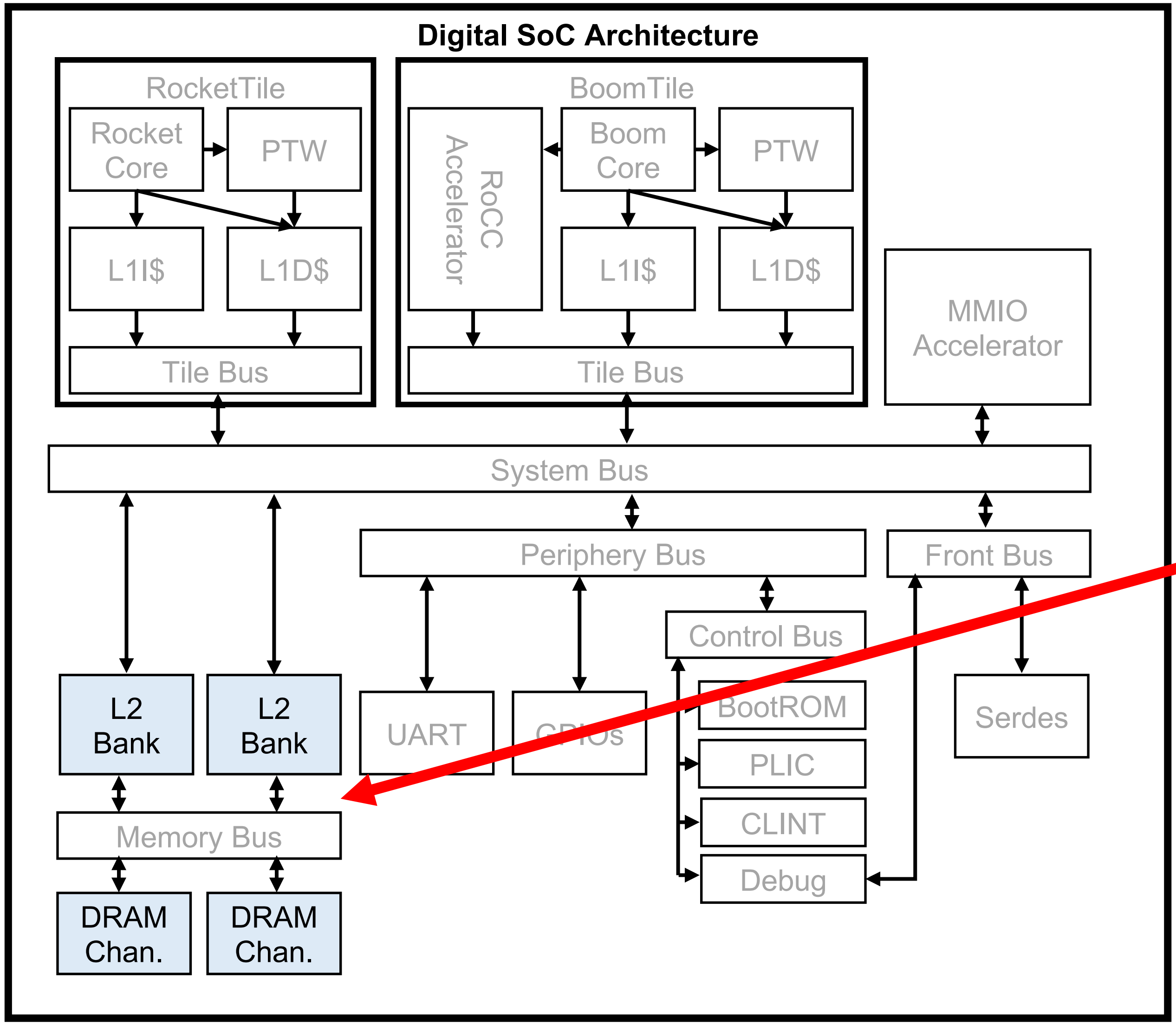
Constellation

A parameterized Chisel generator for SoC interconnects

- Protocol-independent transport layer
- Supports TileLink, AXI-4
- Highly parameterized
- Deadlock-freedom
- Virtual-channel wormhole-routing



L2/DRAM



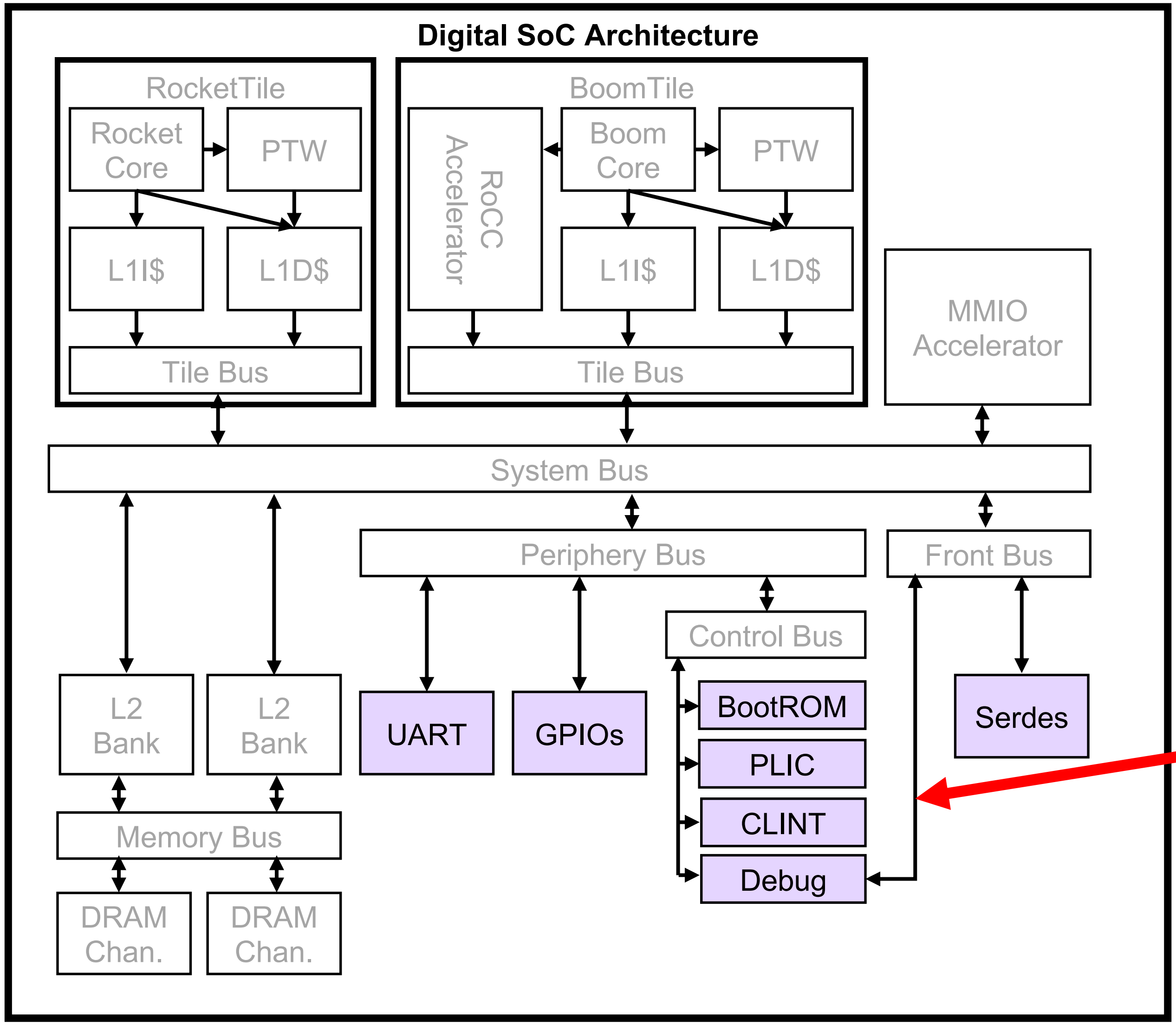
Shared memory:

- Open-source TileLink L2 developed by SiFive
  - Directory-based coherence with MOESI-like protocol
  - Configurable capacity/banking
- Support broadcast-based coherence in no-L2 systems
- Support incoherent memory systems

DRAM:

