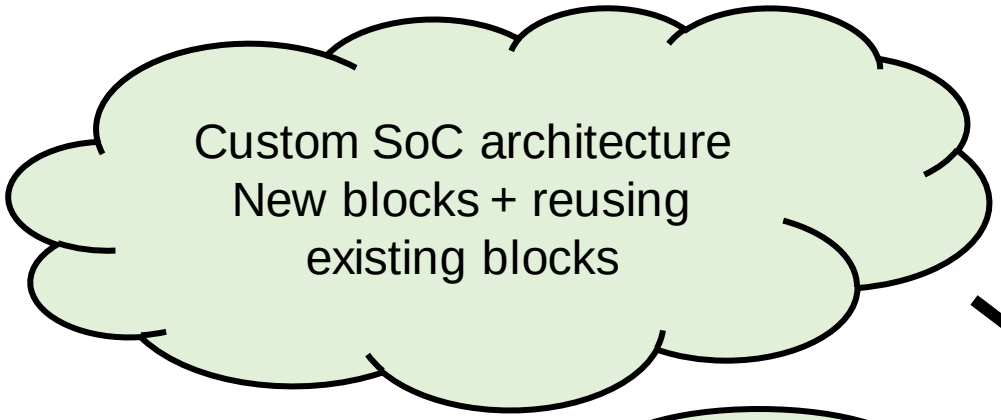
Chipyard Framework Overview

Jerry Zhao
UC Berkeley
jzh@berkeley.edu

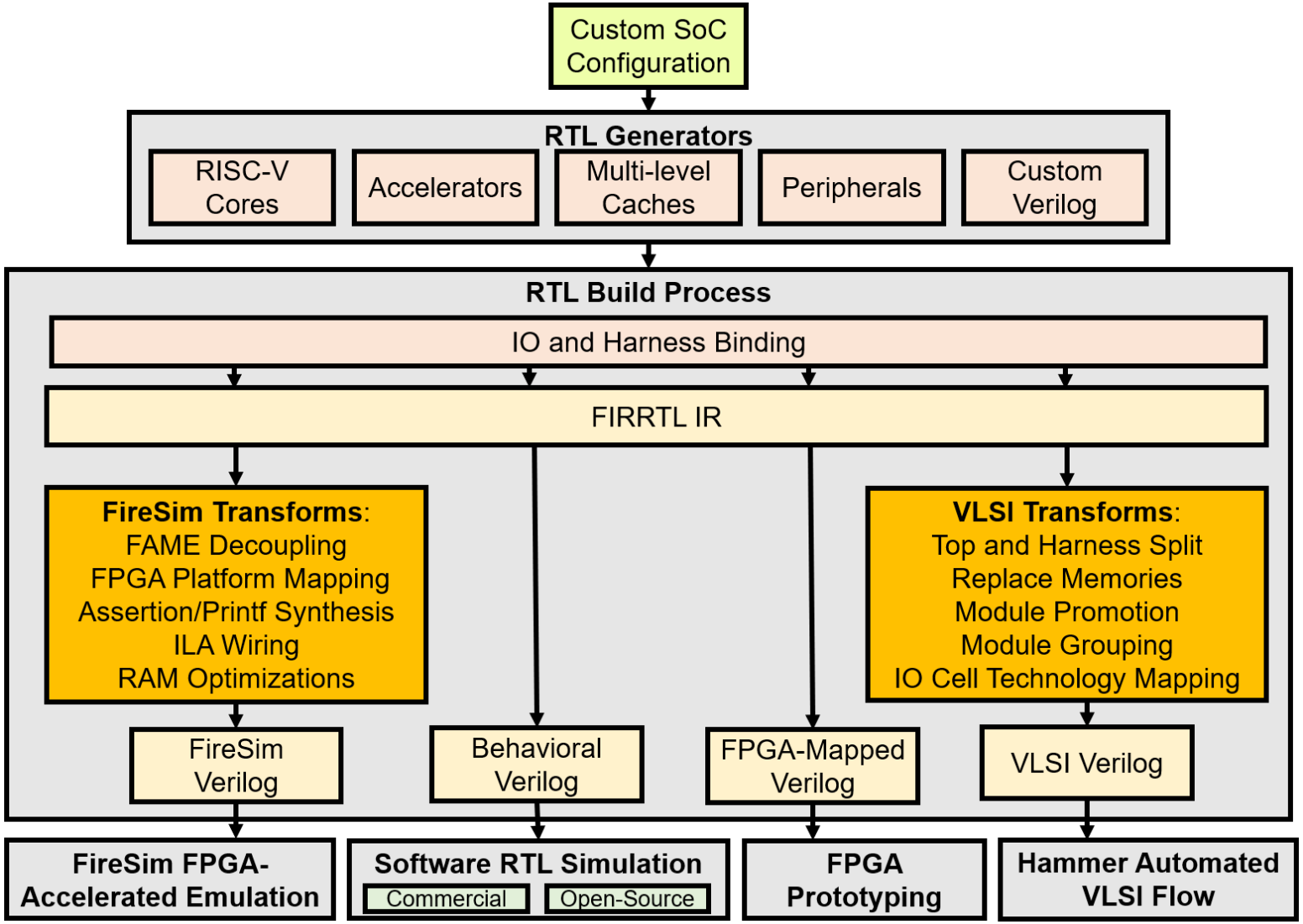
Berkeley Architecture Research

Use Cases

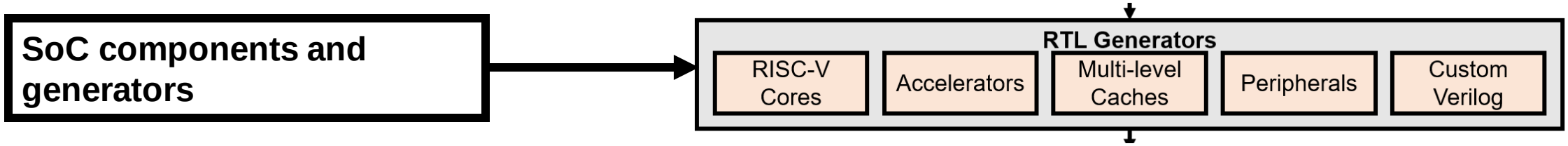


- RTL Simulation running test binaries
- FPGA-accelerated simulation running full workloads
- FPGA prototyping for fast cool demos
- Tape-out the SoC to get actual silicon results

