RISC-V "Rocket Chip" SoC Generator in Chisel

Yunsup Lee

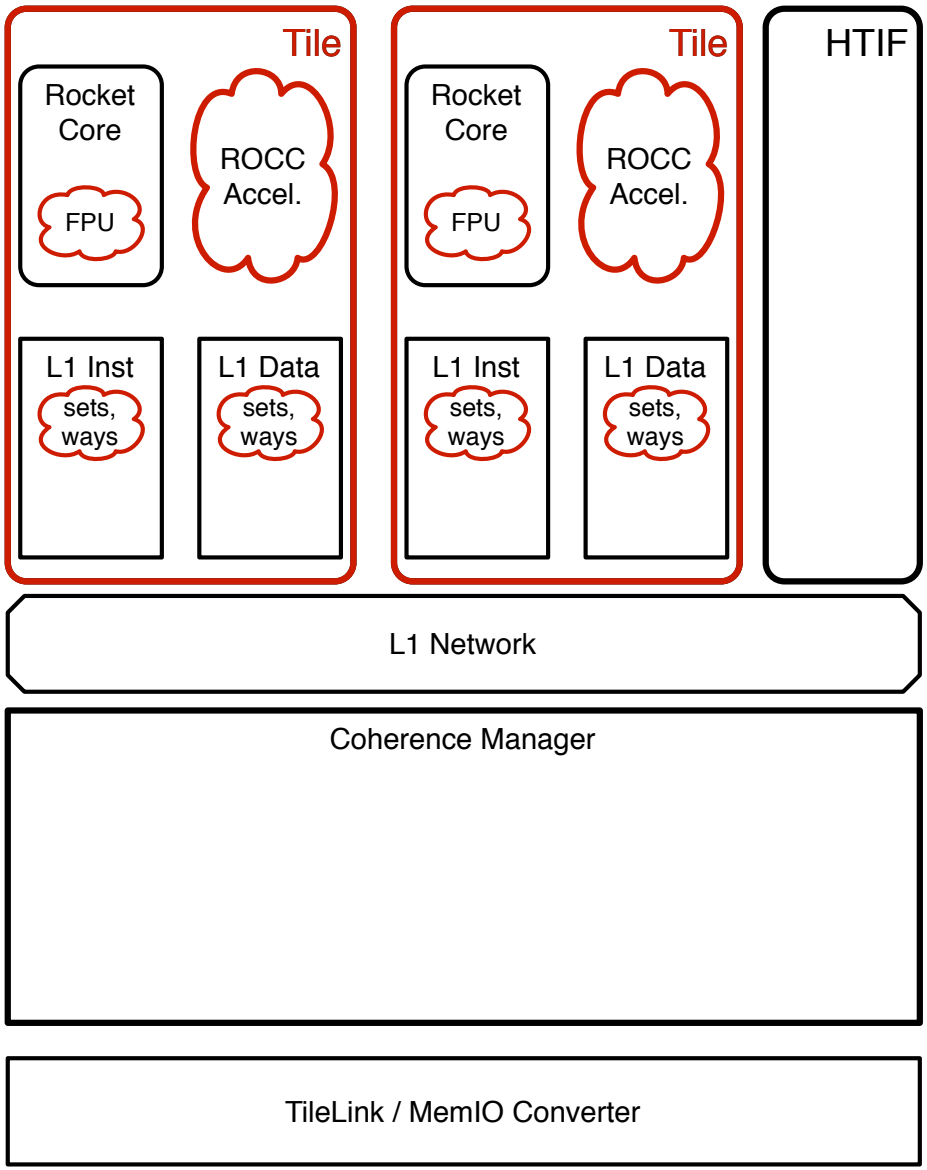
UC Berkeley

yunsup@eecs.berkeley.edu

What is the Rocket Chip SoC Generator?

- Parameterized SoC generator written in Chisel
- Generates Tiles
  - (Rocket) Core + Private Caches
- Generates Uncore (Outer Memory System)
  - Coherence Agent
  - Shared Caches
  - DMA Engines
  - Memory Controllers
- Glues all the pieces together