- AXI-4 DRAM interface to external memory controller
- Interfaces with DRAMSim

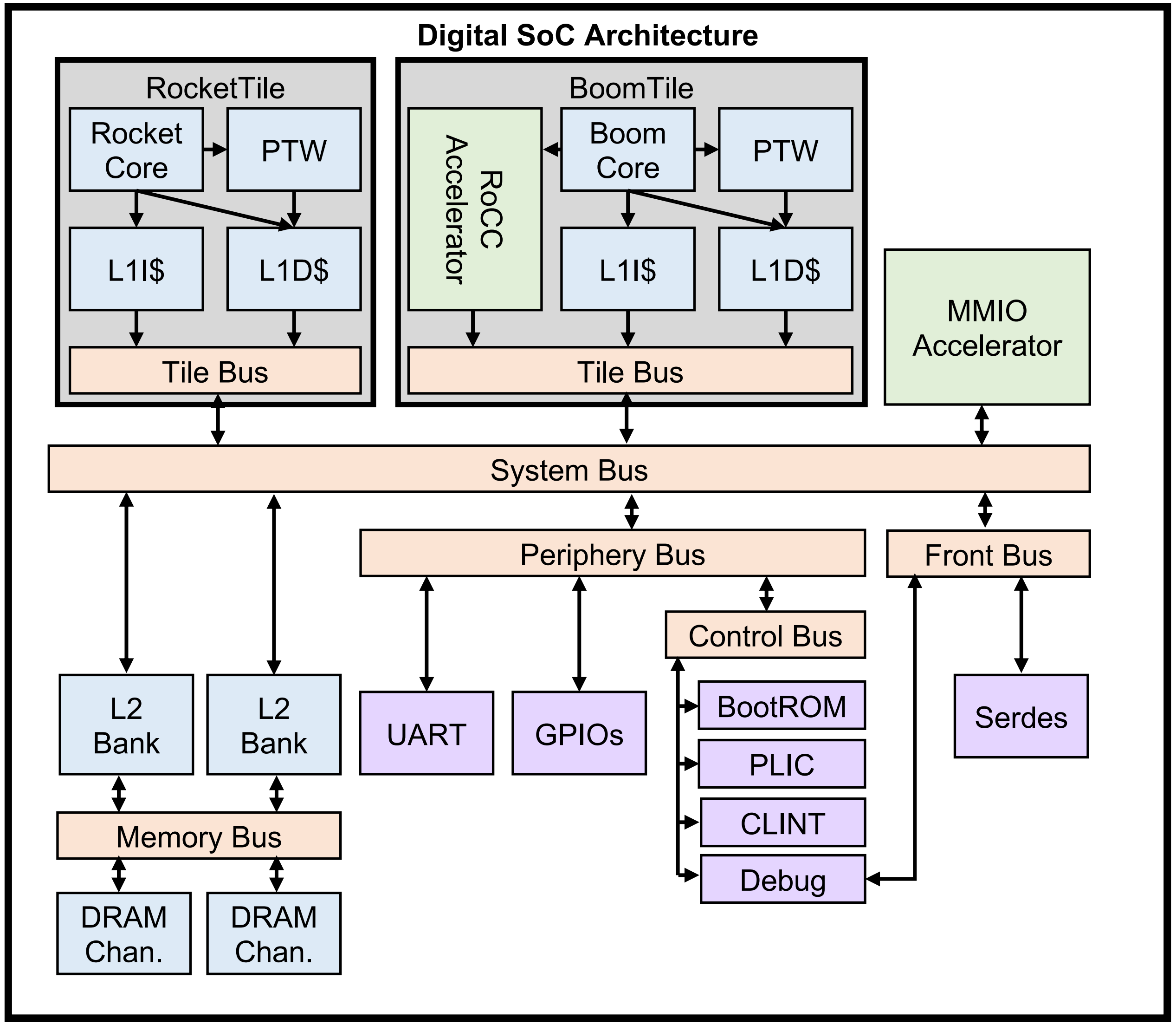
Peripherals and IO



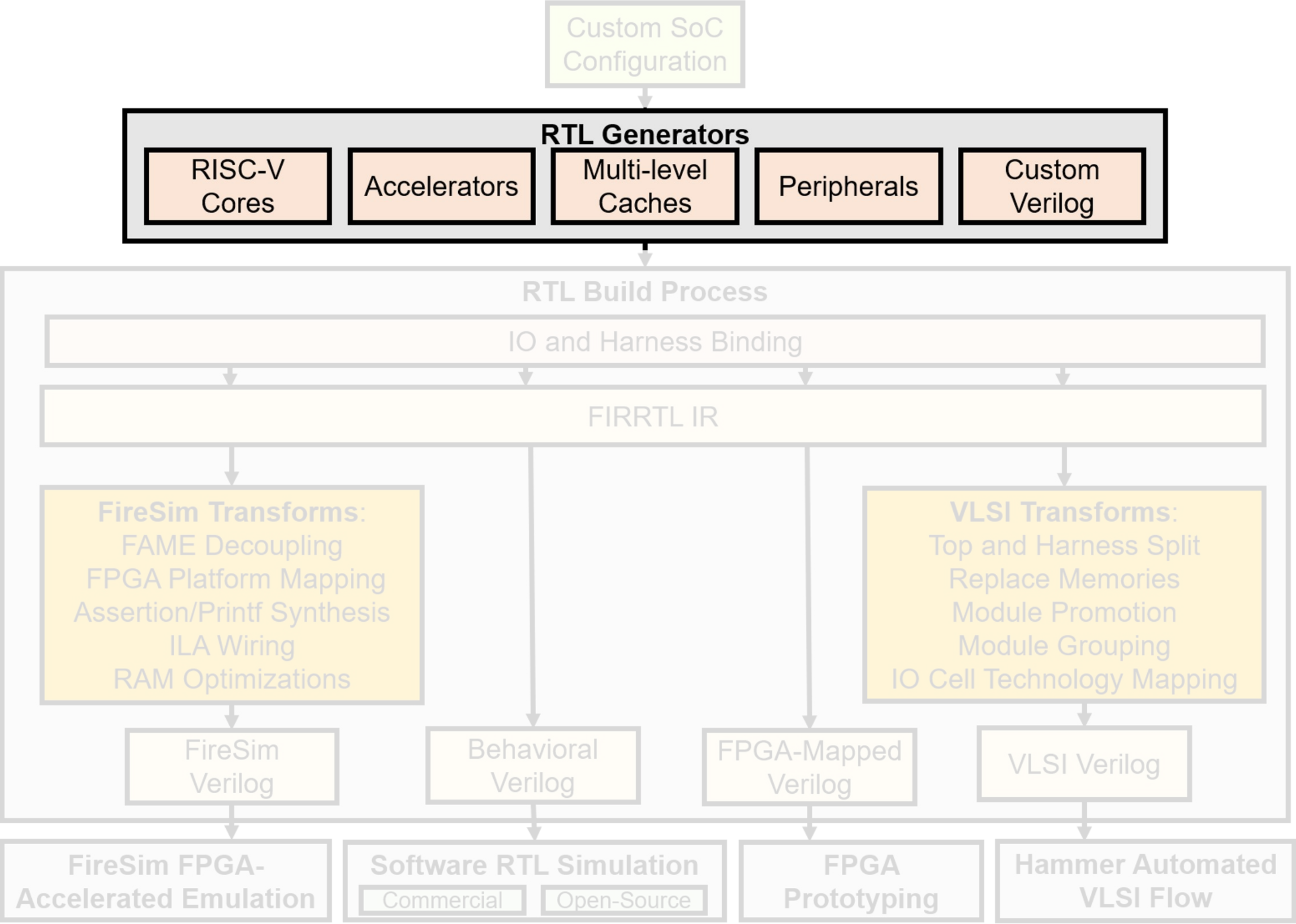
Peripherals and IO:

- Open-source RocketChip blocks
  - Interrupt controllers
  - JTAG, Debug module, BootROM
  - UART, GPIOs, SPI, I2C, PWM, etc.
- TestChipIP: useful IP for test chips
  - Clock-management devices
  - SerDes
  - Scratchpads