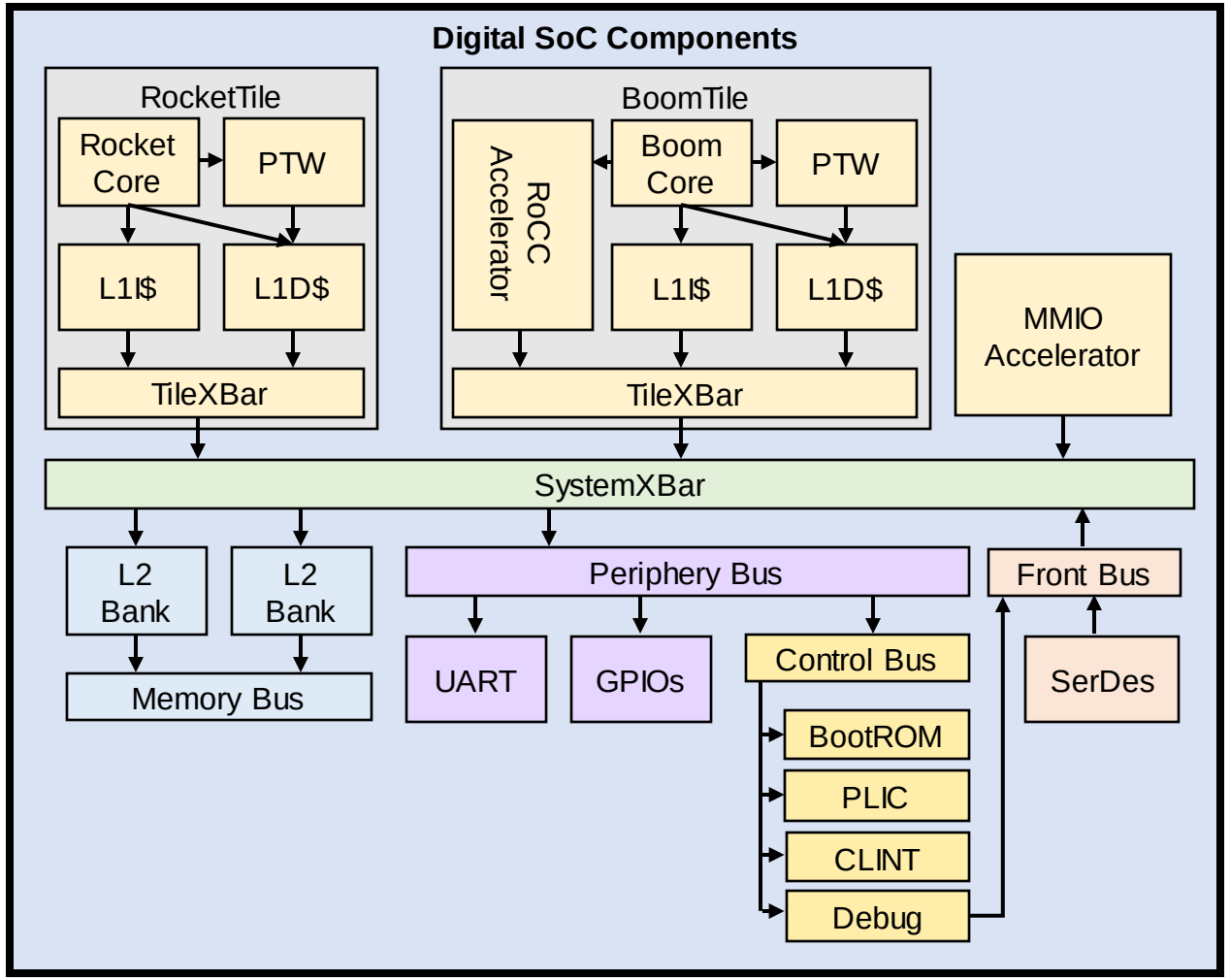
Chipyard Organization



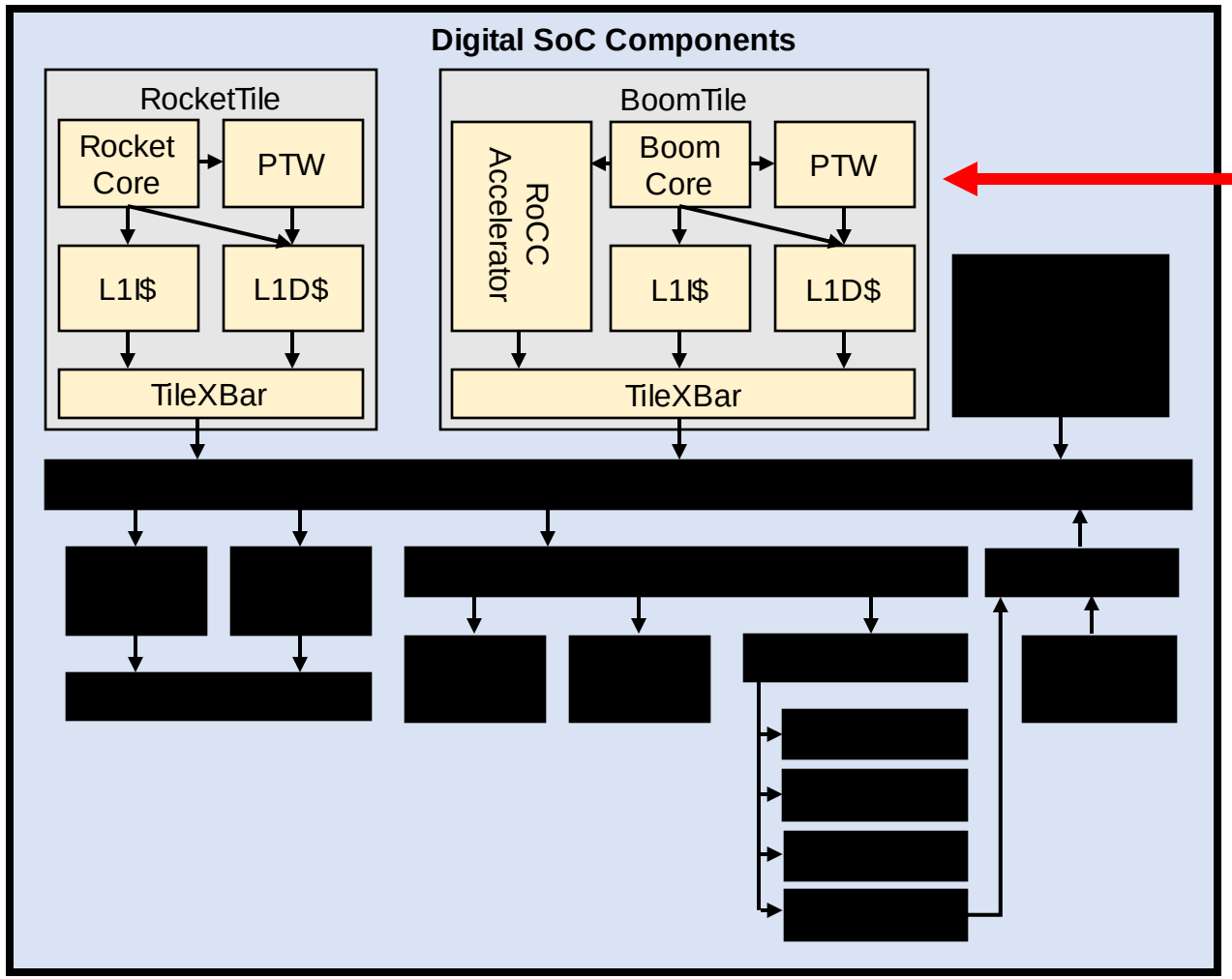
Chipyard Organization



SoC Organization



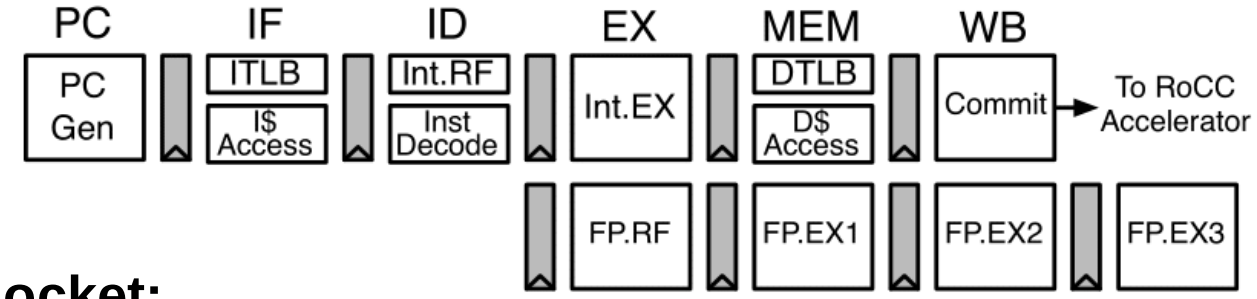
SoC Organization



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

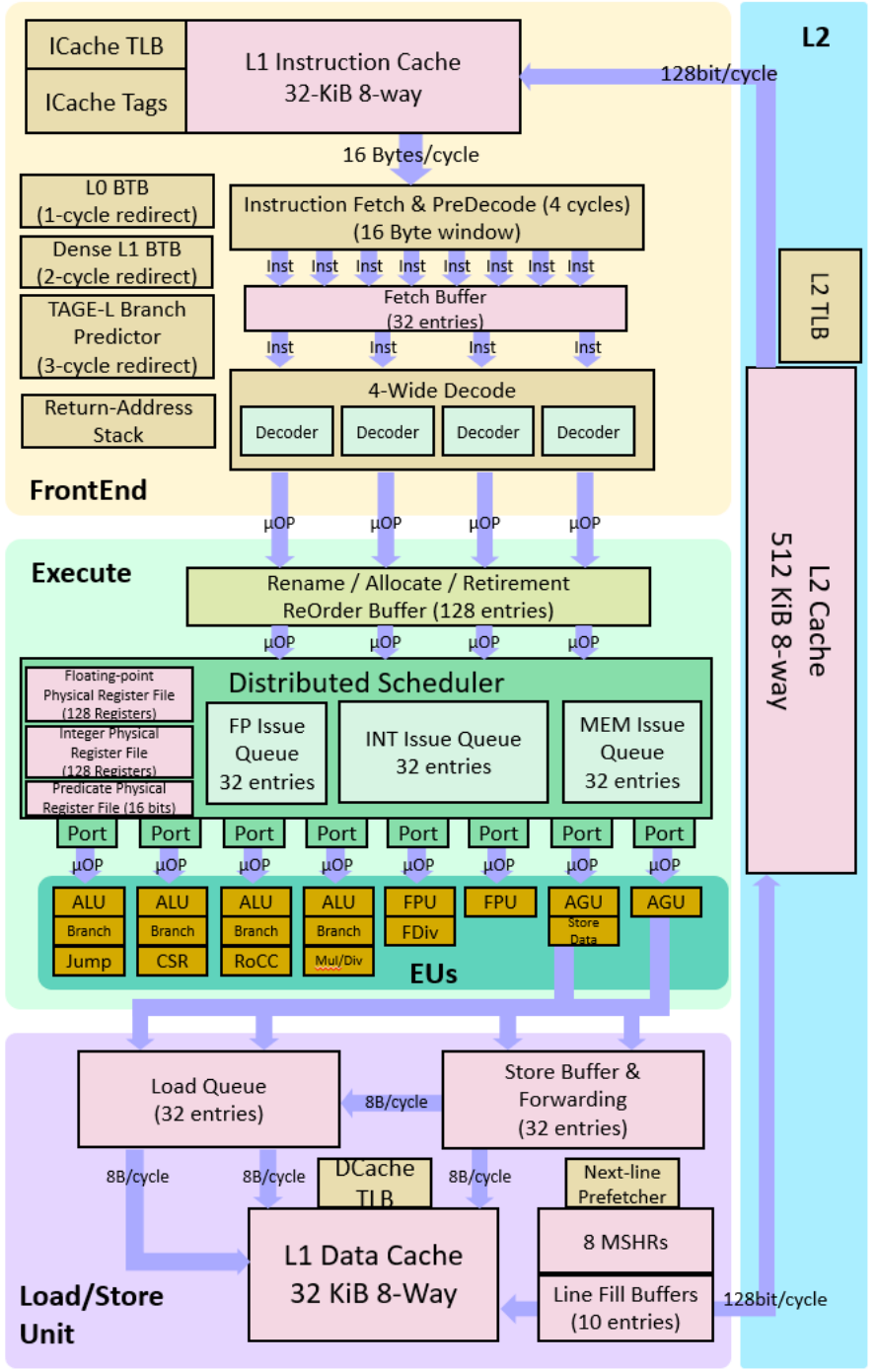


Rocket:

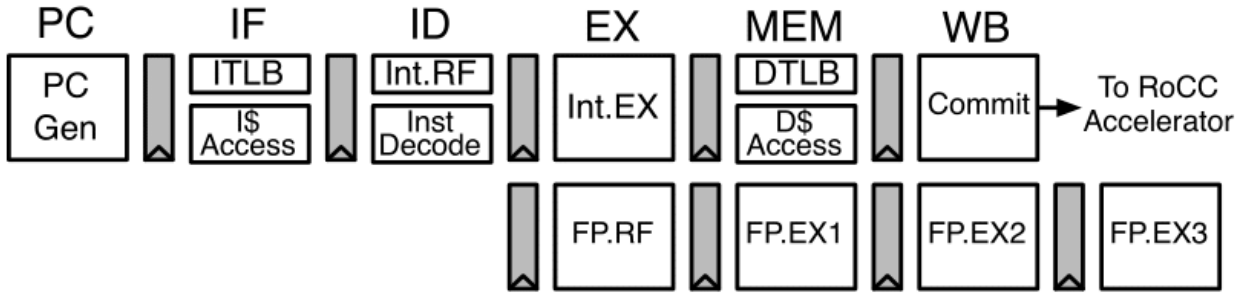
- First open-source RISC-V CPU
- In-order, single-issue RV64GC core
- Efficient design point for low-power devices

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 systems

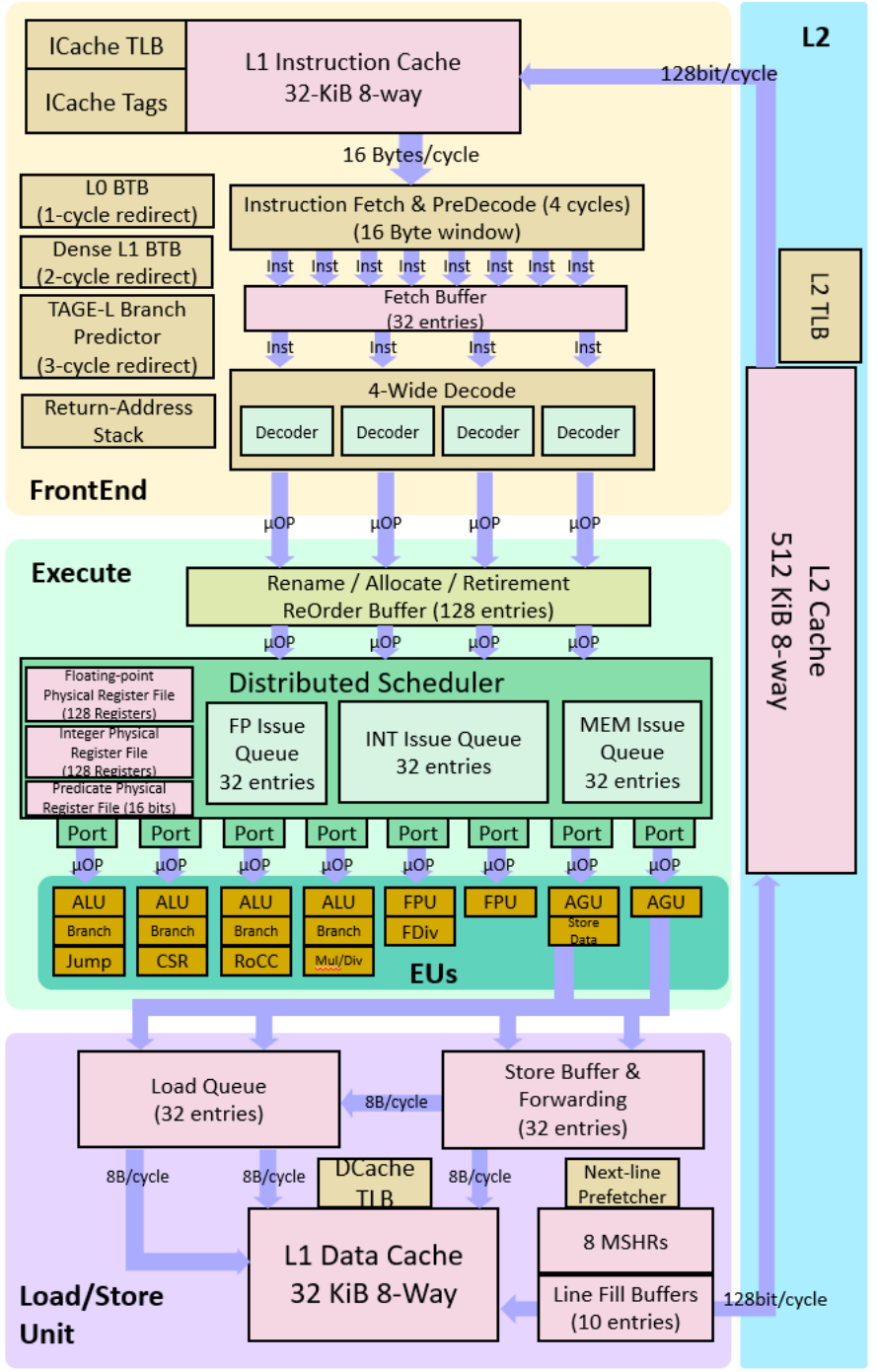


Rocket and BOOM

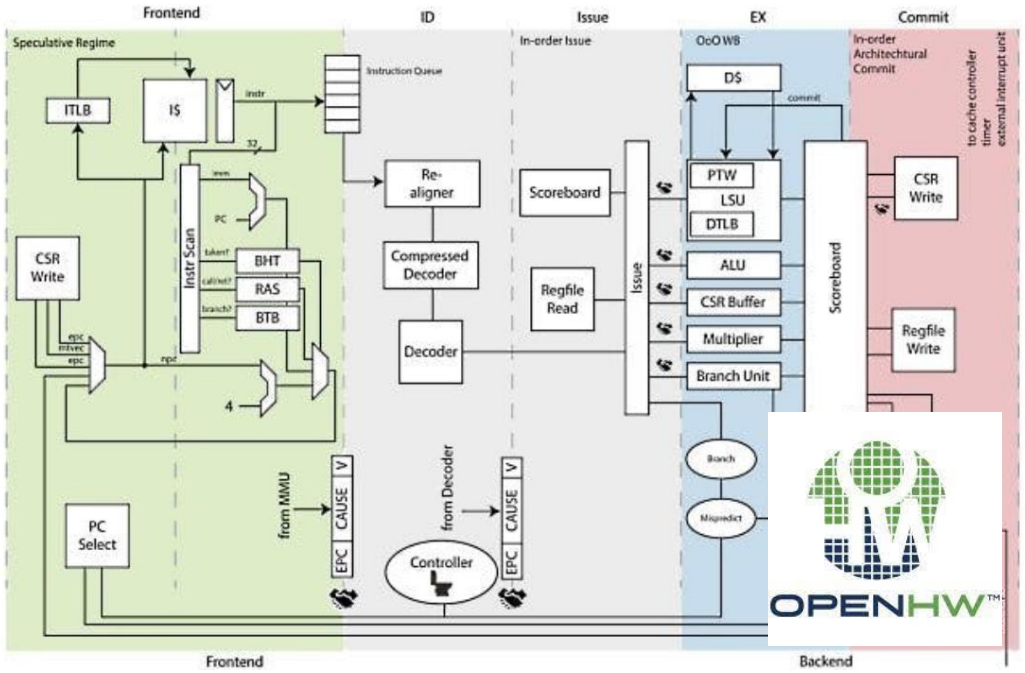


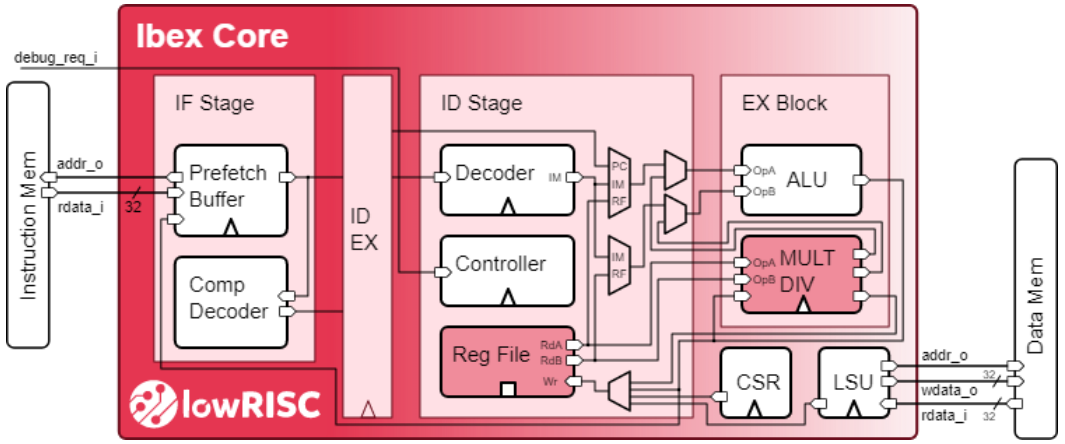
Rocket and SonicBOOM:

- 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



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

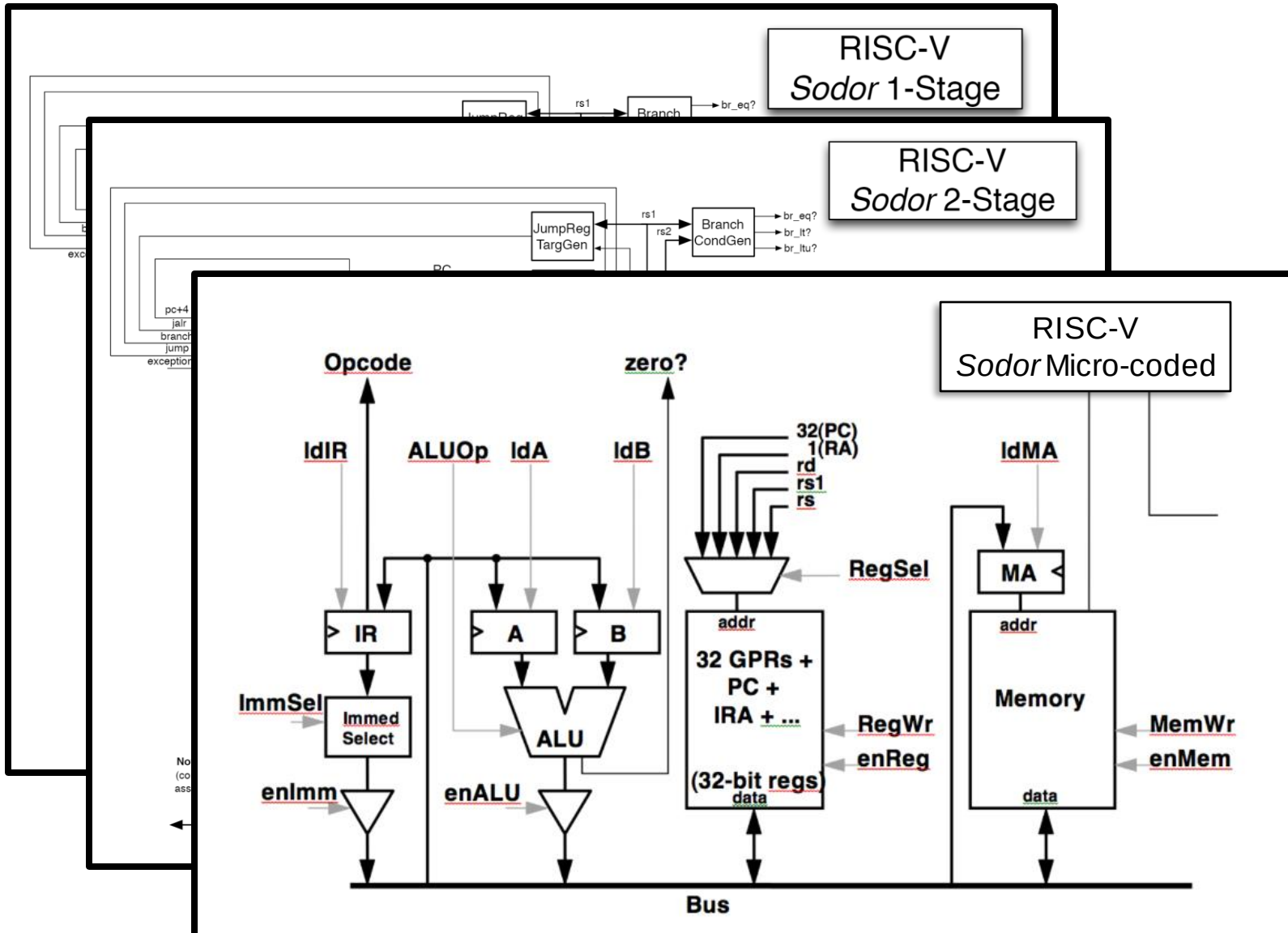
Ibex (Formerly Zero-RISCY):