SoC Architecture

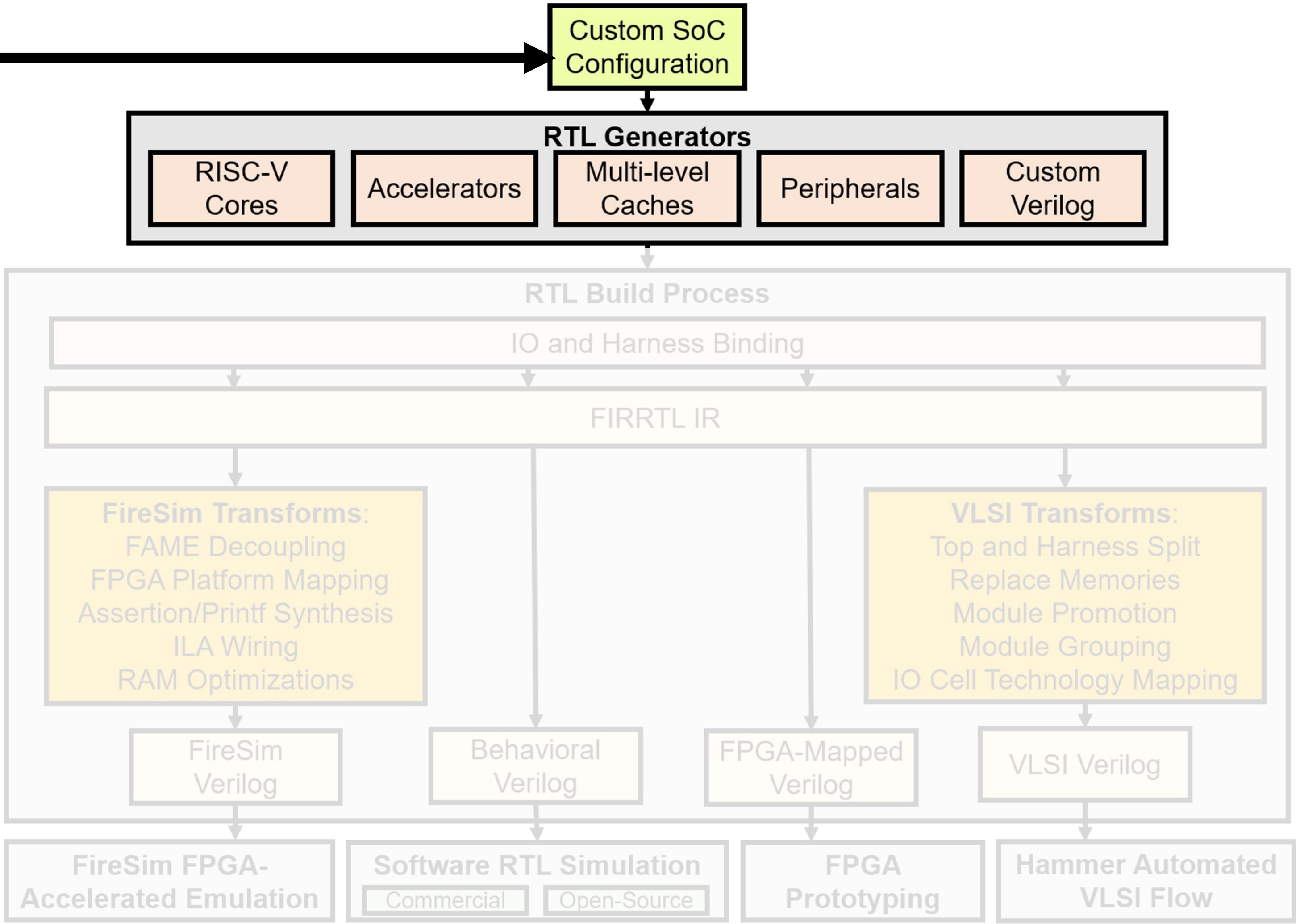


CHIPYARD Organization

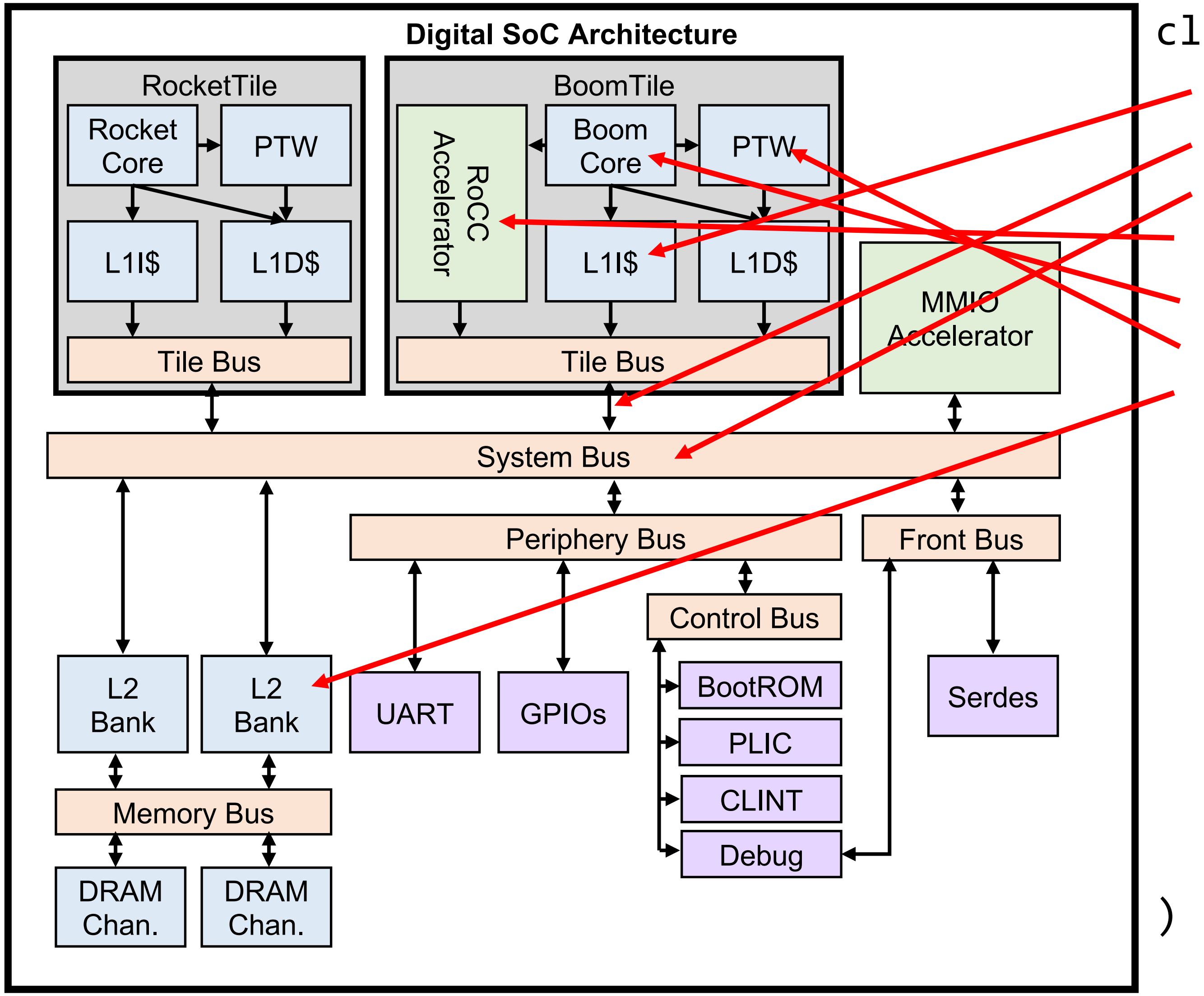


CHIPYARD Organization

SoC Configuration

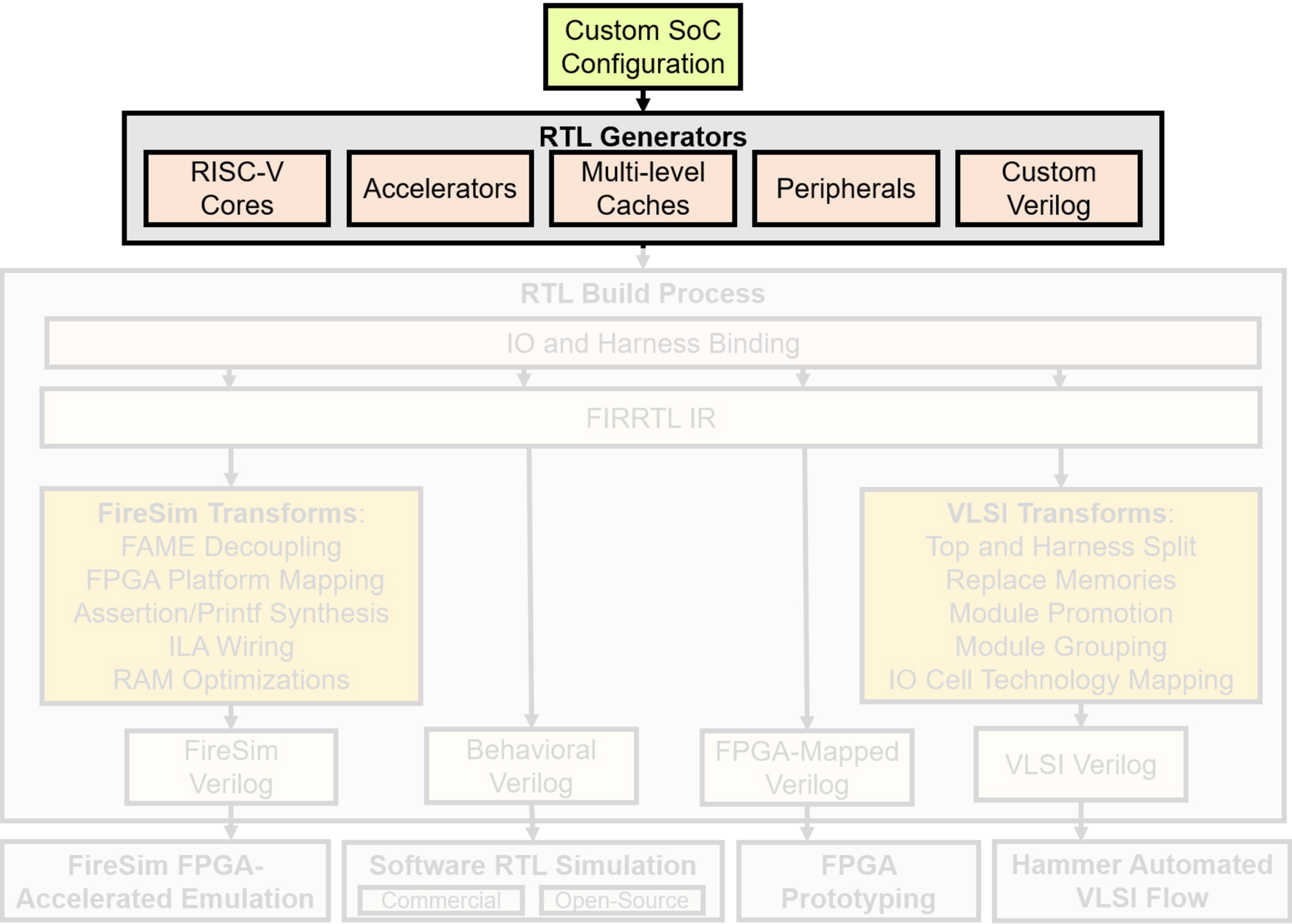


Composable Configurations



```
class CustomConfig extends Config(
  new WithL1CacheWays(4) ++
  new WithAsyncTiles ++
  new WithSystemBusWidth(128) +
  new WithFPGemmini ++
  new With3WideBooms ++
  new WithL2TLBs(512) ++
  new WithL2Sets(1024) ++

  new WithDefaultGemmini ++
  new WithNRocketCores(1) ++
  new WithNBoomCores(1) ++
  new WithBootROM ++
  new WithUART ++
  new WithJtagDTM ++
  new WithGPIOs ++
  new WithInclusiveCache(512) ++
)
```

CHIPYARD Organization



CHIPYARD Organization

SW RTL Simulation:

- RTL-level simulation with Verilator or VCS
- Hands-on tutorial next

FPGA prototyping:

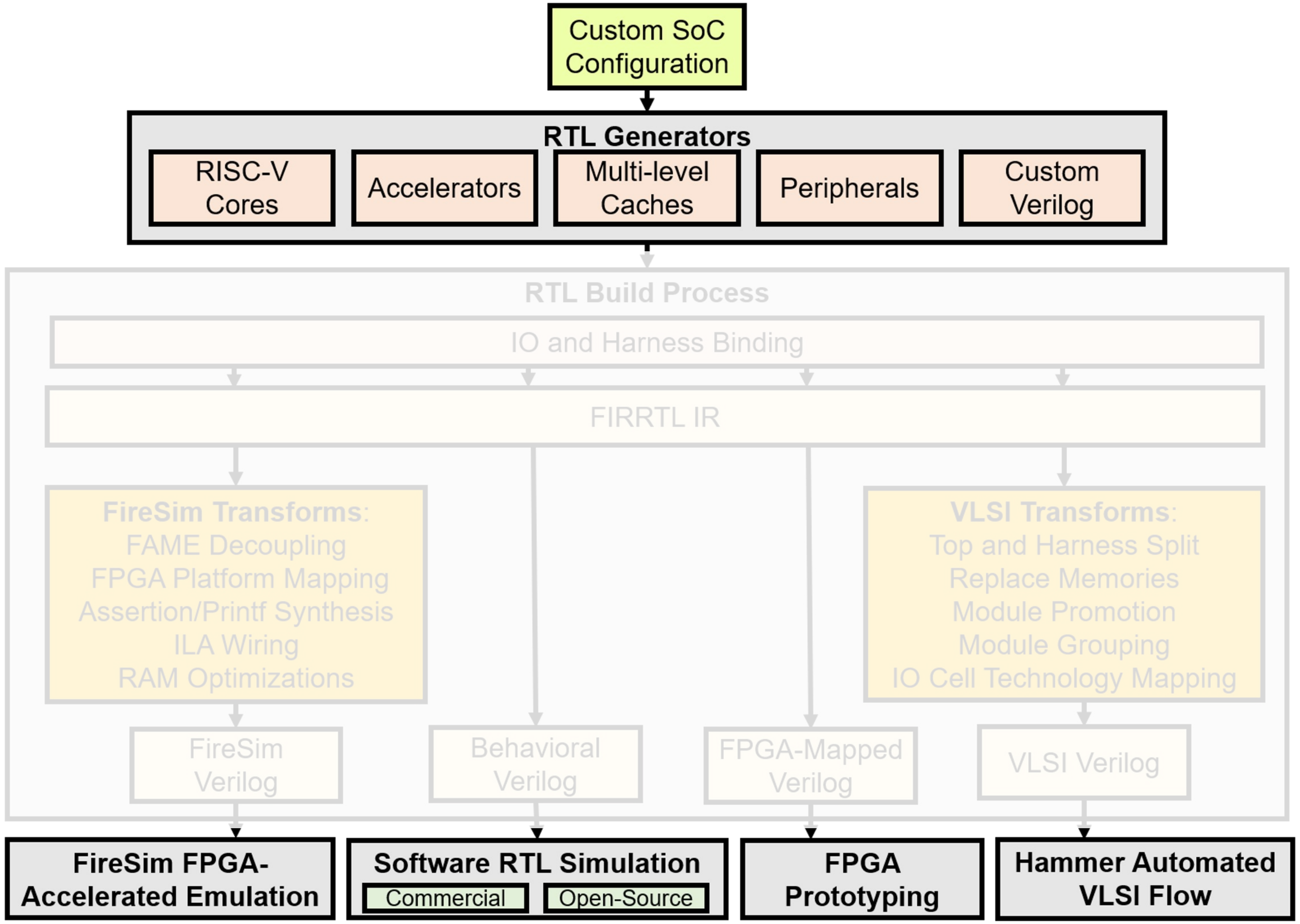
- Fast, non-deterministic prototypes
- Bringup platform for taped-out chips

Hammer VLSI flow:

- Tapeout a custom config in some process technology
- Overview of flow later

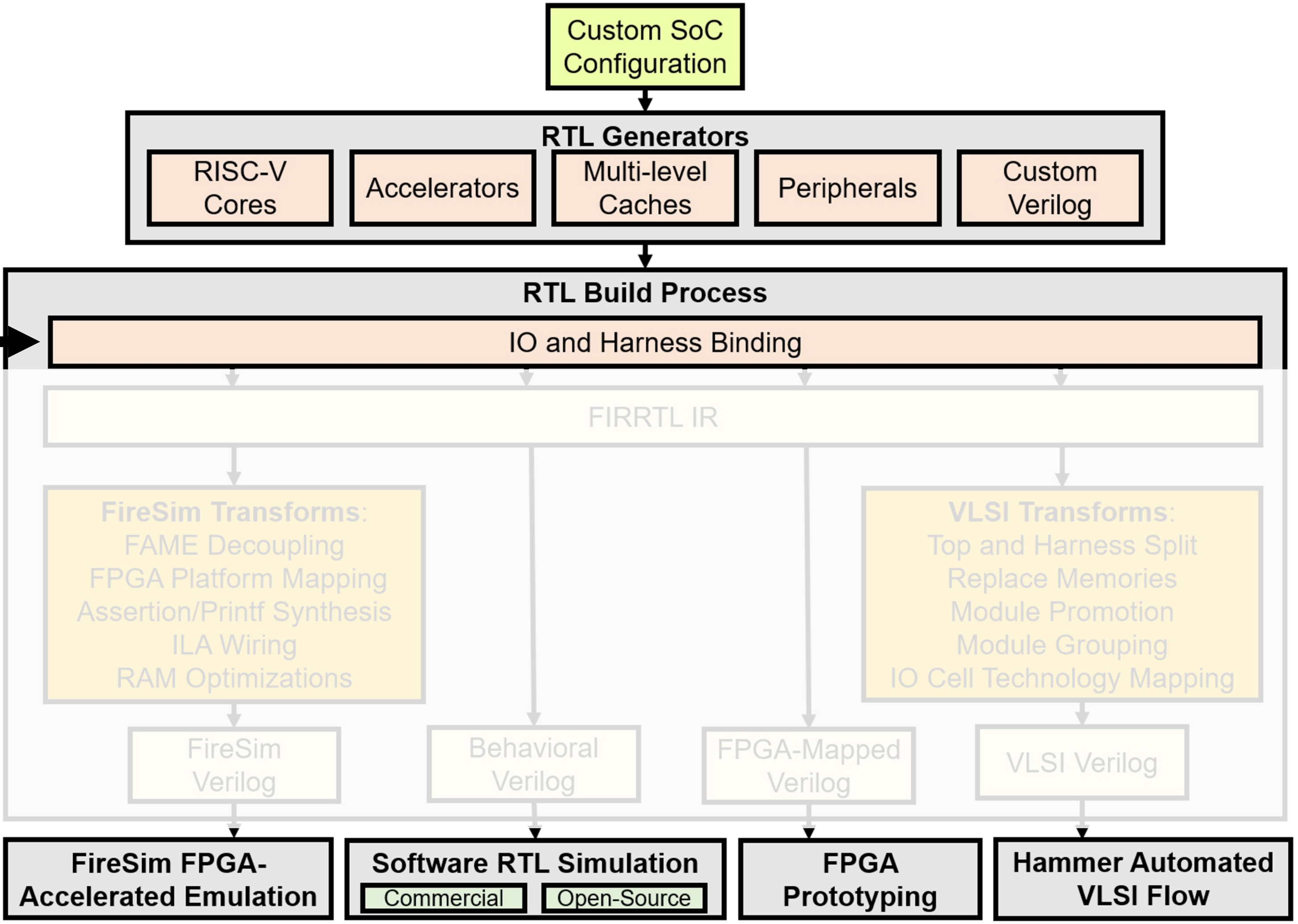
FireSim:

- Fast, accurate FPGA-accelerated simulations
- Hands-on tutorial later

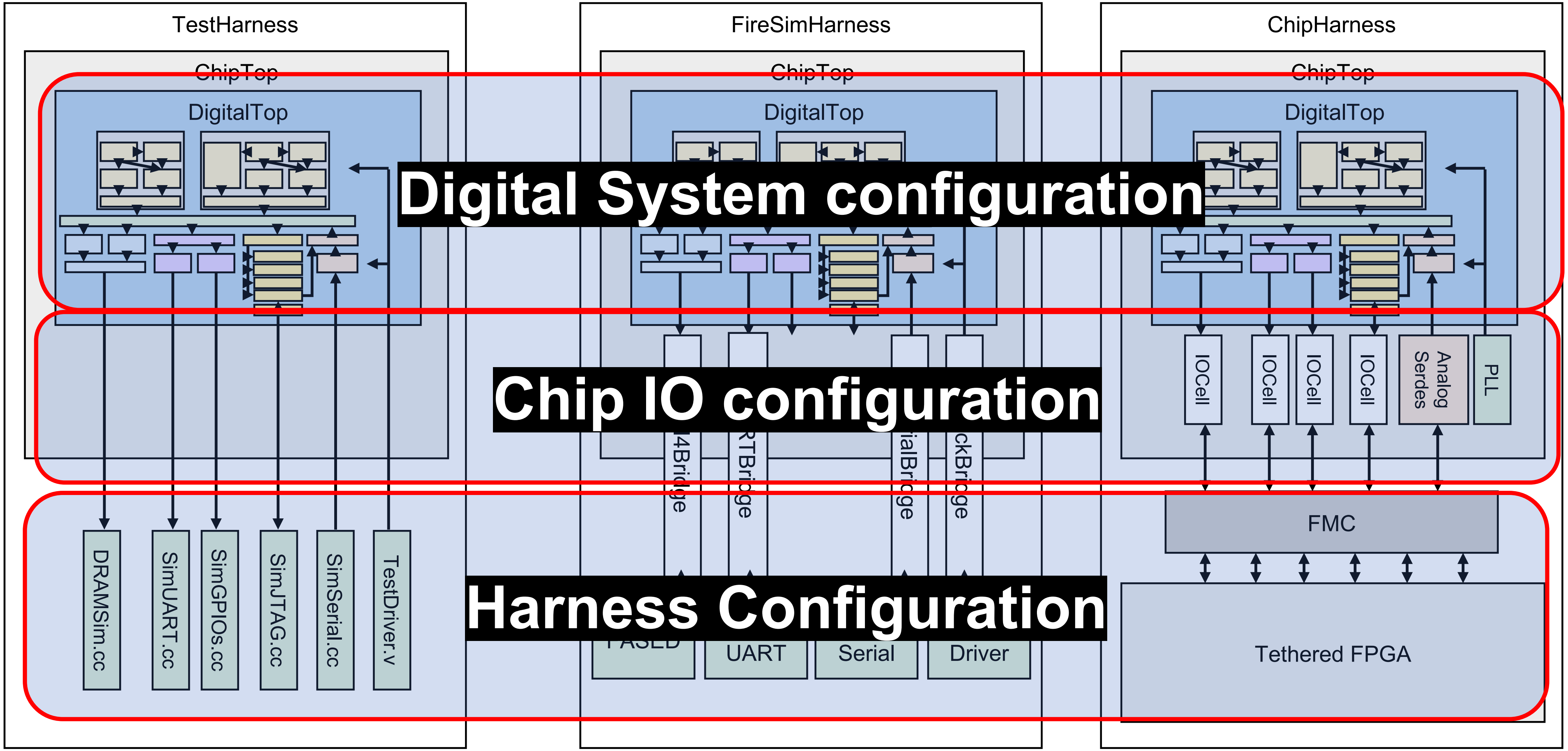


CHIPYARD Organization

IO and Harness configuration



Multipurpose

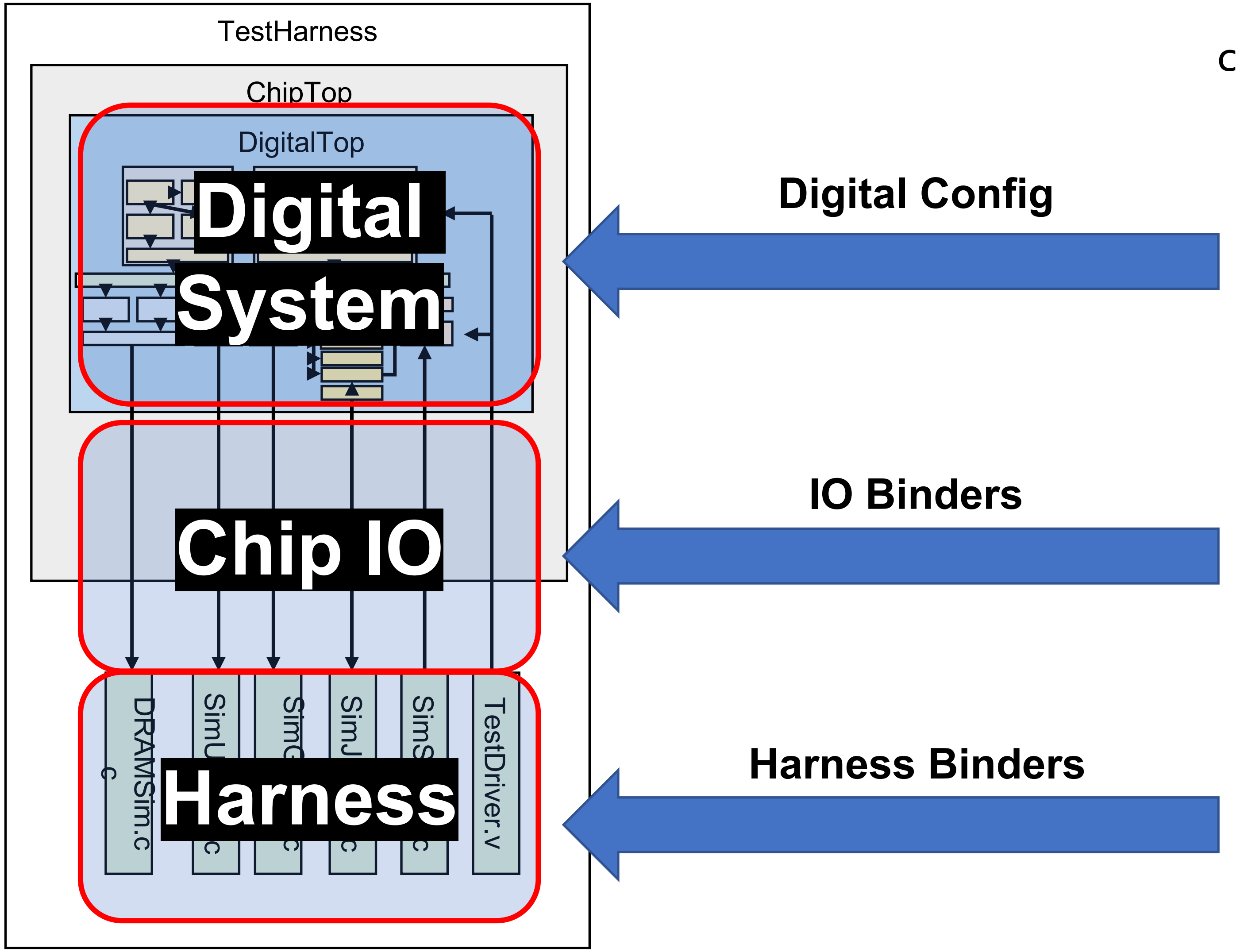


Digital System configuration

Chip IO configuration

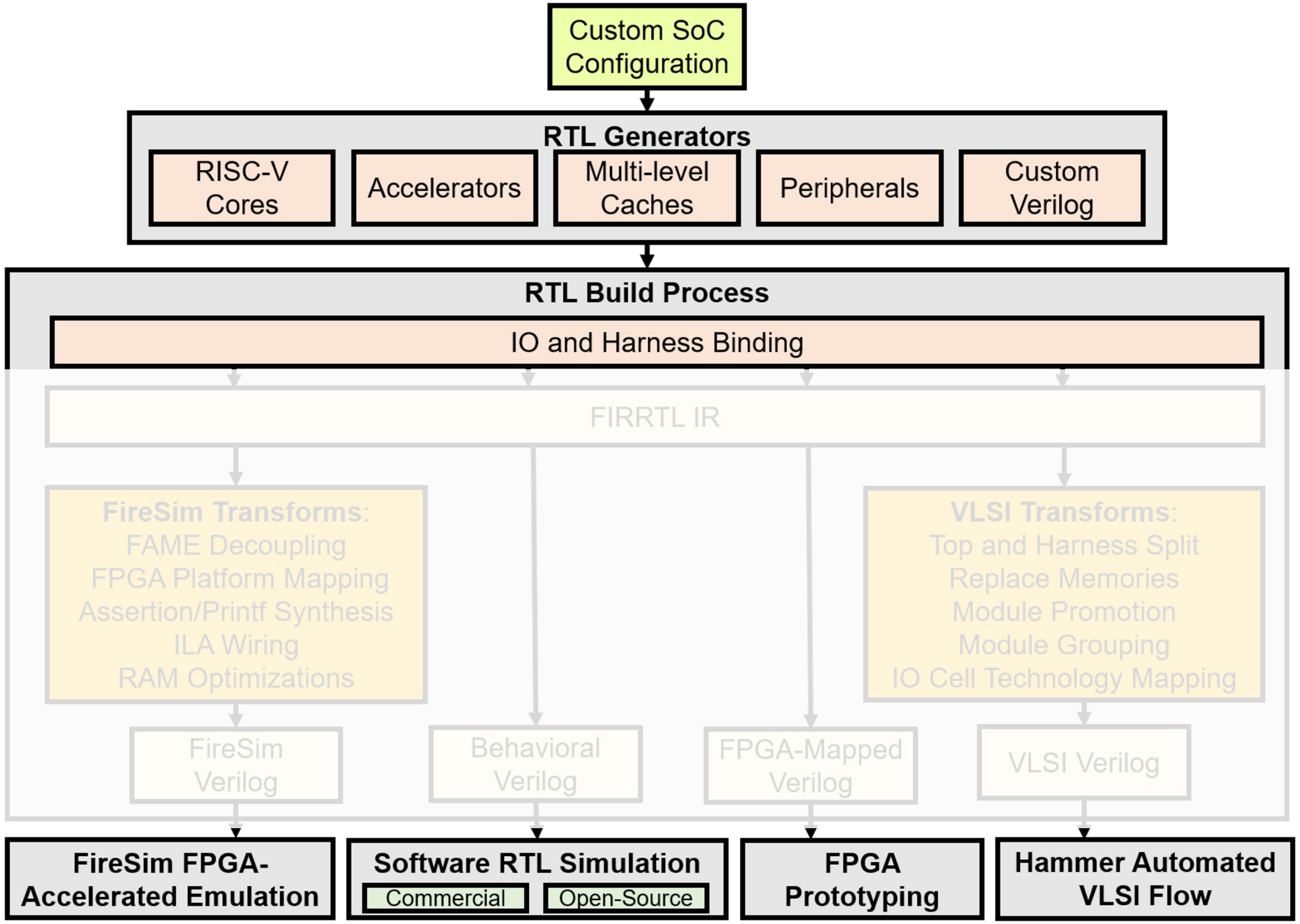
Harness Configuration

Configuring IO + Harness



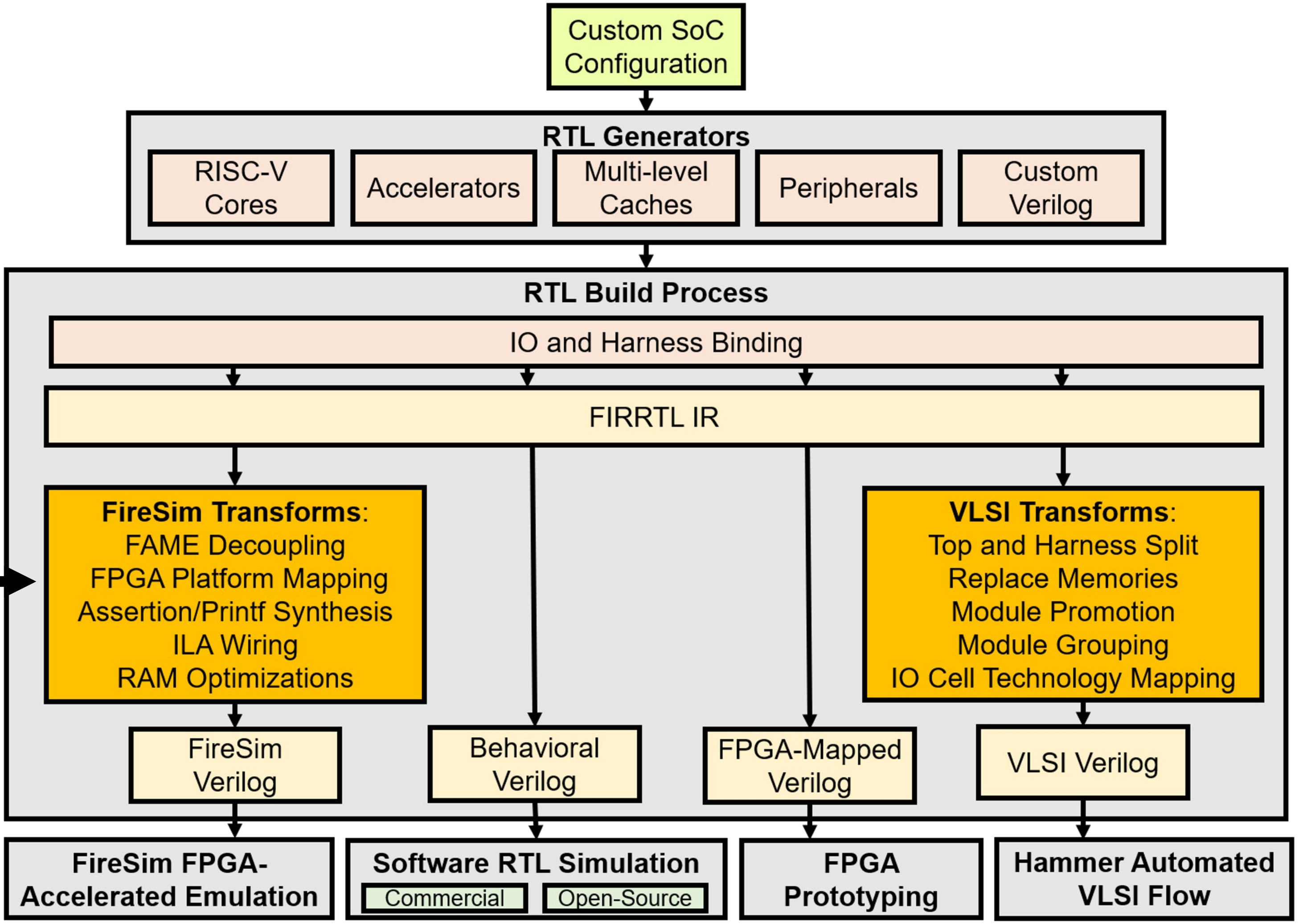
```
class CustomConfig extends Config(
  new WithDefaultGemmini ++
  new WithNRocketCores(1) ++
  new WithNBoomCores(1) ++
  new WithBootROM ++
  new WithUART ++
  new WithJtagDTM ++
  new WithGPIOs ++
  new WithInclusiveCache(512) ++