- 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 Educational Cores

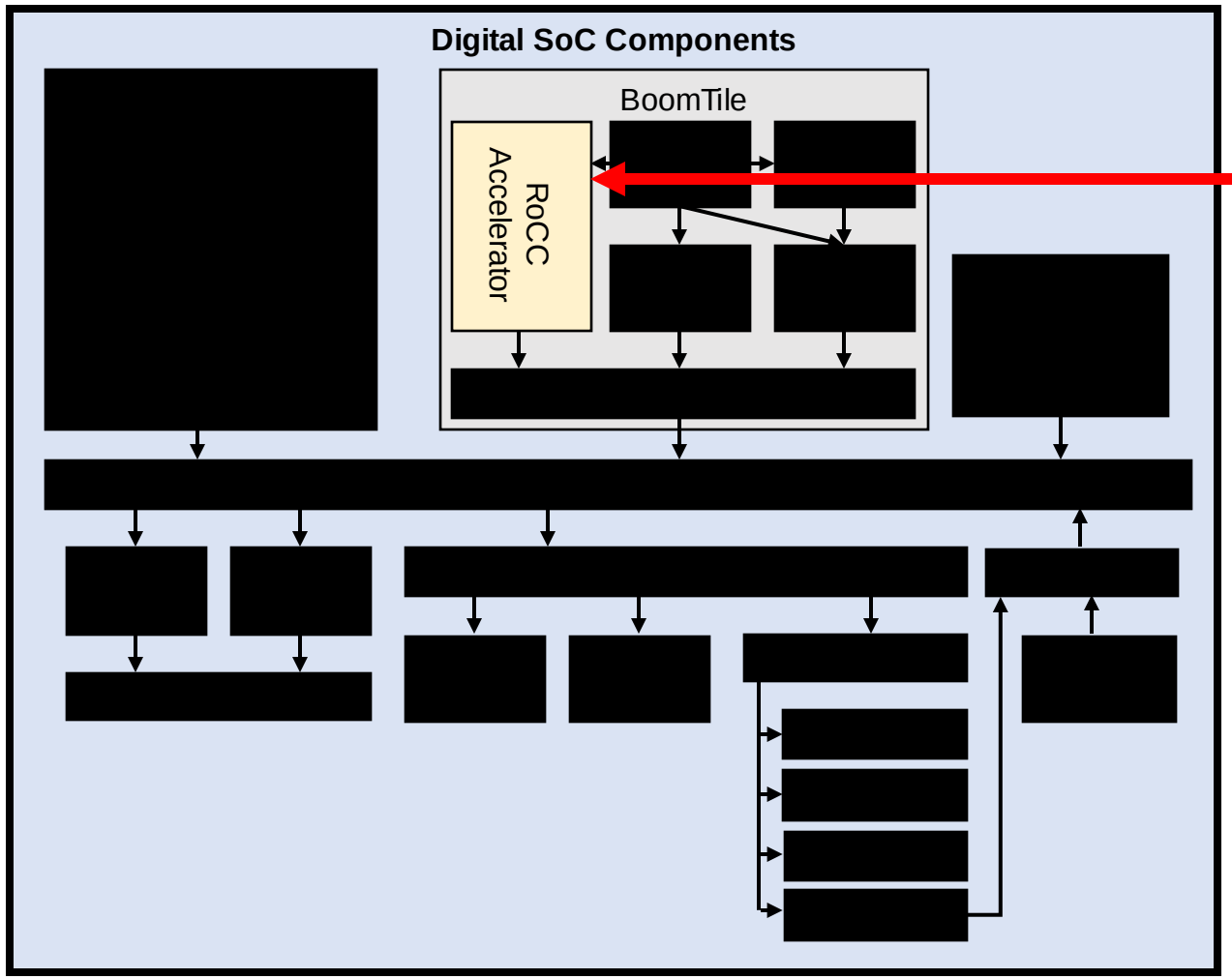
Sodor Core Collection

- 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



Berkeley Architecture Research

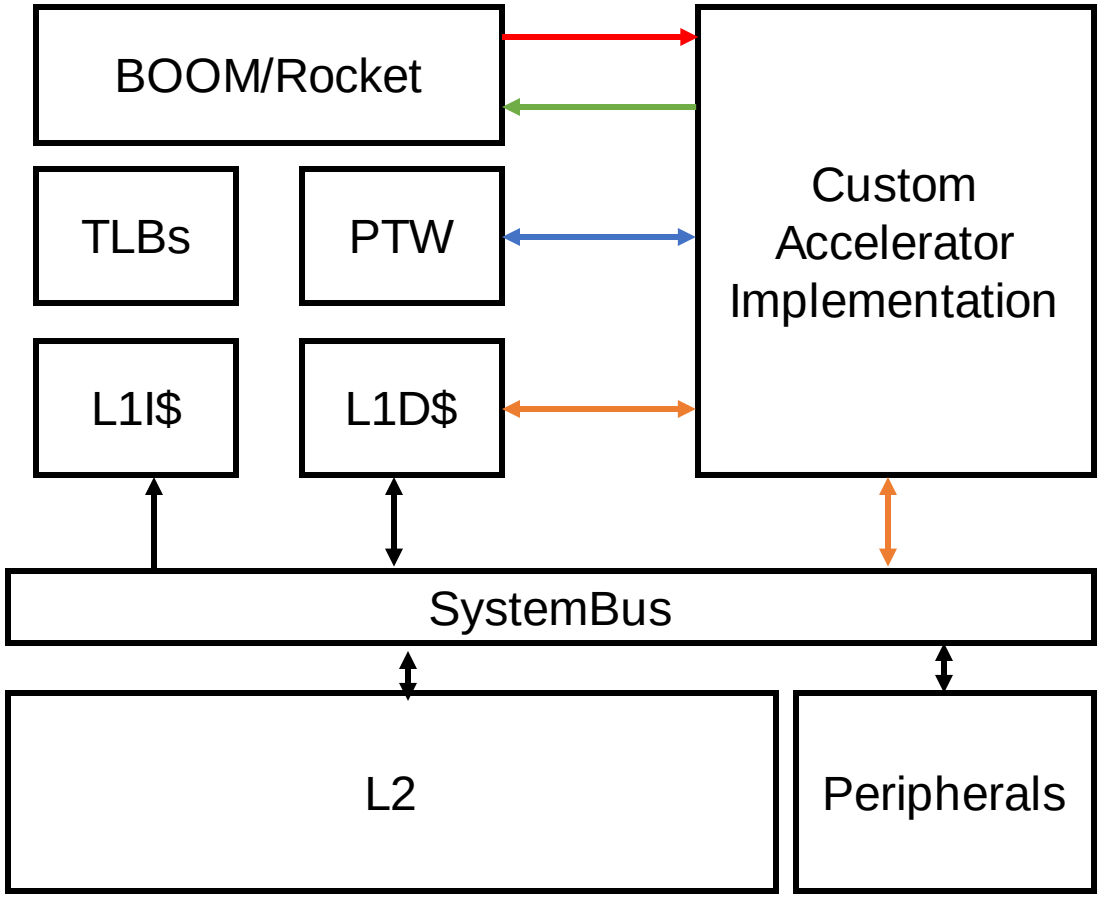
SoC Organization



RoCC Accelerators:

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

RoCC Accelerators



Core automatically decodes + sends custom instructions to accelerator

Accelerator can write back into core registers

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

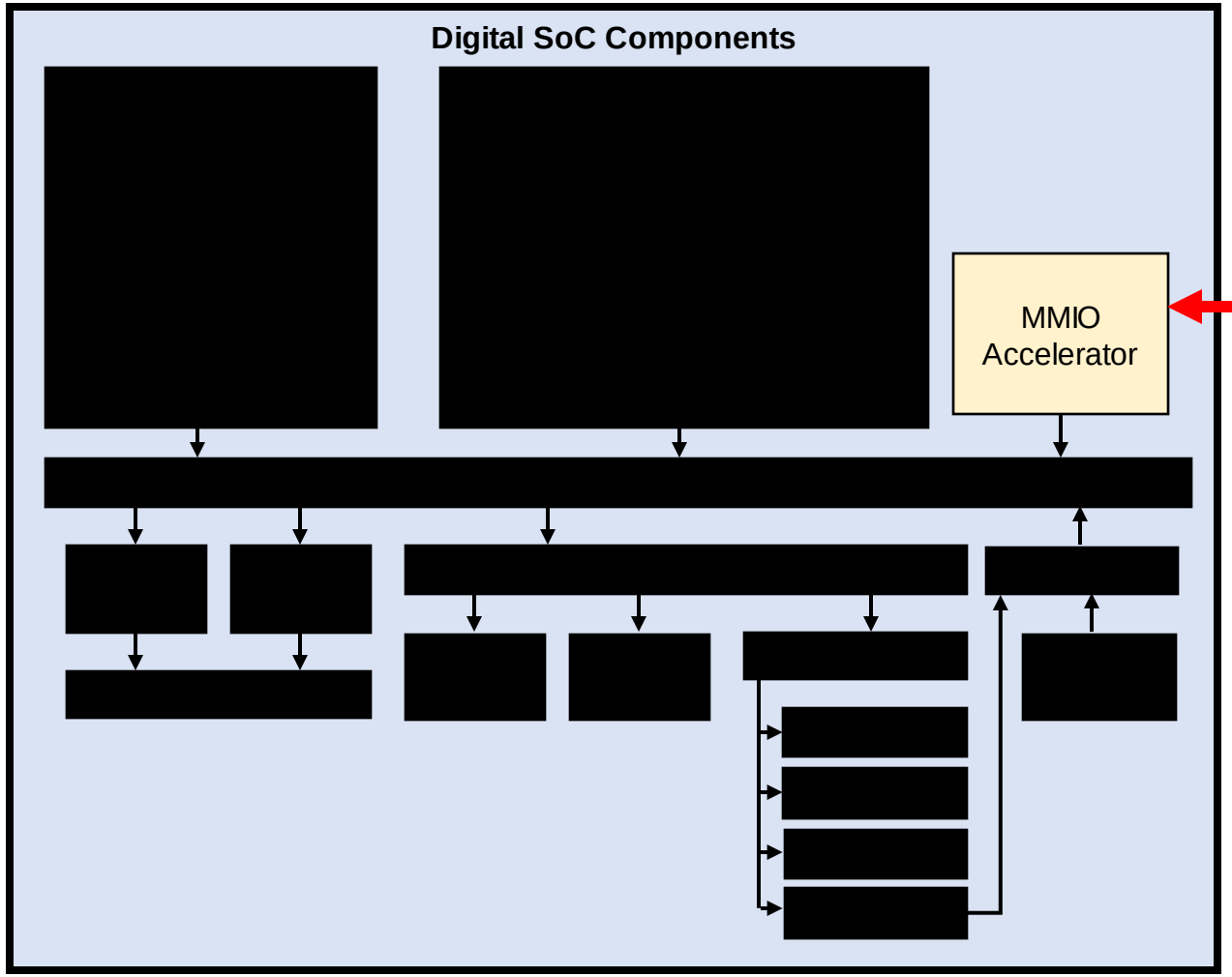
Accelerator can fetch-from/write-to coherent L1 data cache or outer-memory

Flexible interface supports a variety of accelerator designs

Included in Chipyard:

- Gemmini ML accelerator
- Hwacha vector accelerator
- SHA3 accelerator

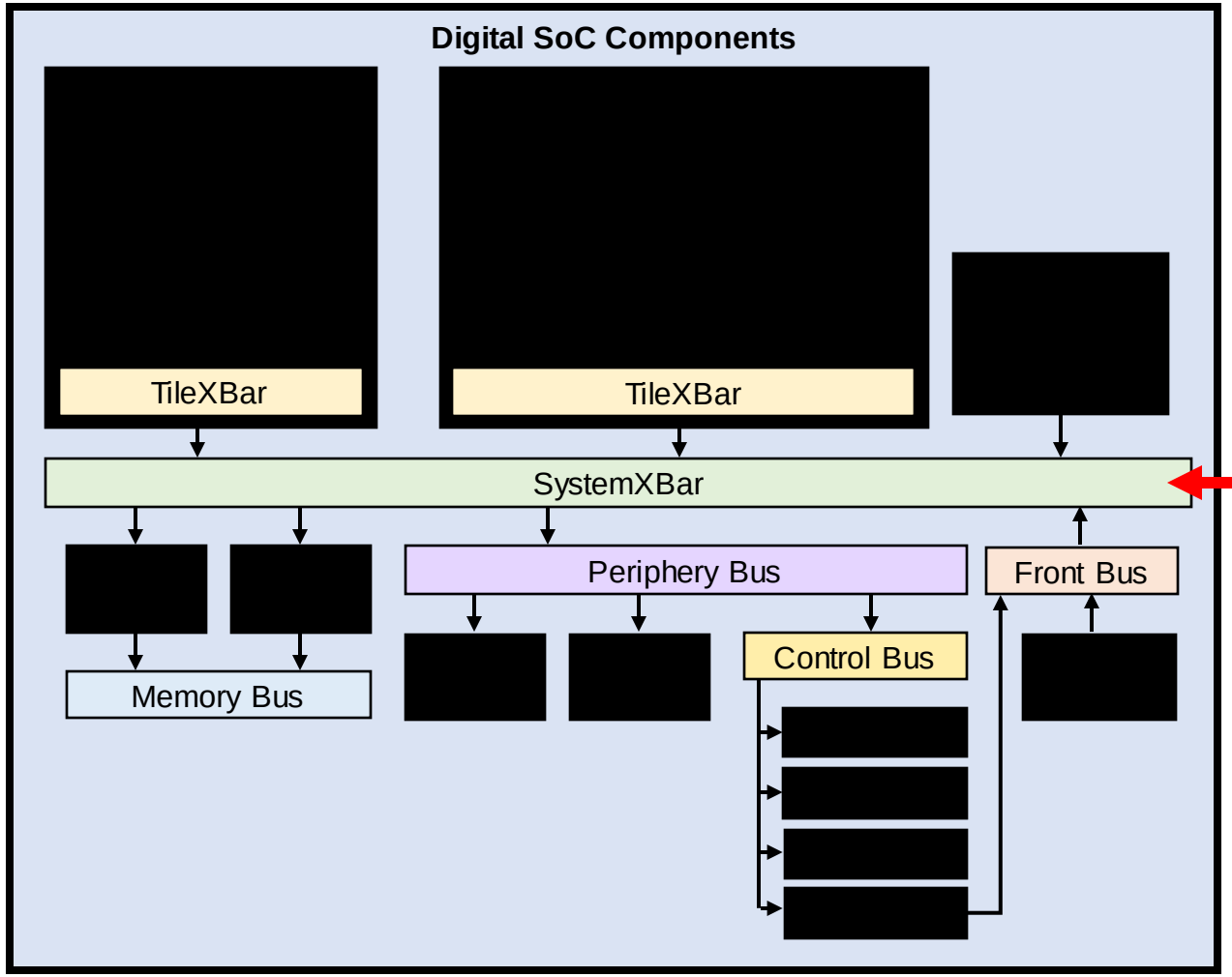
MMIO Accelerators



MMIO Accelerators:

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

SoC Organization



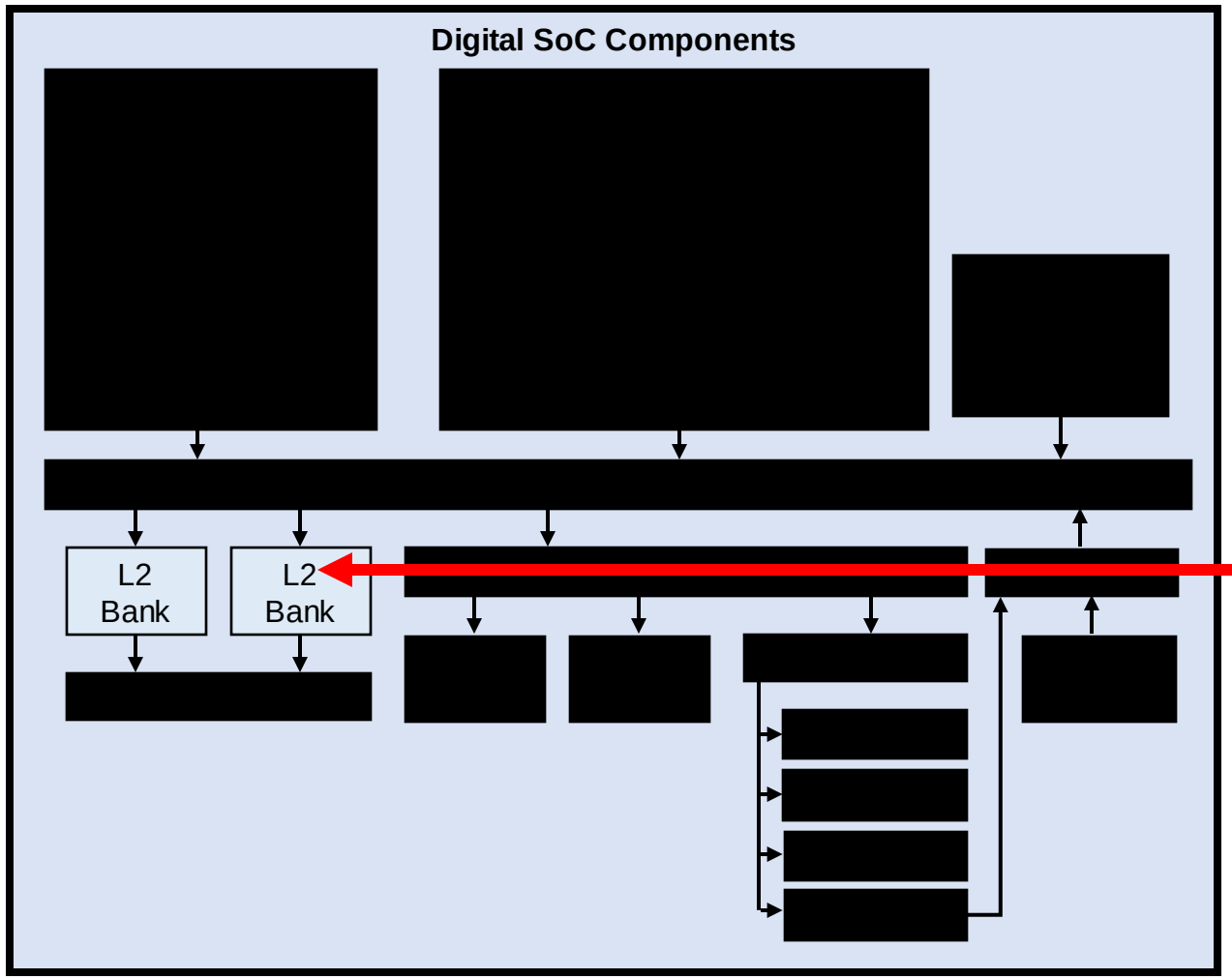
TileLink Standard:

- TileLink is open-source chip-scale interconnect standard
- Cache-coherent
- 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

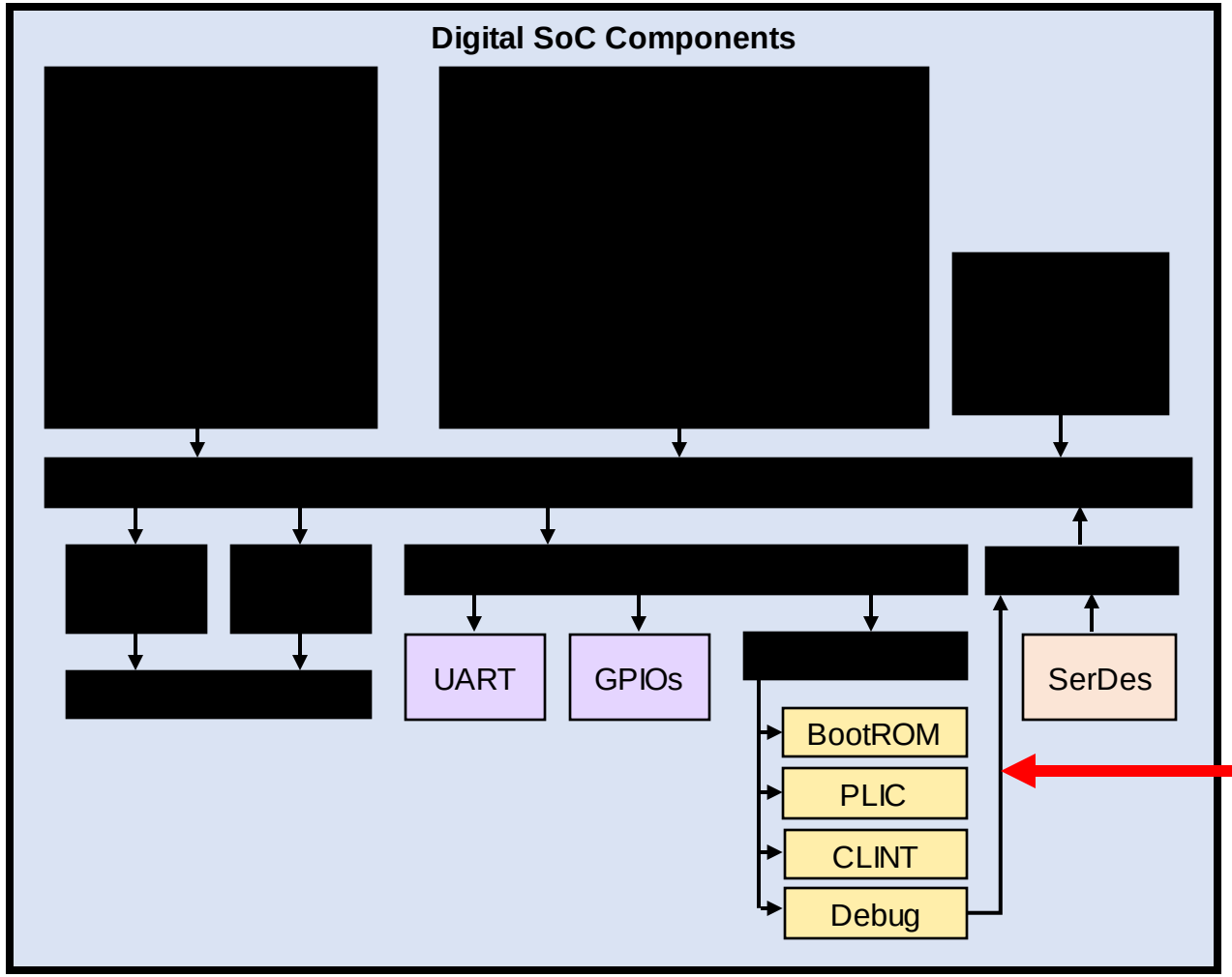
SoC Organization



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

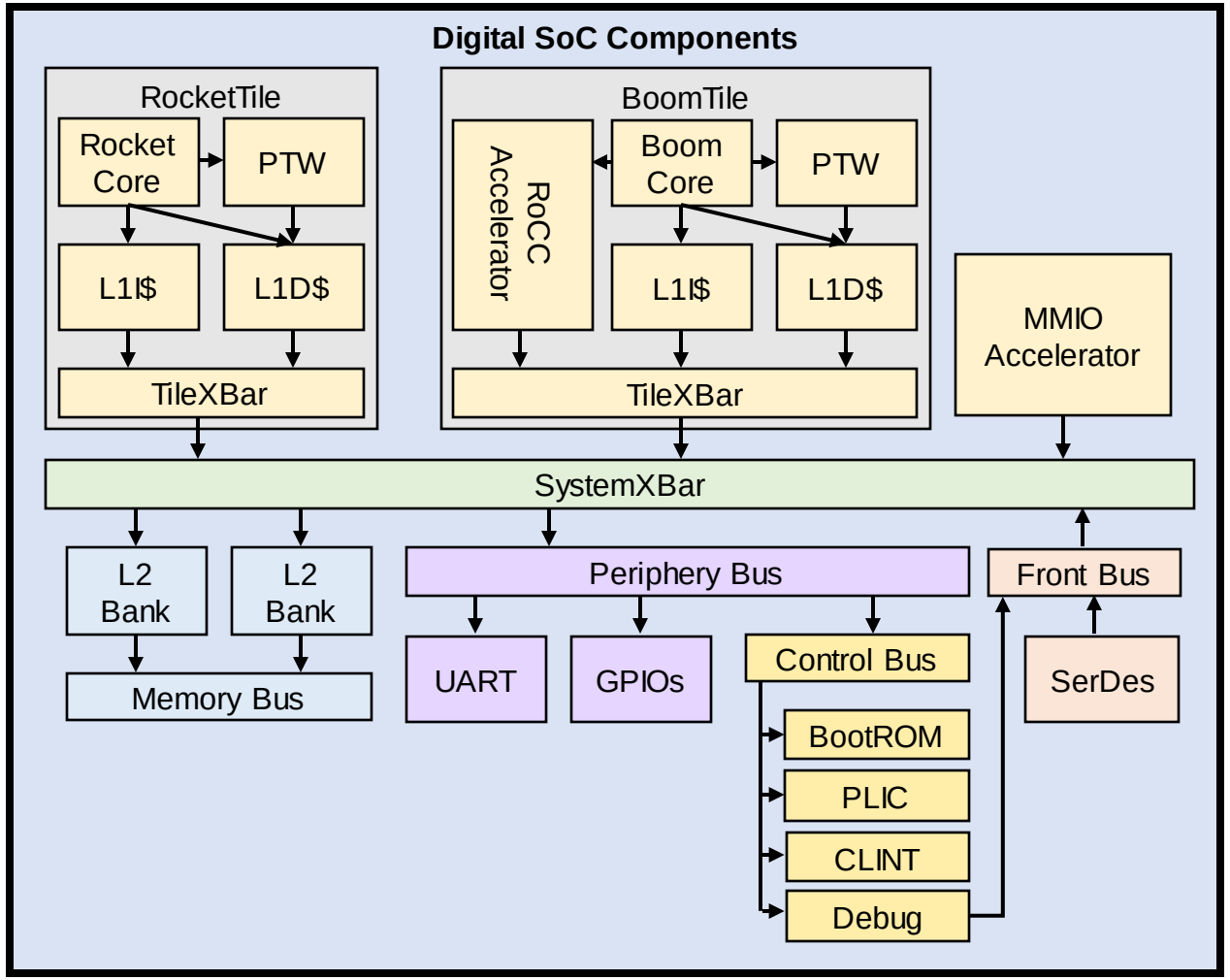
SoC Organization



Peripherals and IO:

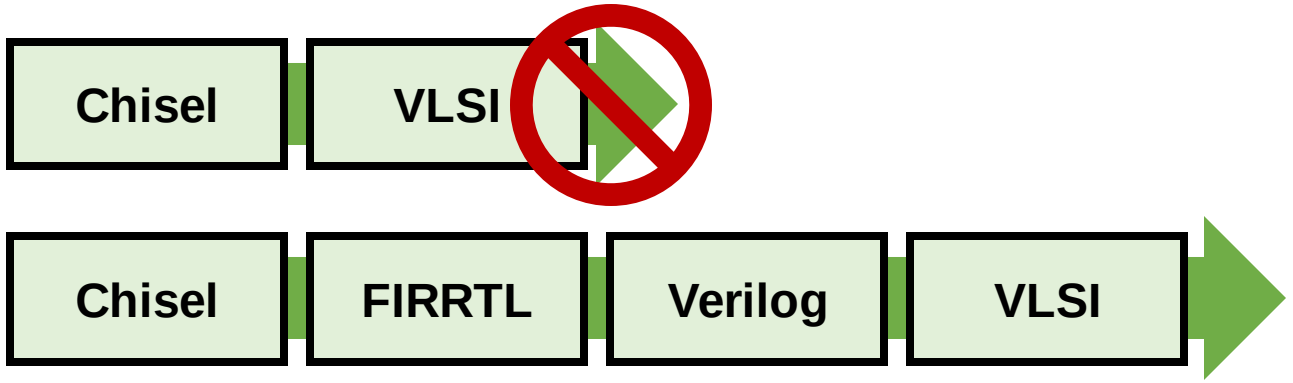
- Open-source RocketChip + SiFive blocks:
  - Interrupt controllers
  - JTAG, Debug module, BootROM
  - UART, GPIOs, SPI, I2C, PWM, etc.
- TestChipIP: useful IP for test chips
  - Clock-management devices
  - SerDes
  - Scratchpads

SoC Organization



Chisel

- Chisel – Hardware Construction Language built on Scala
- What Chisel **IS NOT:**
  - **NOT** Scala-to-gates
  - **NOT** HLS
  - **NOT** tool-oriented language
- What Chisel **IS:**
  - Productive language for **generating** hardware
  - Leverage **OOP/Functional programming** paradigms
  - Enables design of **parameterized generators**
  - **Designer-friendly**: low barrier-to-entry, high reward
  - **Backwards-compatible:** integrates with Verilog black-boxes



Chisel Example

```
// Generalized FIR filter parameterized by coefficients
class FirFilter(bitWidth: Int, coeffs: Seq[Int]) extends Module {
  val io = IO(new Bundle {
    val in = Input(UInt(bitWidth.W))
    val out = Output(UInt(bitWidth.W))
  })
  val zs = Wire(Vec(coeffs.length, UInt(bitWidth.W)))
  zs(0) := io.in
  for (i <- 1 until coeffs.length) {
    zs(i) := RegNext(zs(i-1))
  }
  val products = zs zip coeffs map {
    case (z, c) => z * c.U
  }
  io.out := products.reduce(_ + _)
}
```