  new WithIOCellModels ++


  new WithDRAMSim ++
  new WithSimUART ++
  new WithSimJTAG ++
  new WithSimSerial

)
```

CHIPYARD Organization



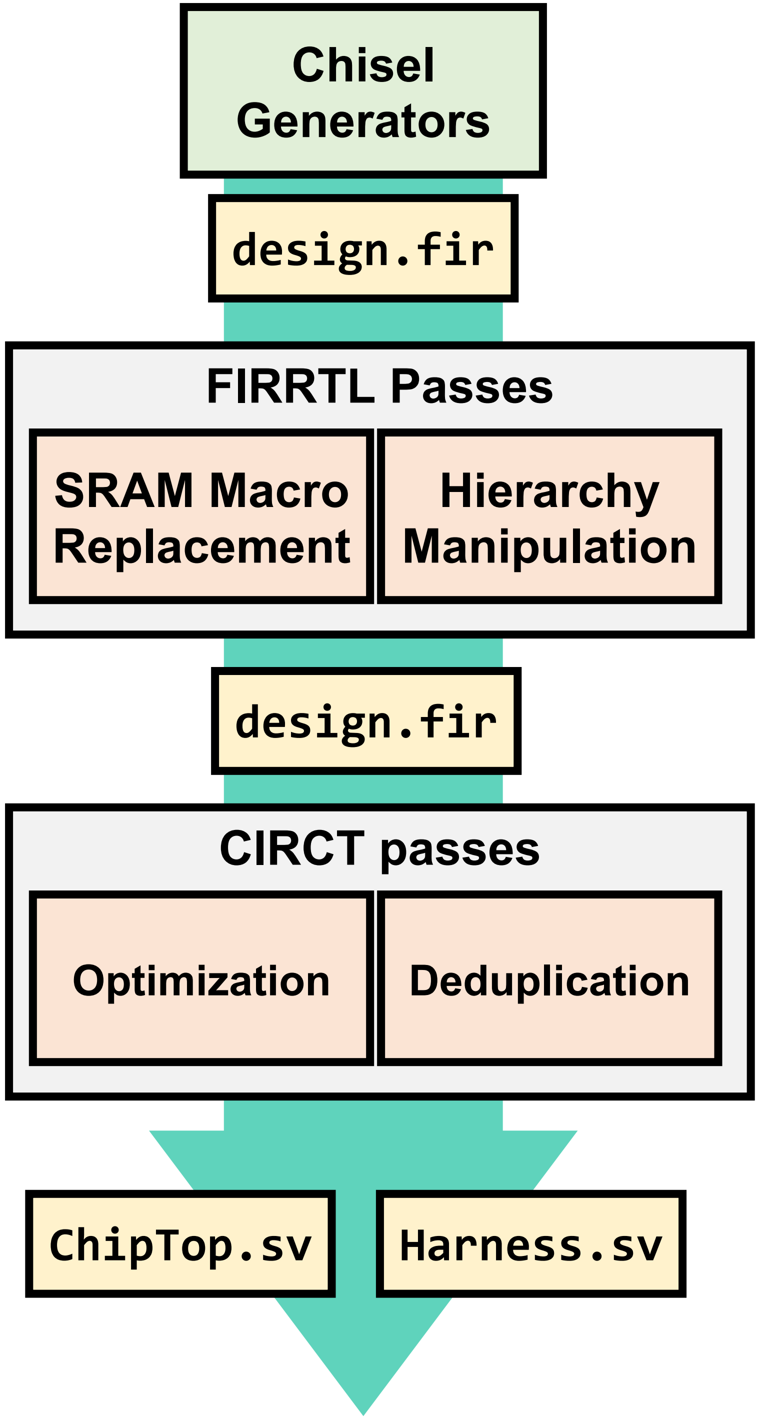
CHIPYARD Organization

FIRRTL/CIRCT Transforms



Elaboration Flow

- **Chisel** programs generate **FIRRTL** representations of hardware
- **FIRRTL passes** transform the target netlist
  - FireSim’s AutoCounter/AutoILA/Printf use FIRRTL passes
  - VLSI flows can use FIRRTL passes to adjust the module hierarchy
- **NEW in Chipyard 1.9.1: CIRCT Backend**
  - CIRCT generates tool-friendly synthesizable Verilog from FIRRTL
  - Very fast/powerful FIRRTL compiler