Flexible parameters:

- Enables development of highly flexible, parameterized HW generators

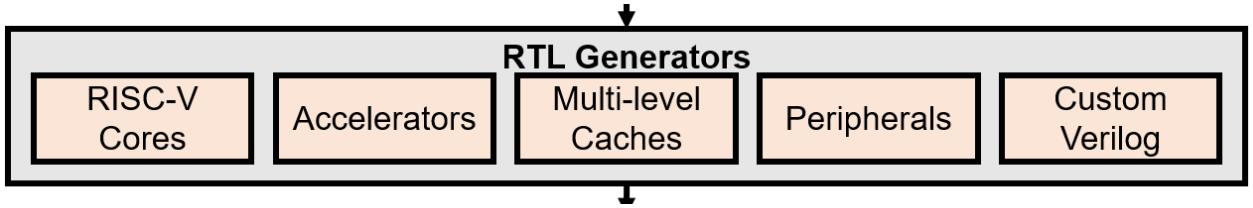
HDL, not HLS:

- Designers reason about wires, registers, gates, IO, etc.
- Familiar Wire, Reg, IO constructs makes Chisel beginner-friendly

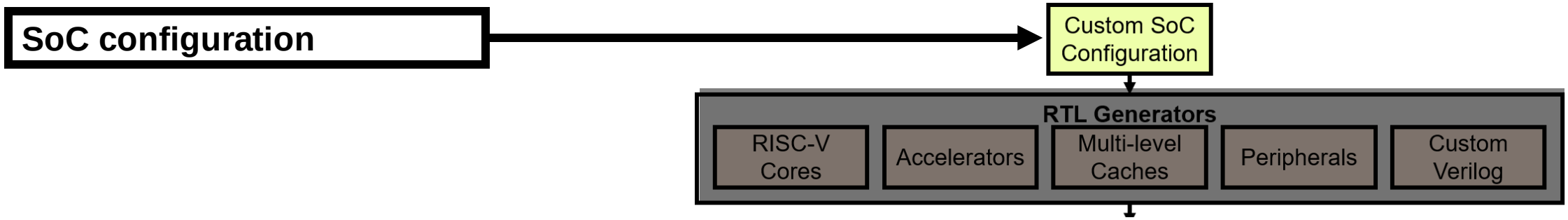
Designer-friendly features

- Powerful OOP/functional programming paradigms
- Strict type-checking encourages “correct-by-construction” design

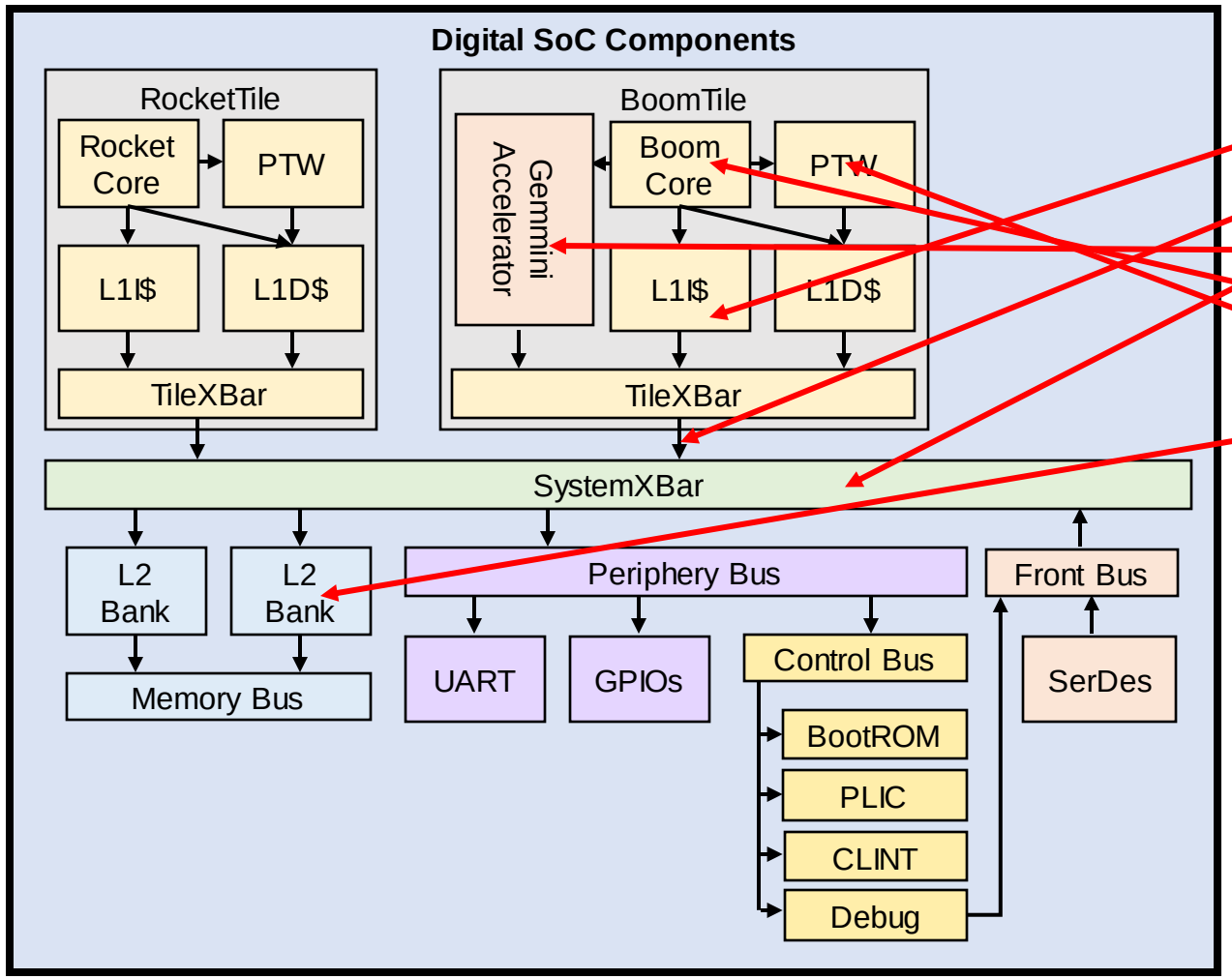
Chipyard Organization



Chipyard Organization

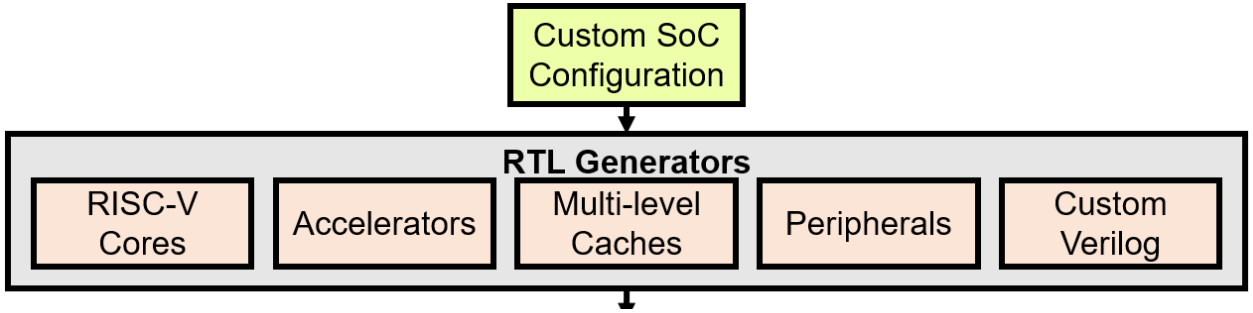


Highly Parameterized Configurations

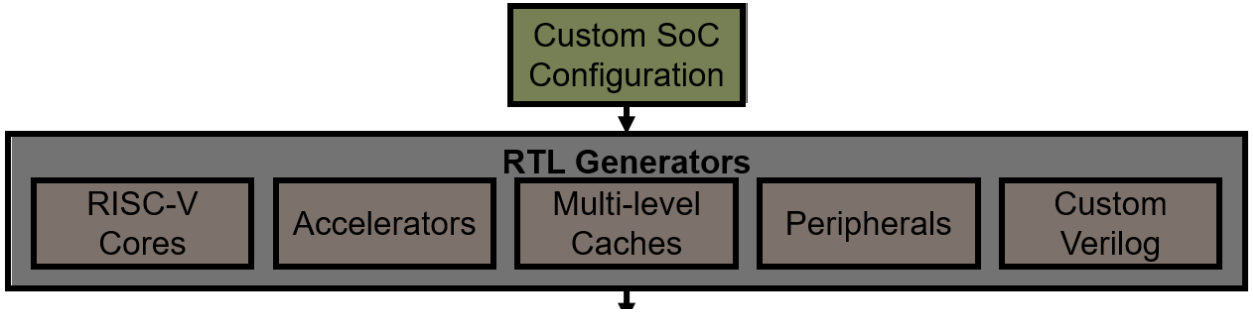



```
class CustomConfig extends Config(
  new WithL1CacheWays(4) ++
  new WithAsyncTiles ++
  new WithSystemBusWidth(128) +
  new WithFPGemmini ++
  new With3WideBooms ++
  new WithL2TLBs(512) ++
  new WithL2Sets(1024) ++

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

Chipyard Organization



Chipyard Organization



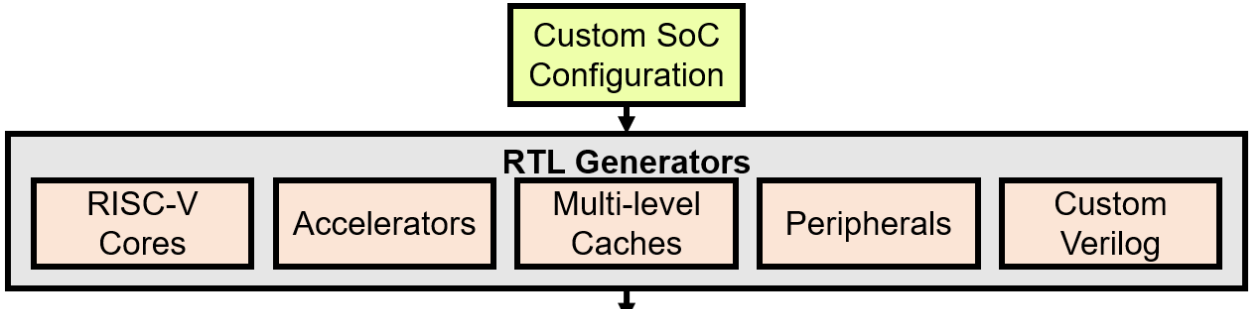
SW RTL Simulation:
- RTL-level simulation with Verilator or VCS
- Hands-on tutorial next

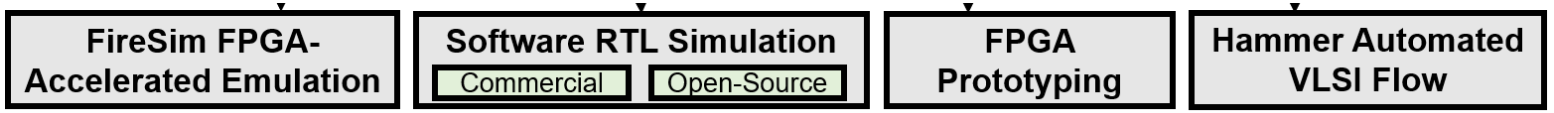