CHIPYARD Organization

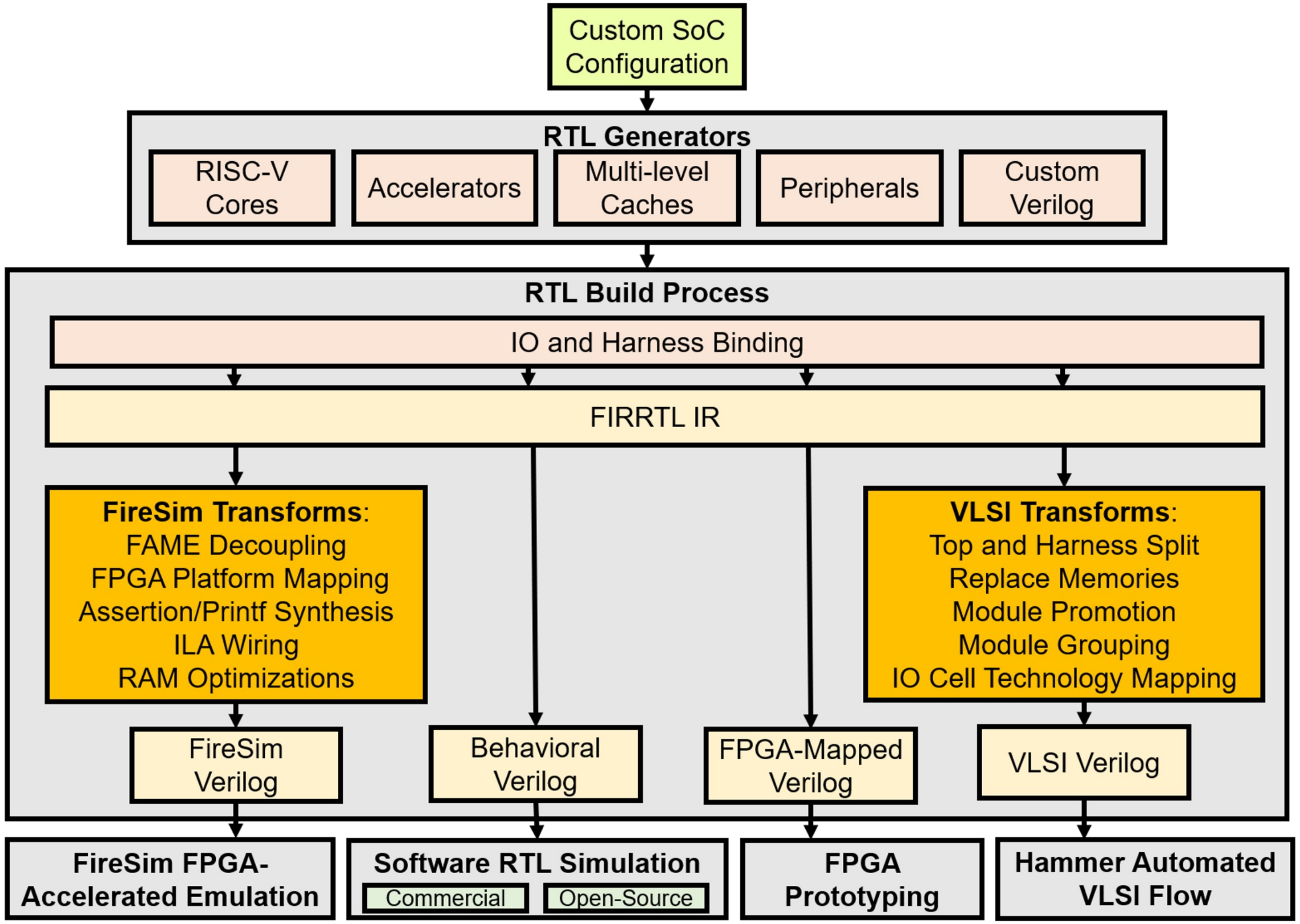
Configs: Describe parameterization of a multi-generator SoC

Generators: Flexible, reusable library of open-source Chisel generators (and Verilog too)

IOBinders/HarnessBinders: Enable configuring IO strategy and Harness features

FIRRTL/CIRCT Passes: Structured mechanism for supporting multiple flows

Target flows: Different use-cases for different types of users



CHIPYARD Learning Curve

Advanced-level

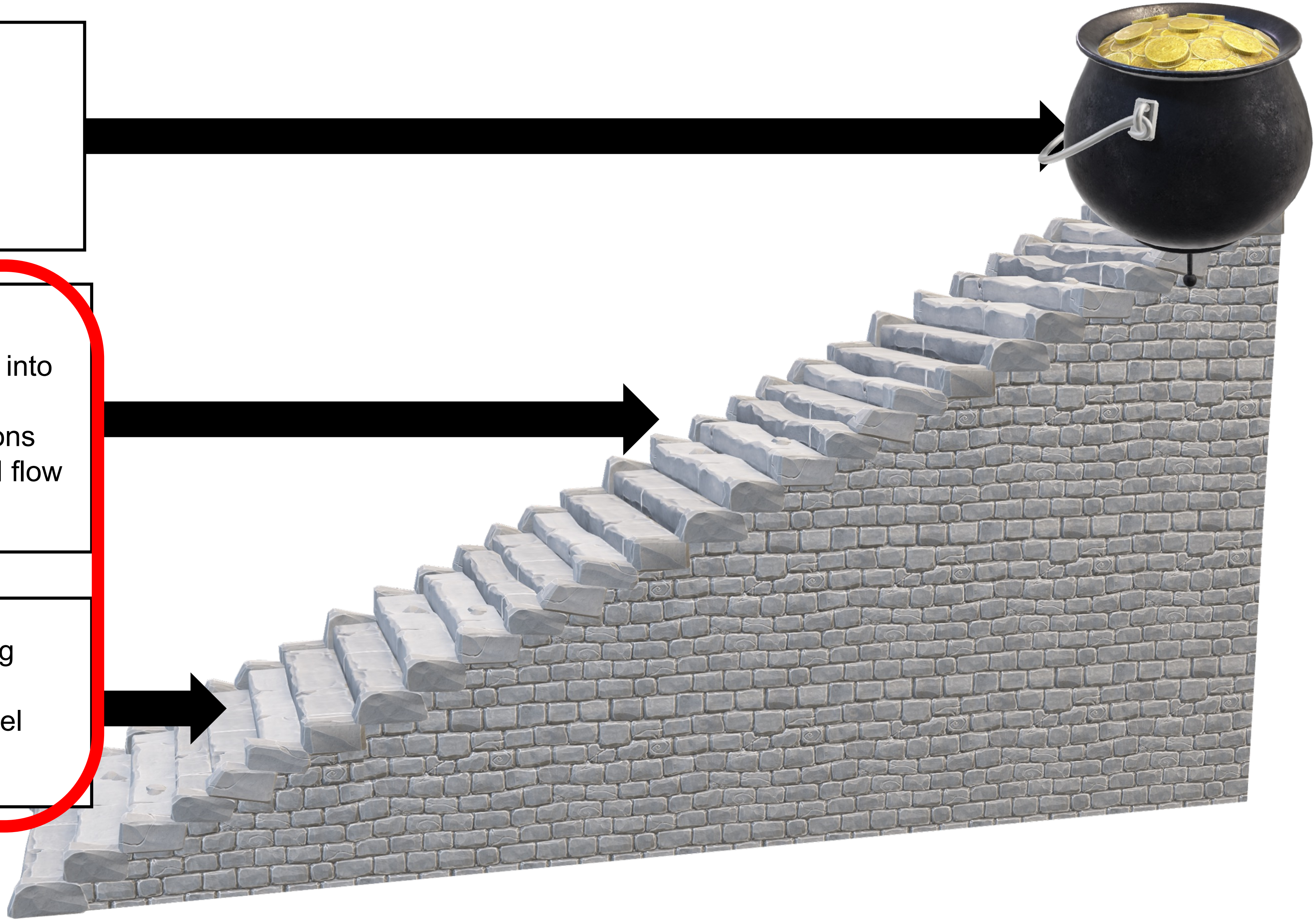
- Configure custom IO/clocking setups
- Develop custom FireSim extensions
- Integrate and tape-out a complete SoC

Evaluation-level

- Integrate or develop custom hardware IP into Chipyard
- Run FireSim FPGA-accelerated simulations
- Push a design through the Hammer VLSI flow
- Build your own system

Exploratory-level

- Configure a custom SoC from pre-existing components
- Generate RTL, and simulate it in RTL level simulation
- Evaluate existing RISC-V designs