FPGA prototyping:
- Fast, non-deterministic prototypes
- Overview of flow later

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

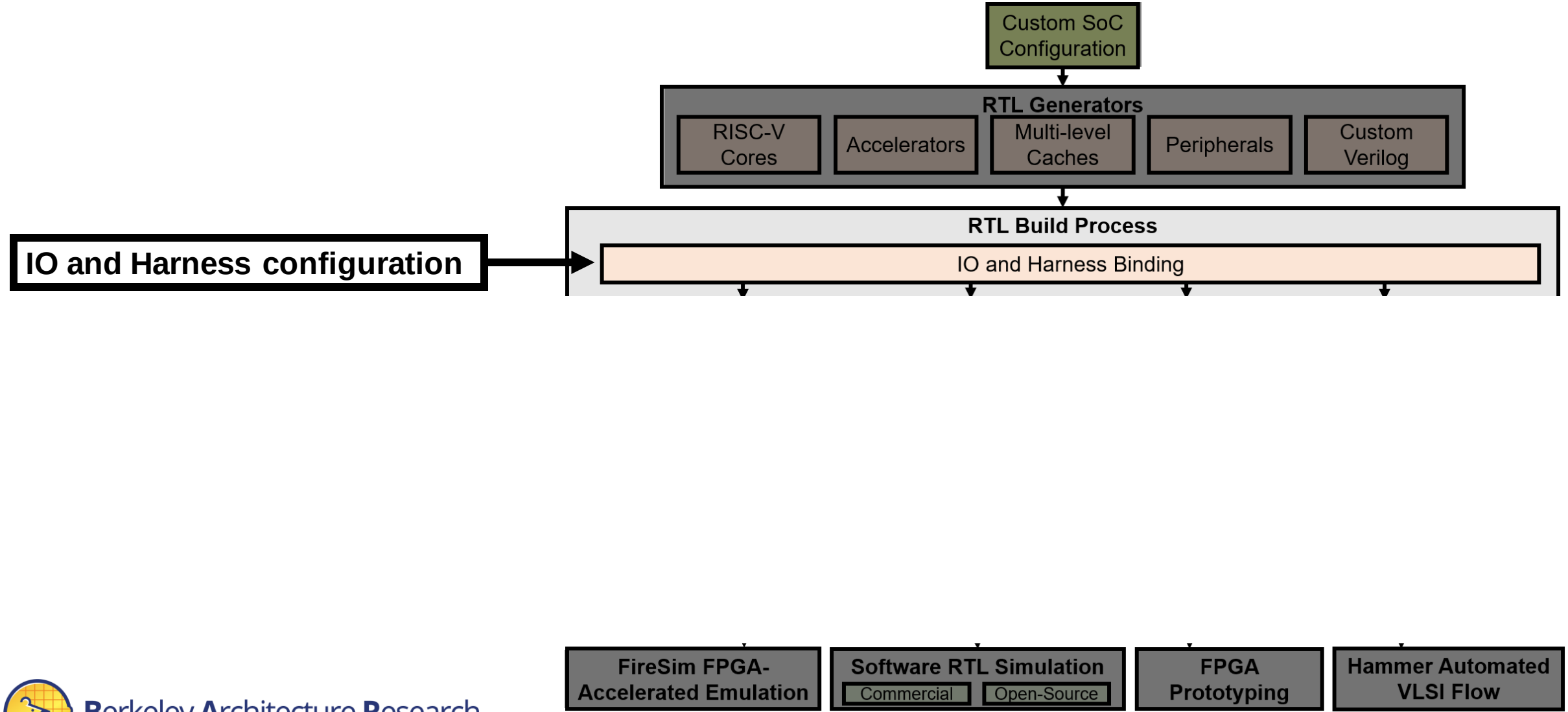
FireSim:
- Fast, accurate FPGA-accelerated simulations
- Hands-on tutorial later

Chipyard Organization

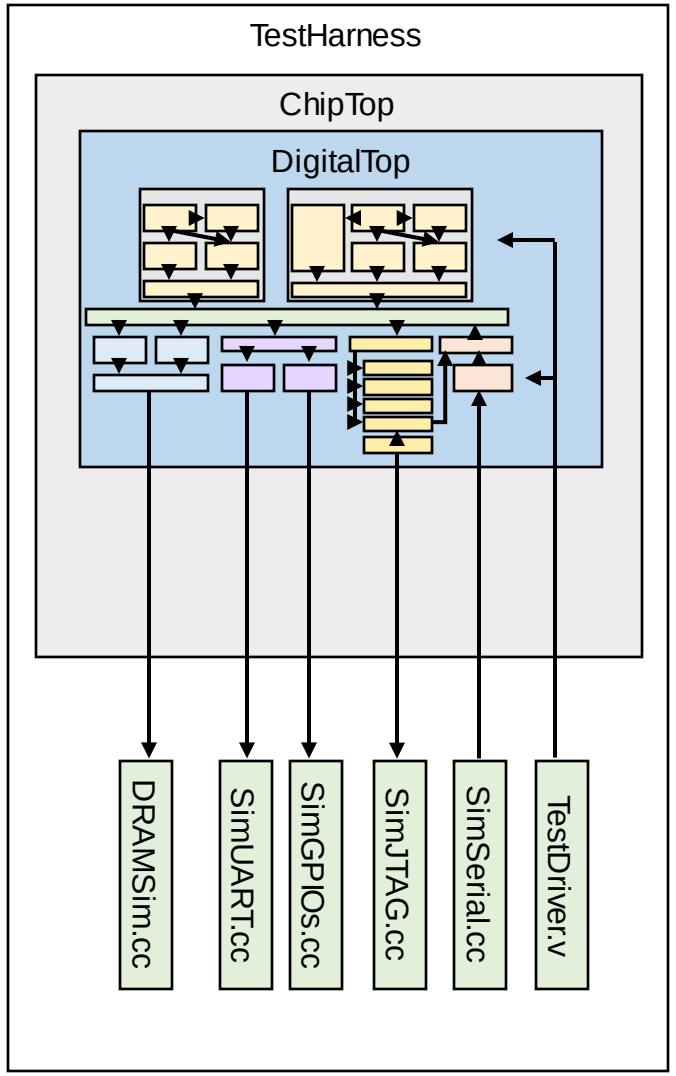


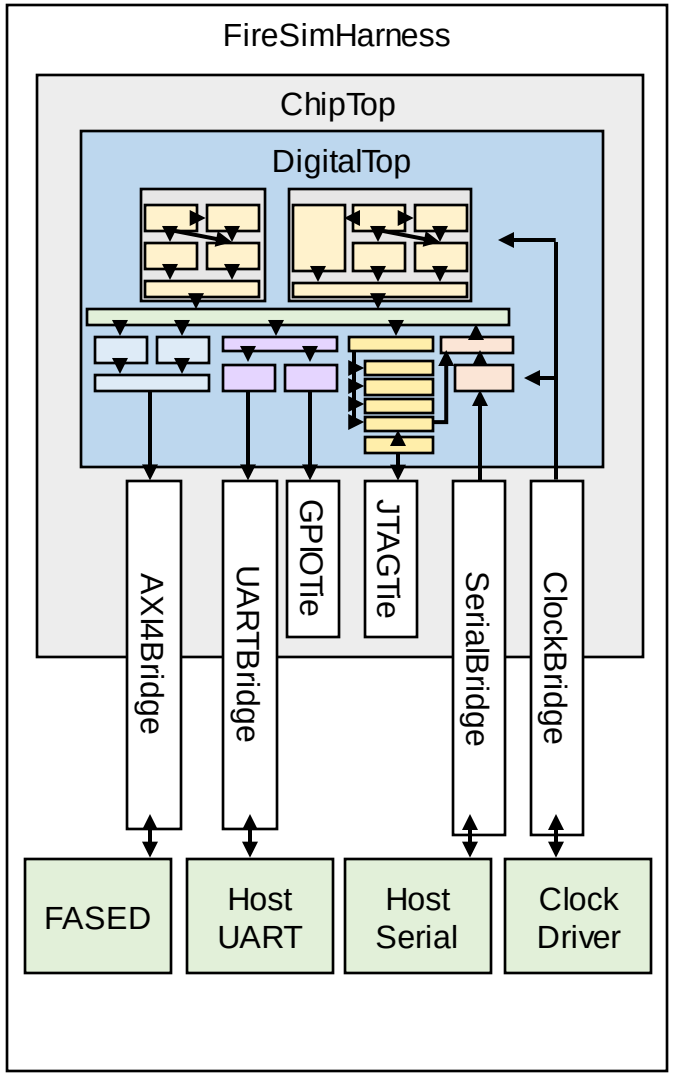


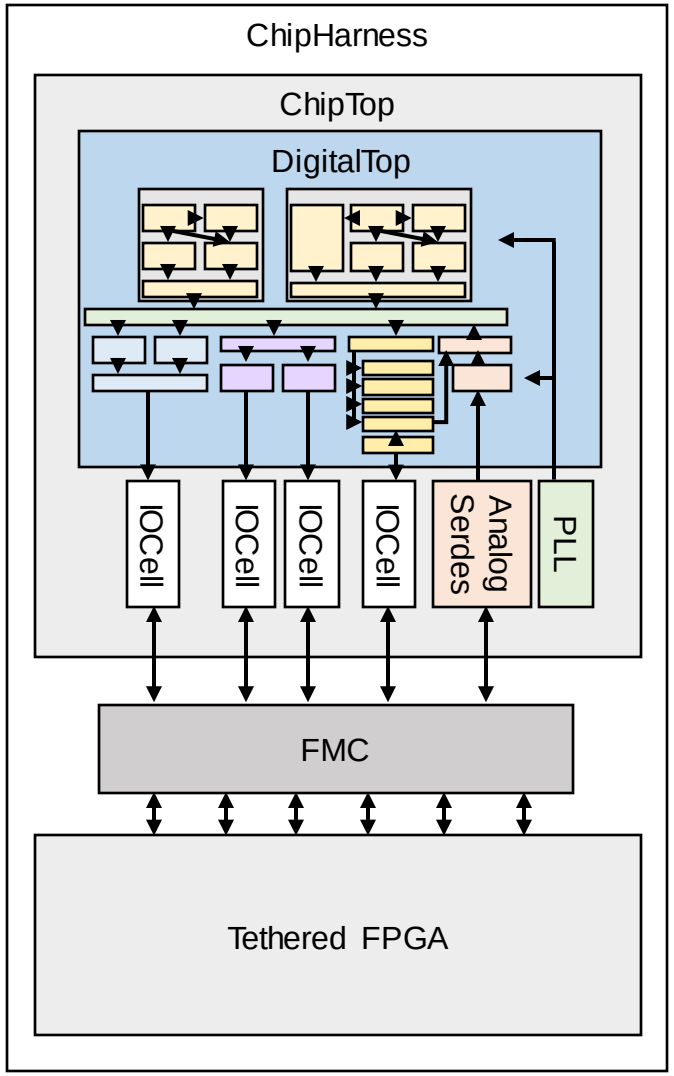
Chipyard Organization



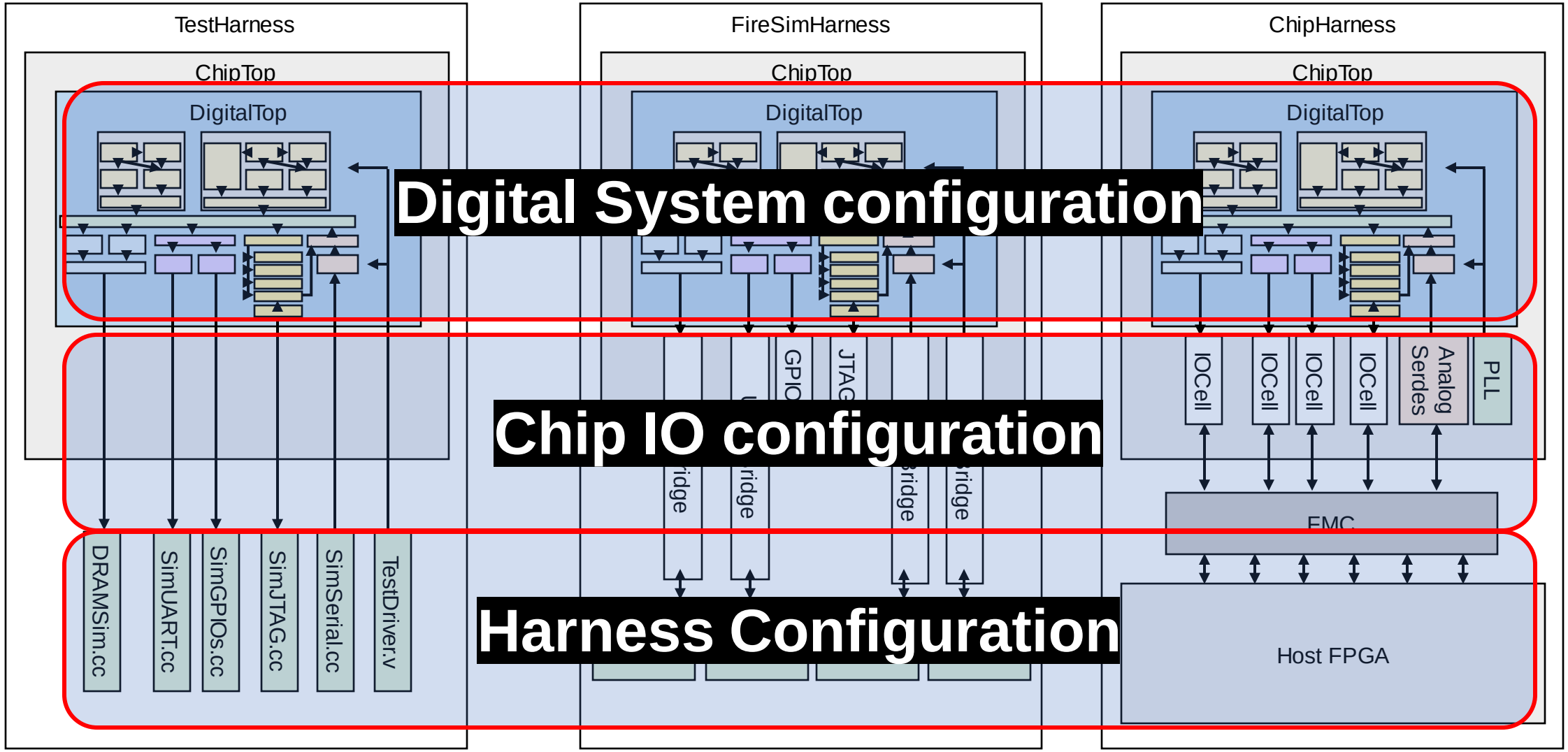
Multipurpose



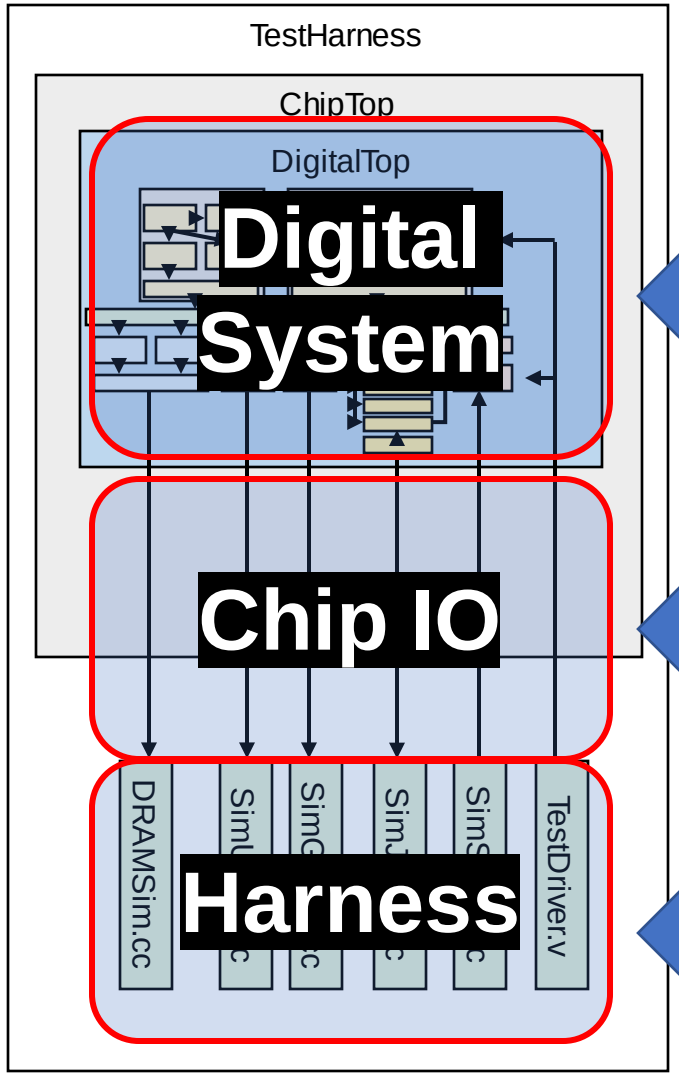




Multipurpose



A Complete Config



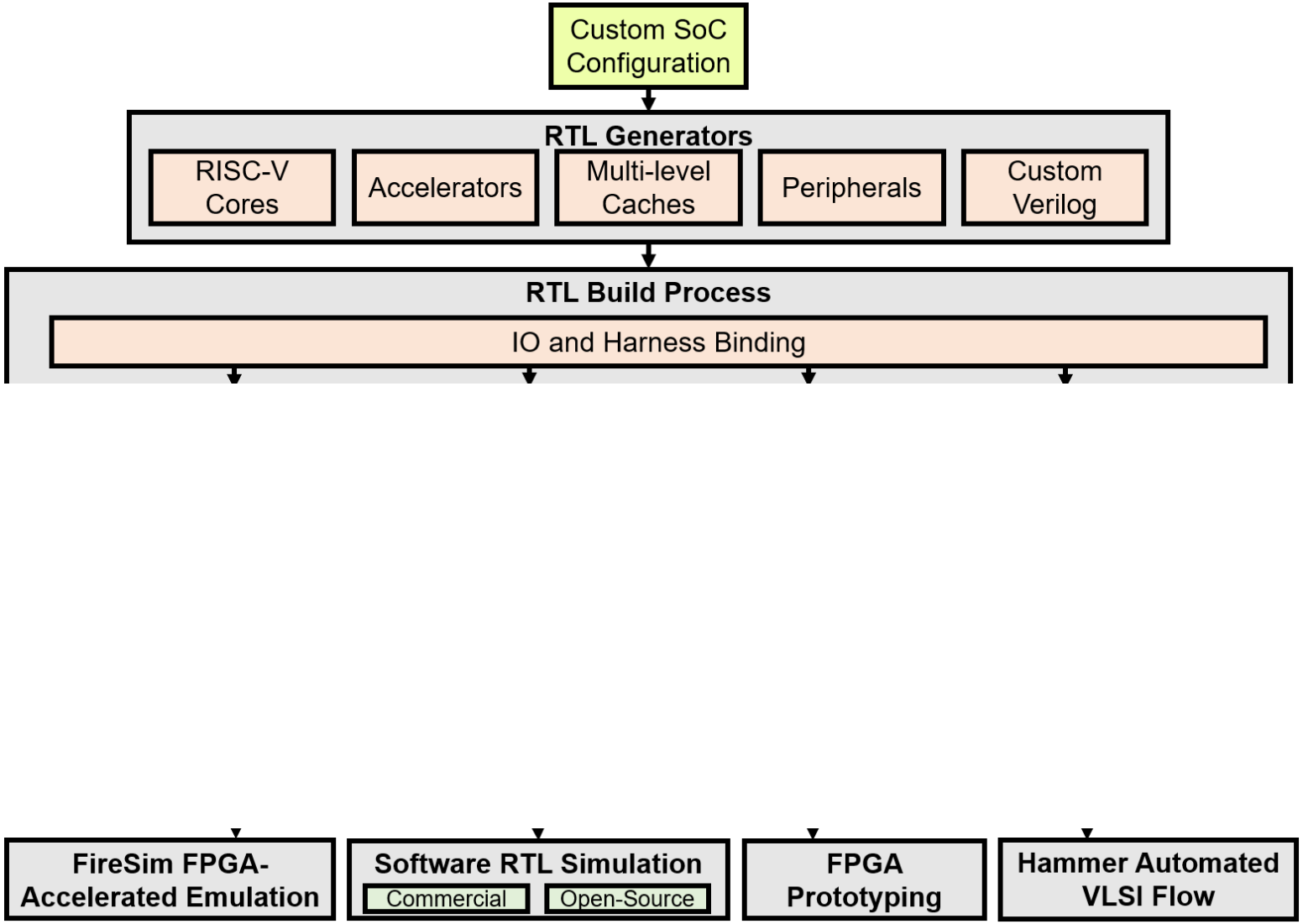
Digital Config

IO Binders

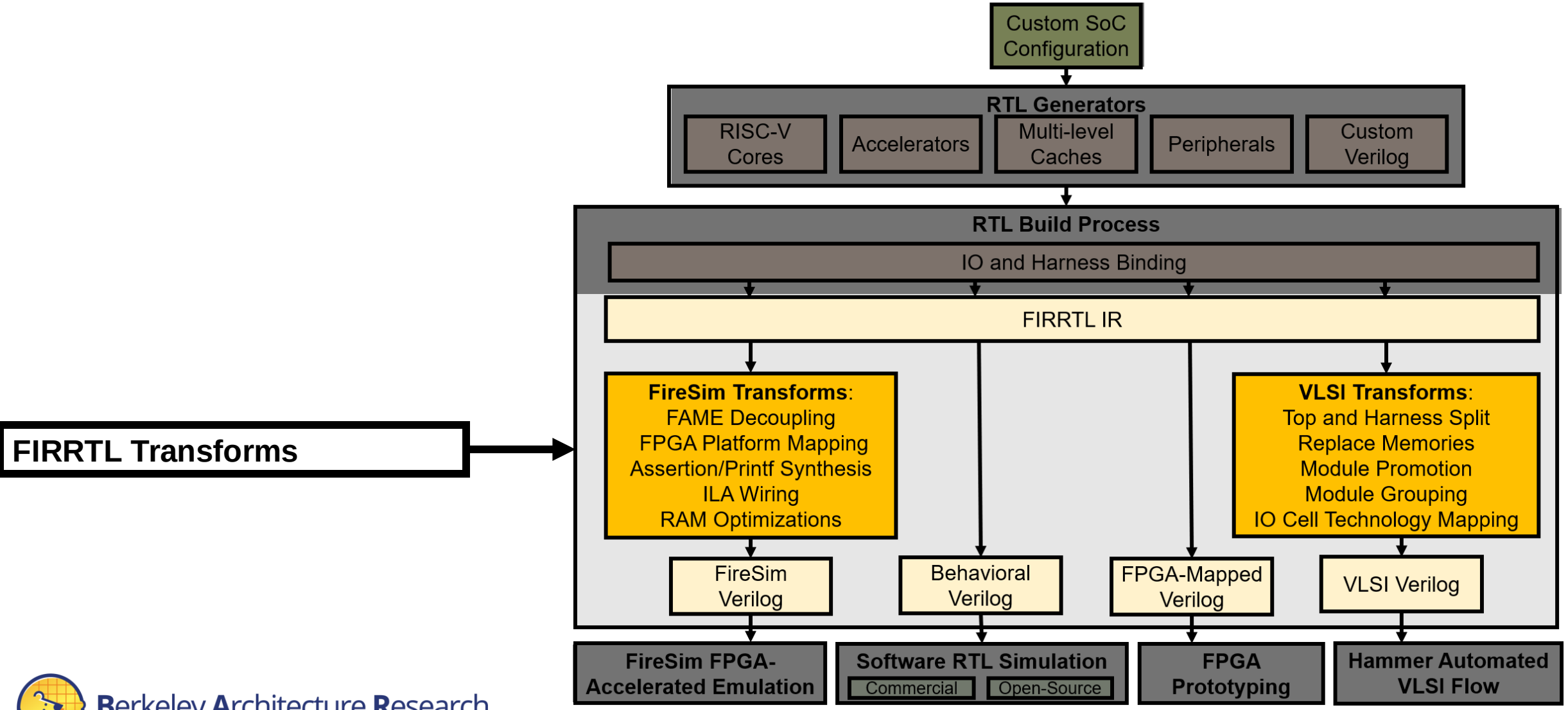
Harness Binders