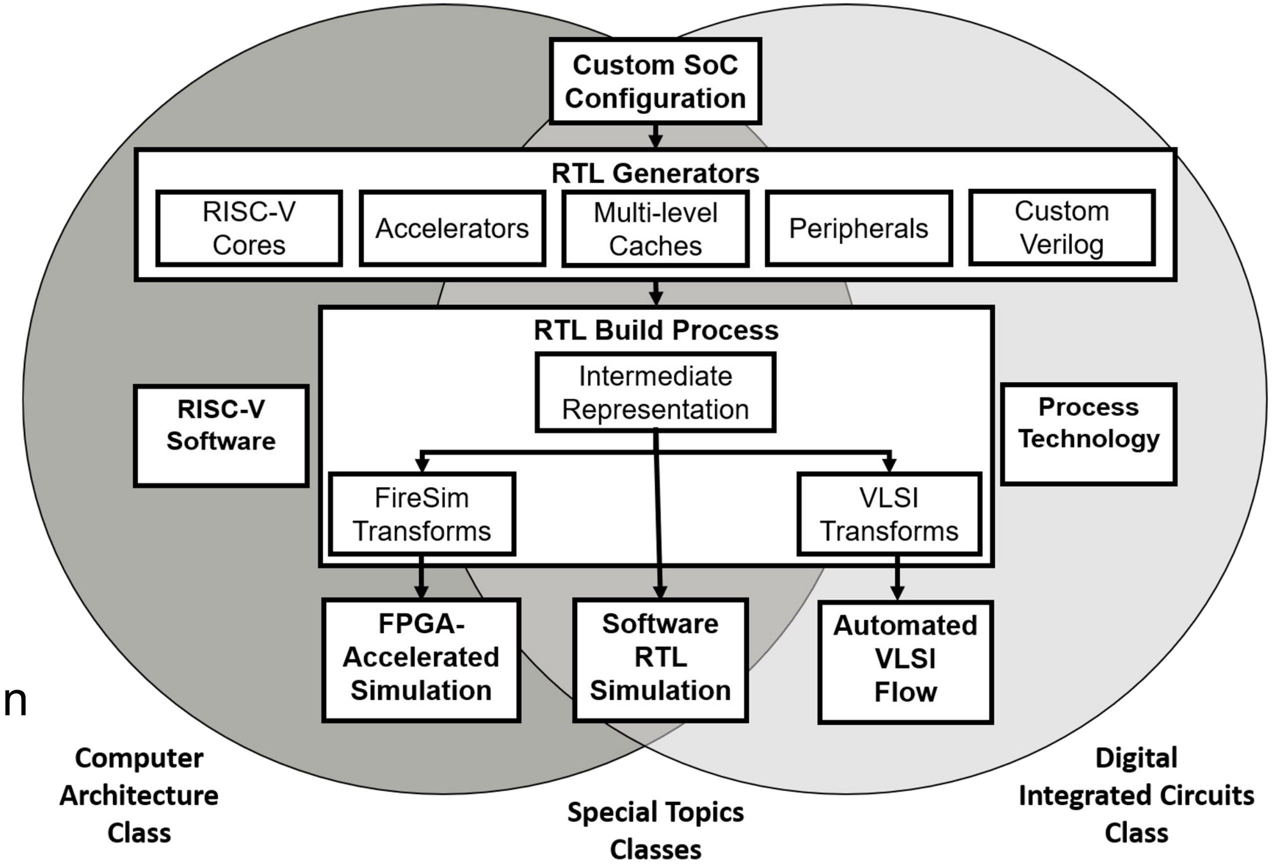
CHIPYARD For Education

Proven in many Berkeley Architecture courses

- Hardware for Machine Learning
- Undergraduate Computer Architecture
- Graduate Computer Architecture
- Advanced Digital ICs
- Tapeout HW design course

Advantages of common shared HW framework

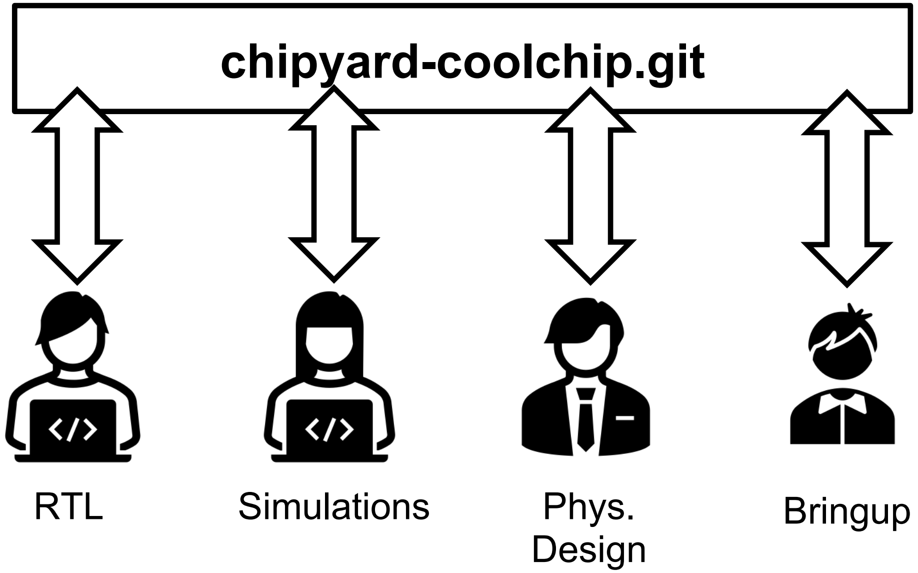
- Reduced ramp-up time for students
- Students learn framework once, reuse it in later courses
- Enables more advanced course projects (tapeout a chip in 1 semester)



CHIPYARD For Tapeouts

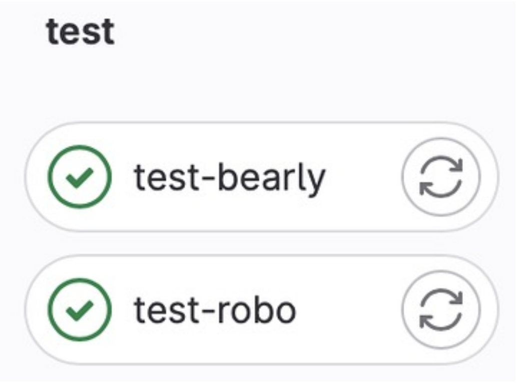
Standard Chipyard "Flow" For Tapeout

1. **RTL Development –** Develop new accelerators/devices and test rapidly
2. **FireSim –** Evaluate your design on real workloads
3. **Hammer -** Reusable/extensible VLSI flow
4. **Bringup –** generate FPGA bringup platforms using Chipyard



Chipyard is a single-source-of-truth for a chip

- Enables parallel workflows across different parts of the flow
- Reproducible environments simplify debugging
- Continuous integration for tapeouts



test

✓ test-bearly

✓ test-robo

CHIPYARD Community

Documentation:

- https://chipyard.readthedocs.io/en/dev/
- 133 pages
- Most of today's tutorial content is covered there

Mailing List:

- google.com/forum/#!forum/chipyard

Open-sourced:

- All code is hosted on GitHub
- Issues, feature-requests, PRs are welcomed

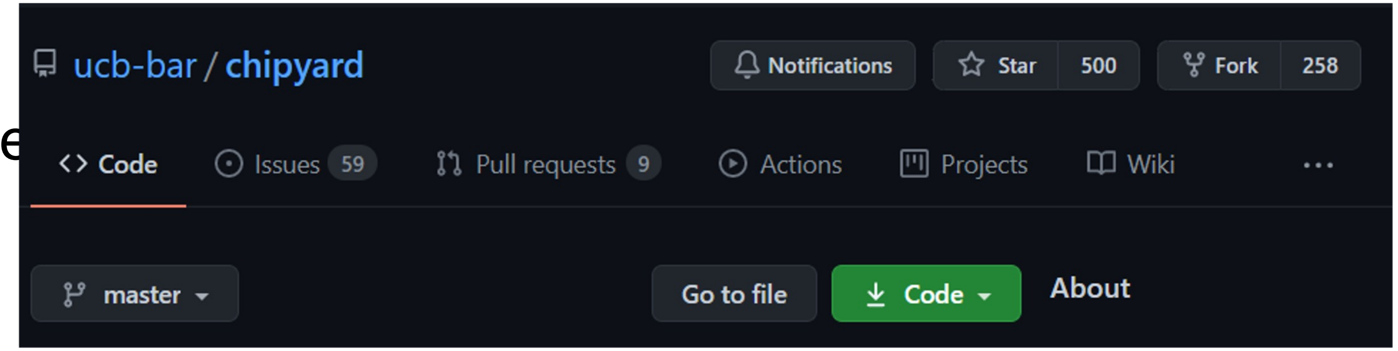
Welcome to Chipyard's documentation!

CHIPYARD

Chipyard is a framework for designing and evaluating full-system hardware using agile teams. It is composed of a collection of tools and libraries designed to provide an integration between open-source and commercial tools for the development of systems-on-chip.

Important

New to Chipyard? Jump to the Initial Repository Setup page for setup instructions.



CHIPYARD

An open, extensible research and design platform for RISC-V SoCs

- Unified framework of parameterized generators
- One-stop-shop for RISC-V SoC design exploration
- Supports variety of flows for multiple use cases
- Open-sourced, community and research-friendly

Welcome to Chipyard's documentation!

Chipyard is a framework for designing and evaluating full-system hardware using agile teams. It is composed of a collection of tools and libraries designed to provide an integration between open-source and commercial tools for the development of systems-on-chip.

Important

New to Chipyard? Jump to the Initial Repository Setup page for setup instructions.