```
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

)
```

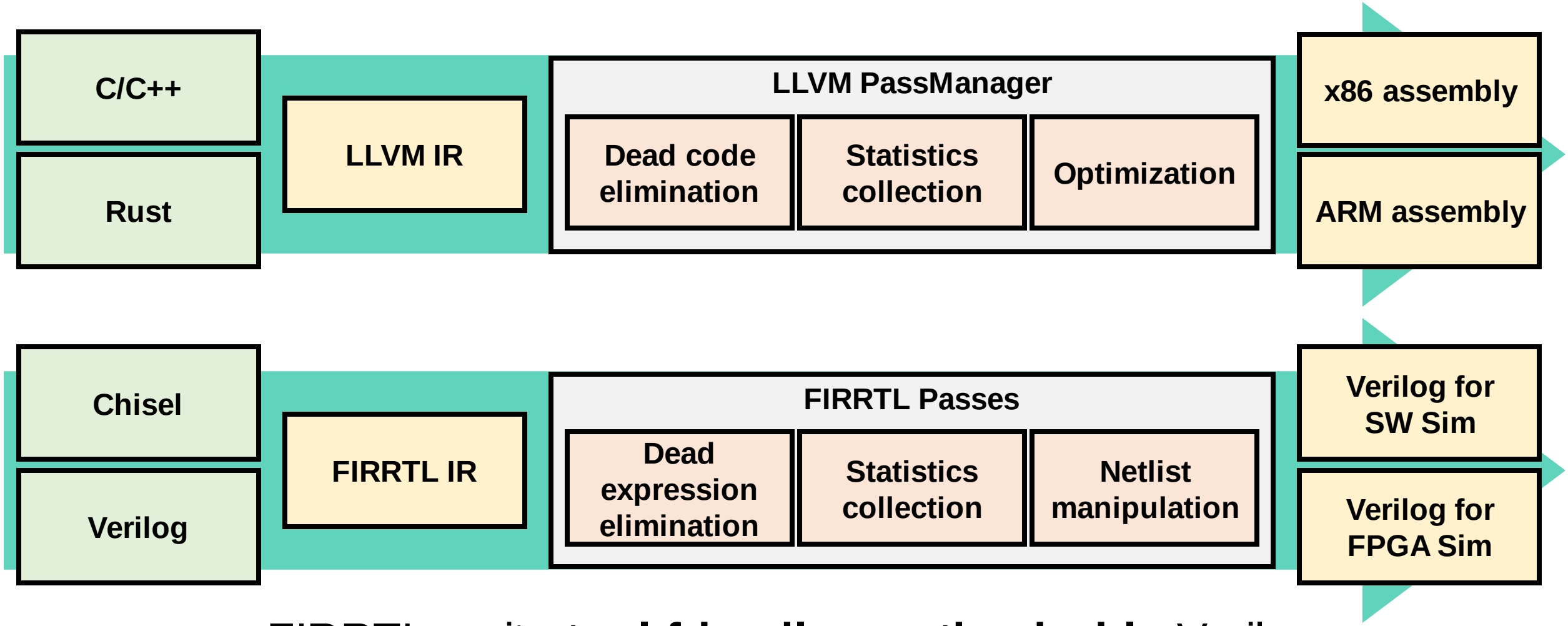
Chipyard Organization



Chipyard Organization



FIRRTL – LLVM for Hardware



FIRRTL emits **tool-friendly, synthesizable** Verilog

Chipyard Organization

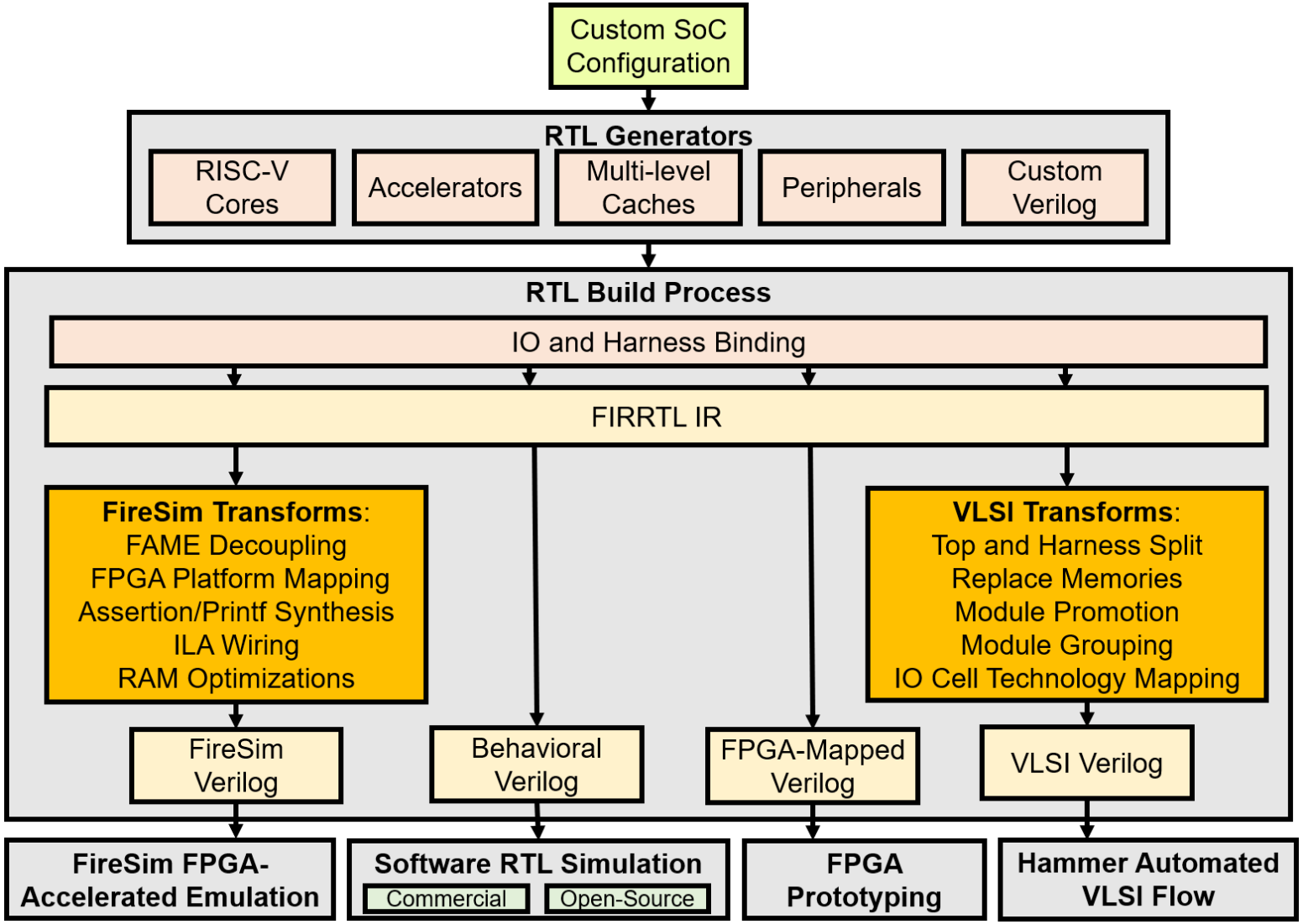
Configs: Describe parameterization of a multi-generator SoC

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

IOBinders/HarnessBinders: Enable configuring IO strategy and Harness features

FIRRTL Passes: Structured mechanism for supporting multiple flows

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

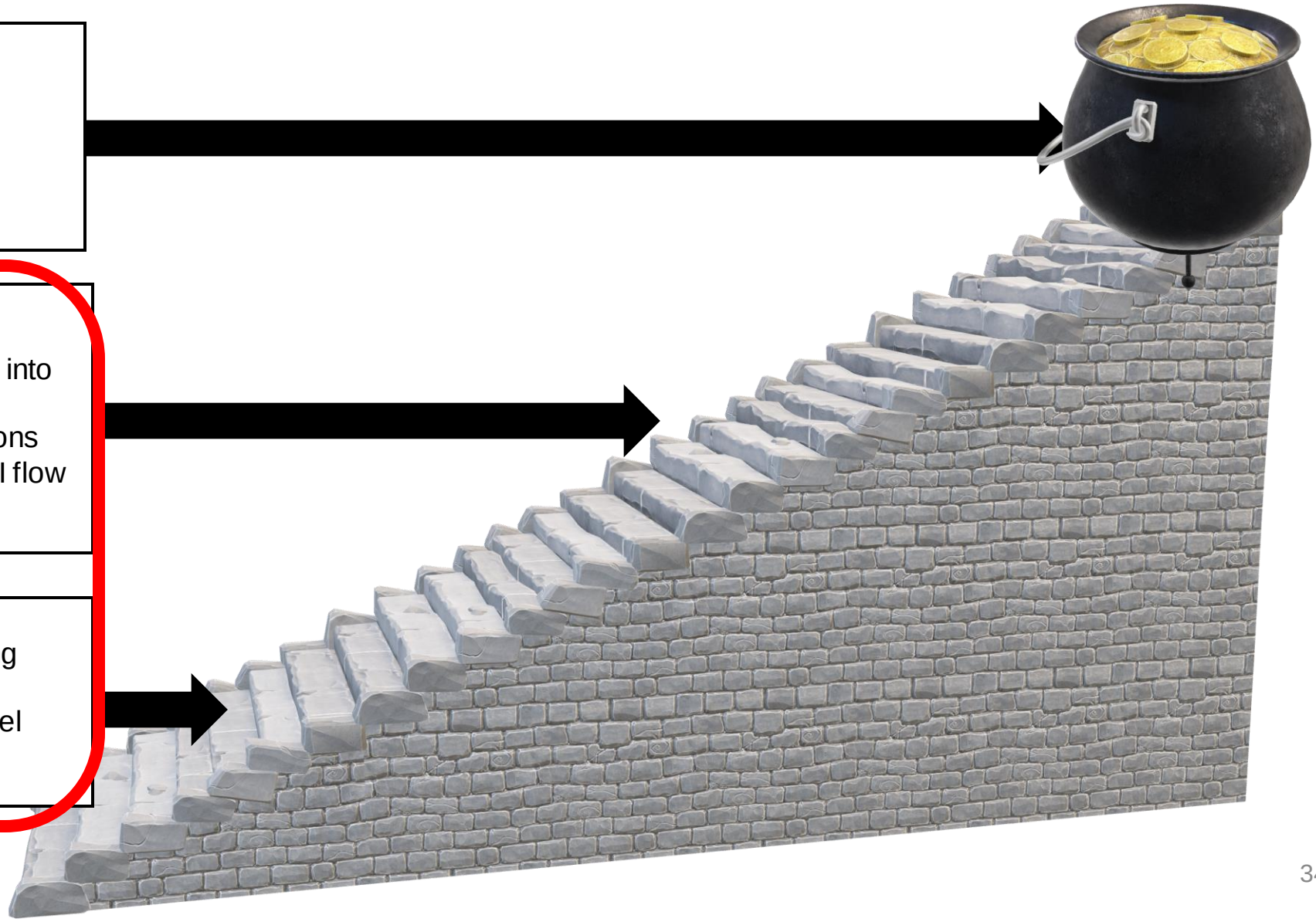


Chipyard Learning Curve

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



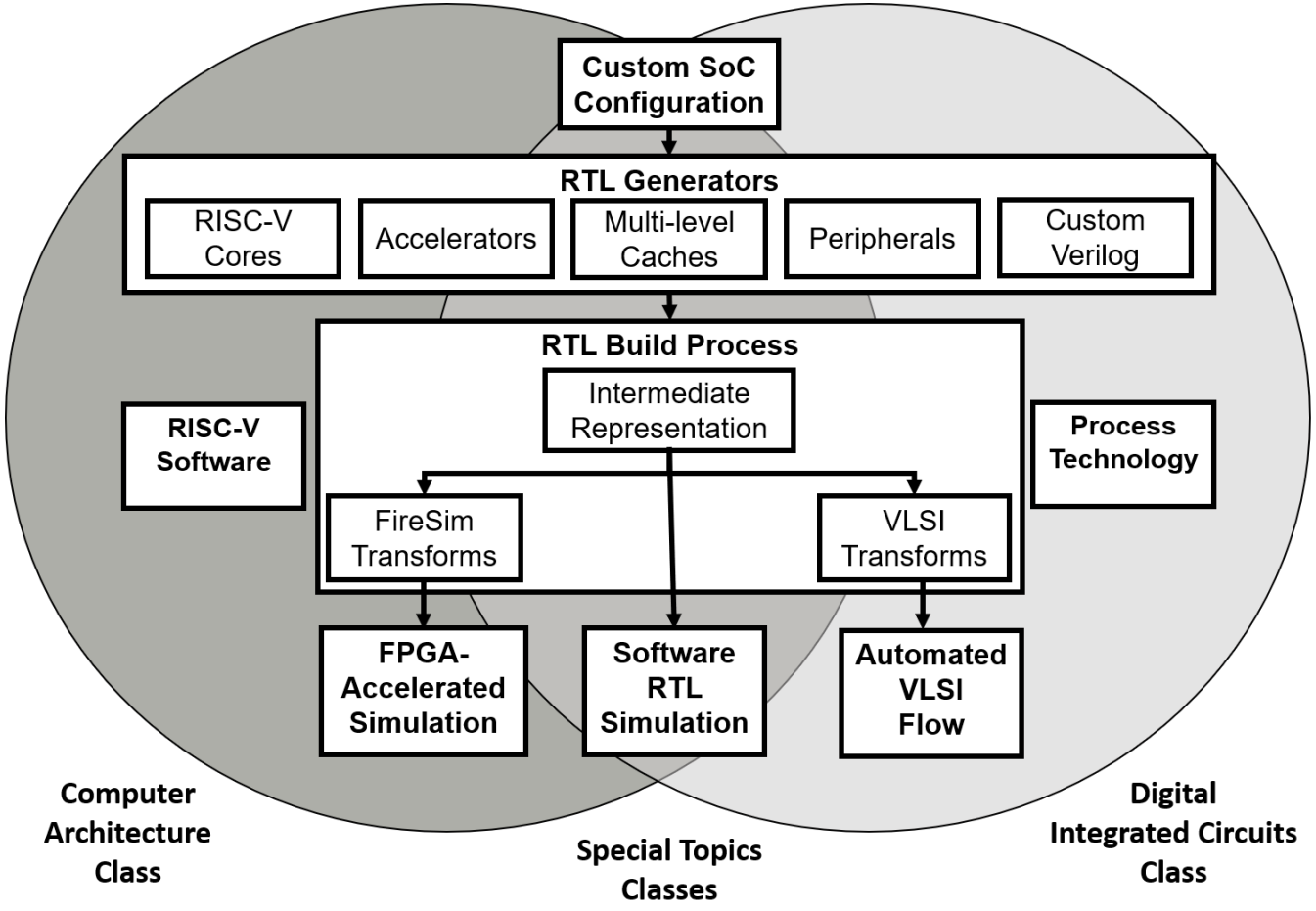
Chipyard is Education Friendly

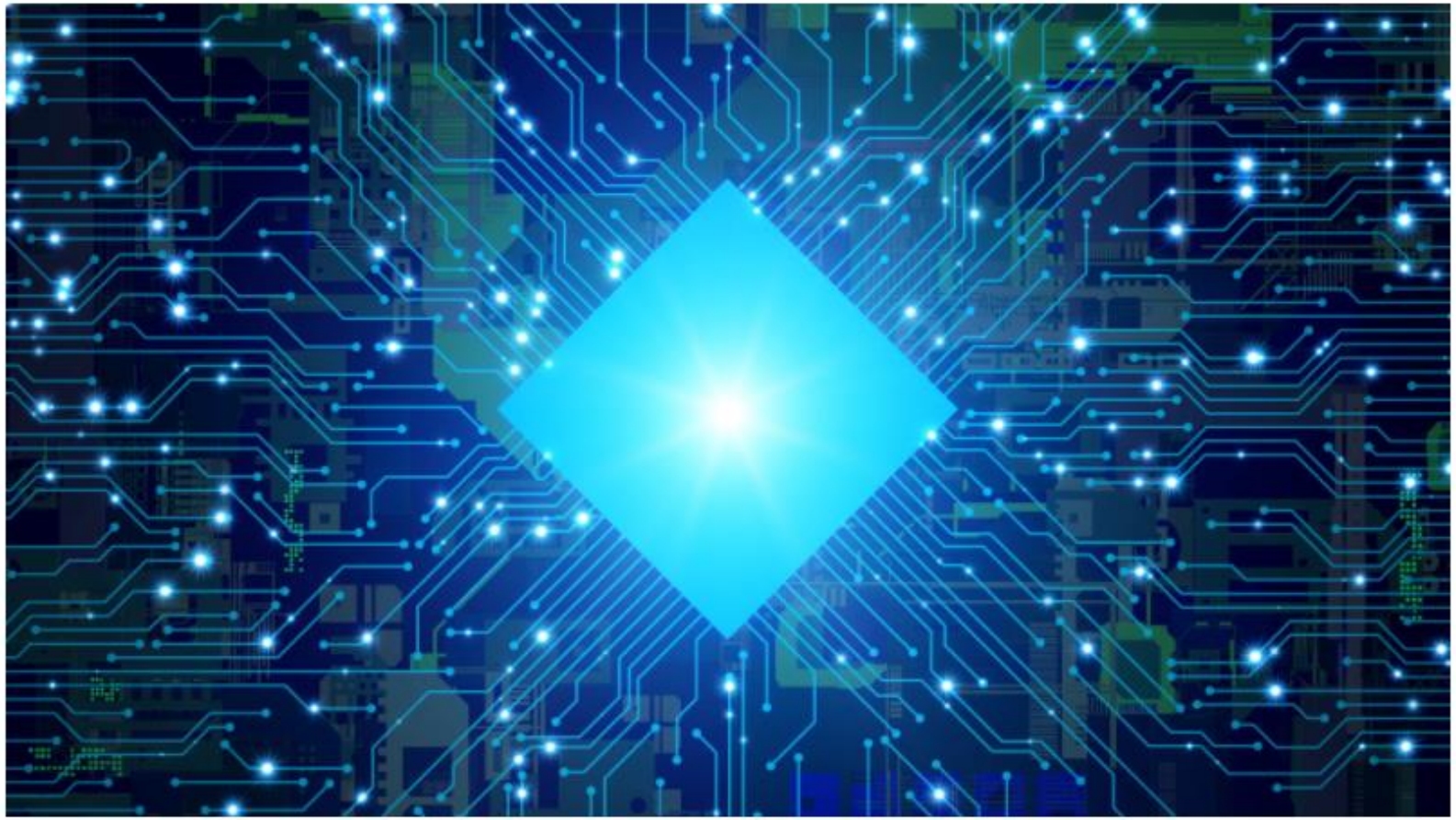
Proven in many Berkeley Architecture courses

- 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)





Berkeley Engineering students pull off novel chip design in a single semester. The class shows successful model for expanding entry into field of semiconductor design

Berkeley engineering students pull off novel chip design in a single semester

Class shows successful model for expanding entry into field of semiconductor design

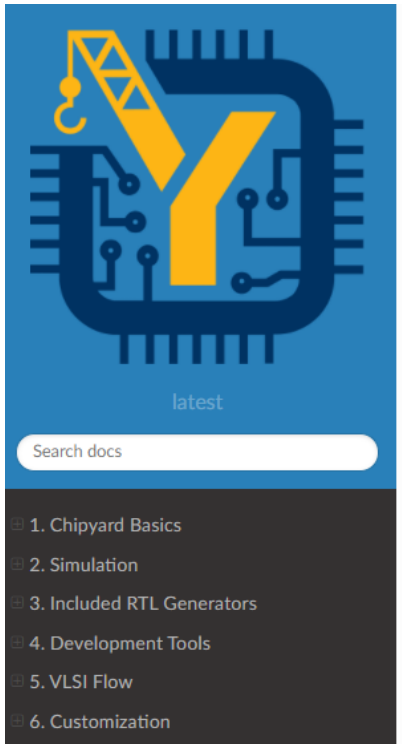
Chipyard is Research-Friendly

- Add new accelerators/custom instructions
- Modify OS/driver/software
- Perform design-space exploration across many parameters
- Test in software and FPGA-sim before tape-out

Chipyard is Community-Friendly

Documentation:

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

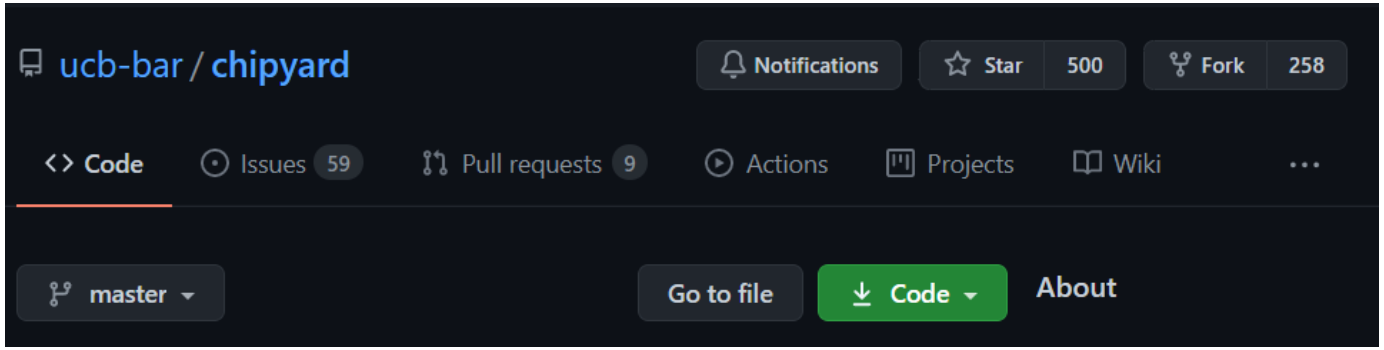


Mailing List:

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

Open-sourced:

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

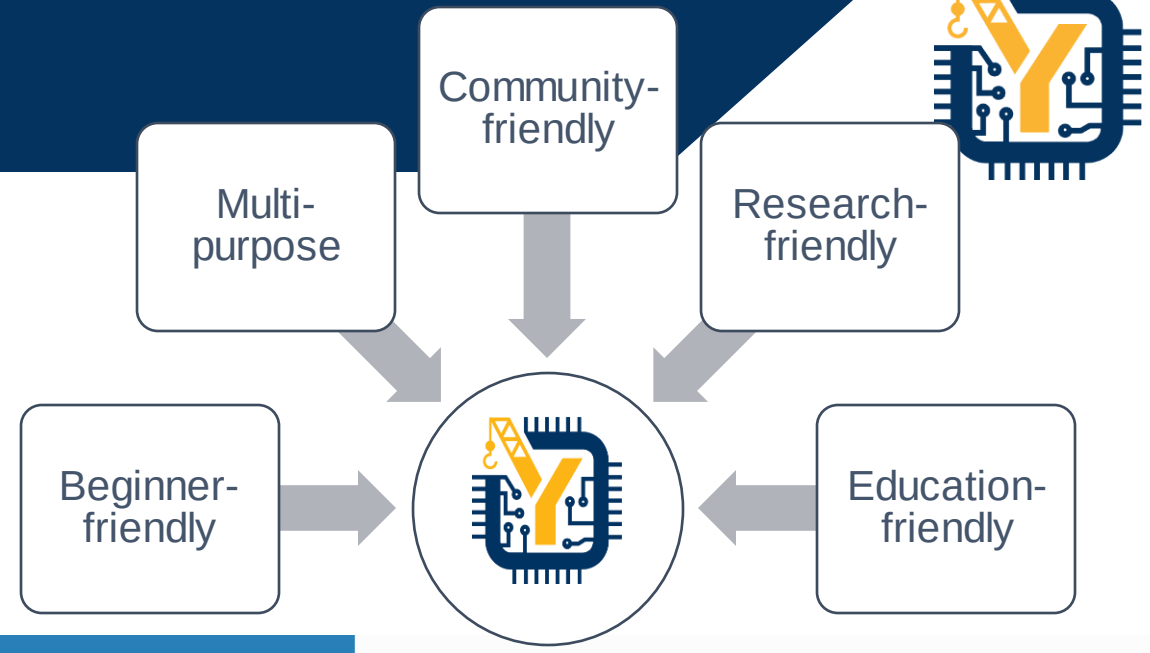


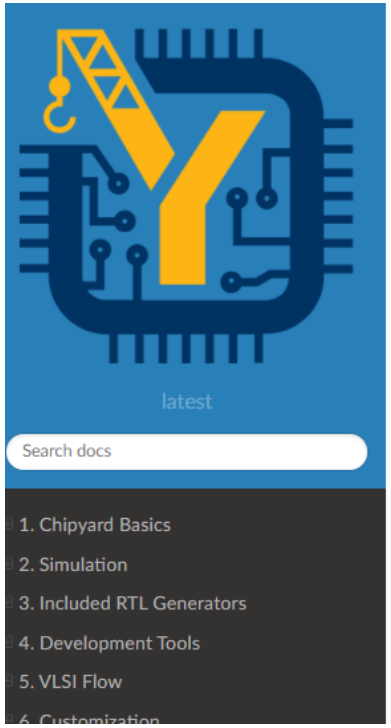
Conclusion

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

Questions?





Berkeley Architecture Research