"Rocket Chip" SoC Generator



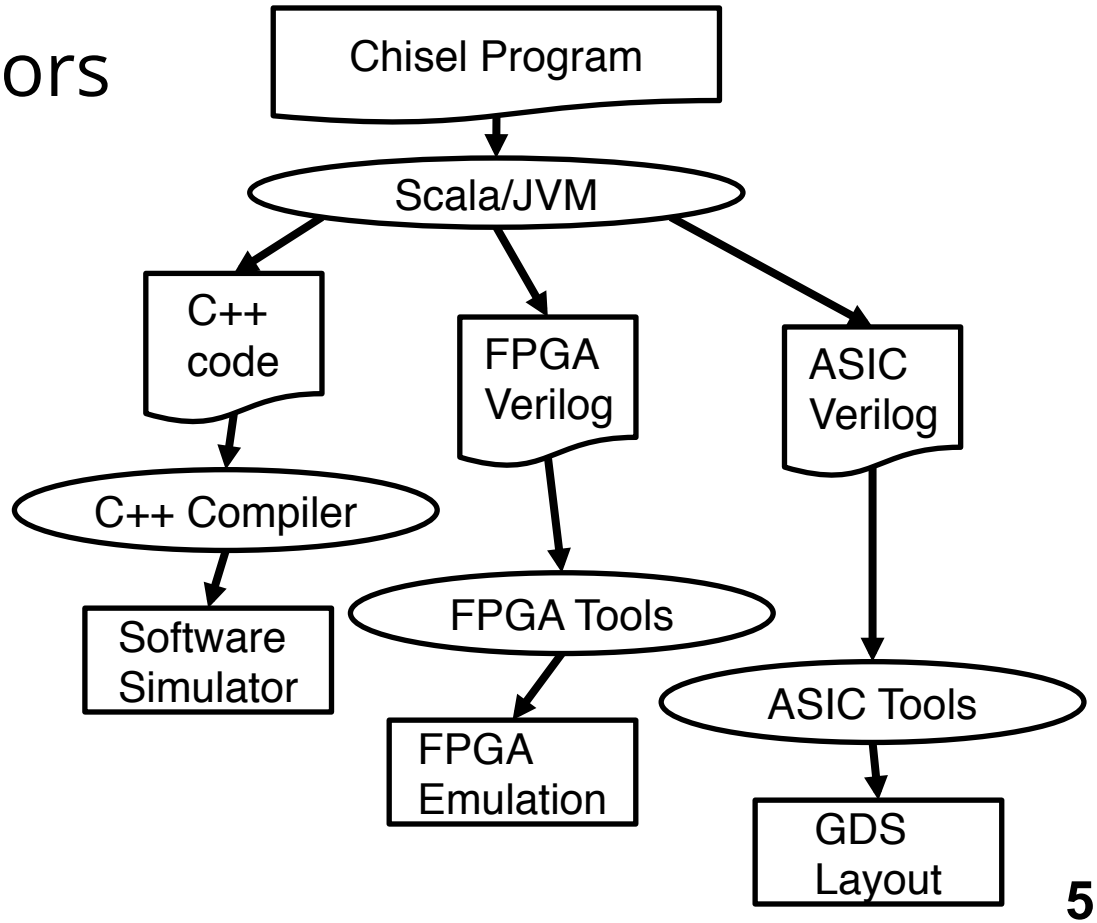
- Generates n Tiles
  - (Rocket) Core
  - RoCC Accelerator
  - L1 I$
  - L1 D$
- Generates HTIF
  - Host DMA Engine
- Generates Uncore
  - L1 Crossbar
  - Coherence Manager
  - Exports MemIO Interface

Why SoC Generators?

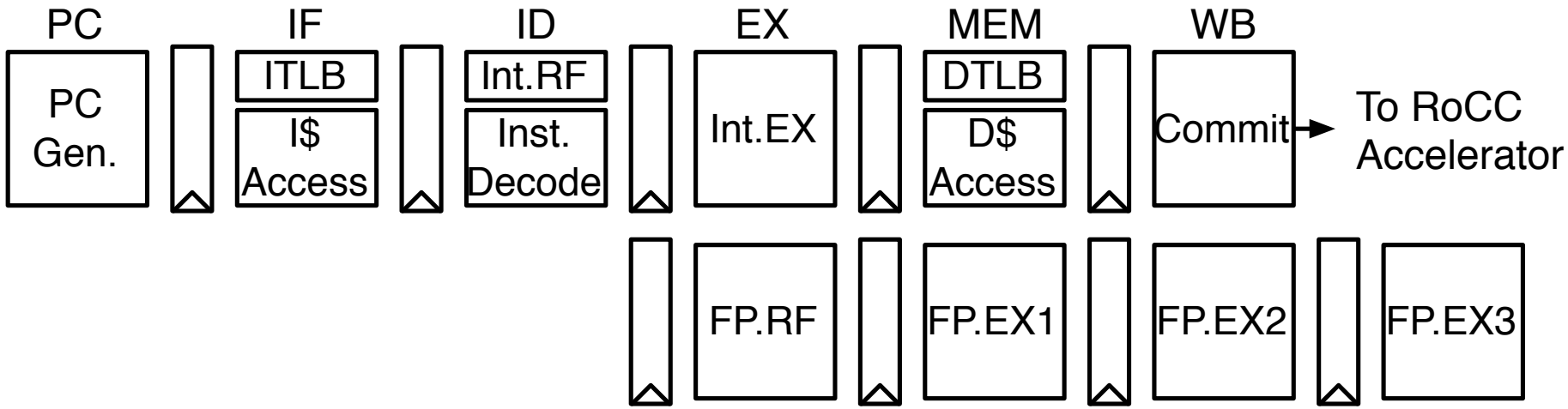
- Helps tune the design under different performance, power, area constraints, and diverse technology nodes
- Parameters include:
  - number of cores
  - instantiation of floating-point units, vector units
  - cache sizes, associativity, number of TLB entries, cache-coherence protocol
  - number of floating-point pipeline stages
  - width of off-chip I/O, and more

Why Chisel?

- RTL generator written in Chisel
  - HDL embedded in Scala
- Full power of Scala for writing generators
  - object-oriented programming
  - functional programming



Rocket Scalar Core



- 64-bit 5-stage single-issue in-order pipeline
- Design minimizes impact of long clock-to-output delays of compiler-generated RAMs
- 64-entry BTB, 256-entry BHT, 2-entry RAS
- MMU supports page-based virtual memory
- IEEE 754-2008-compliant FPU
  - Supports SP, DP FMA with hw support for subnormals

ARM Cortex-A5 vs. RISC-V Rocket

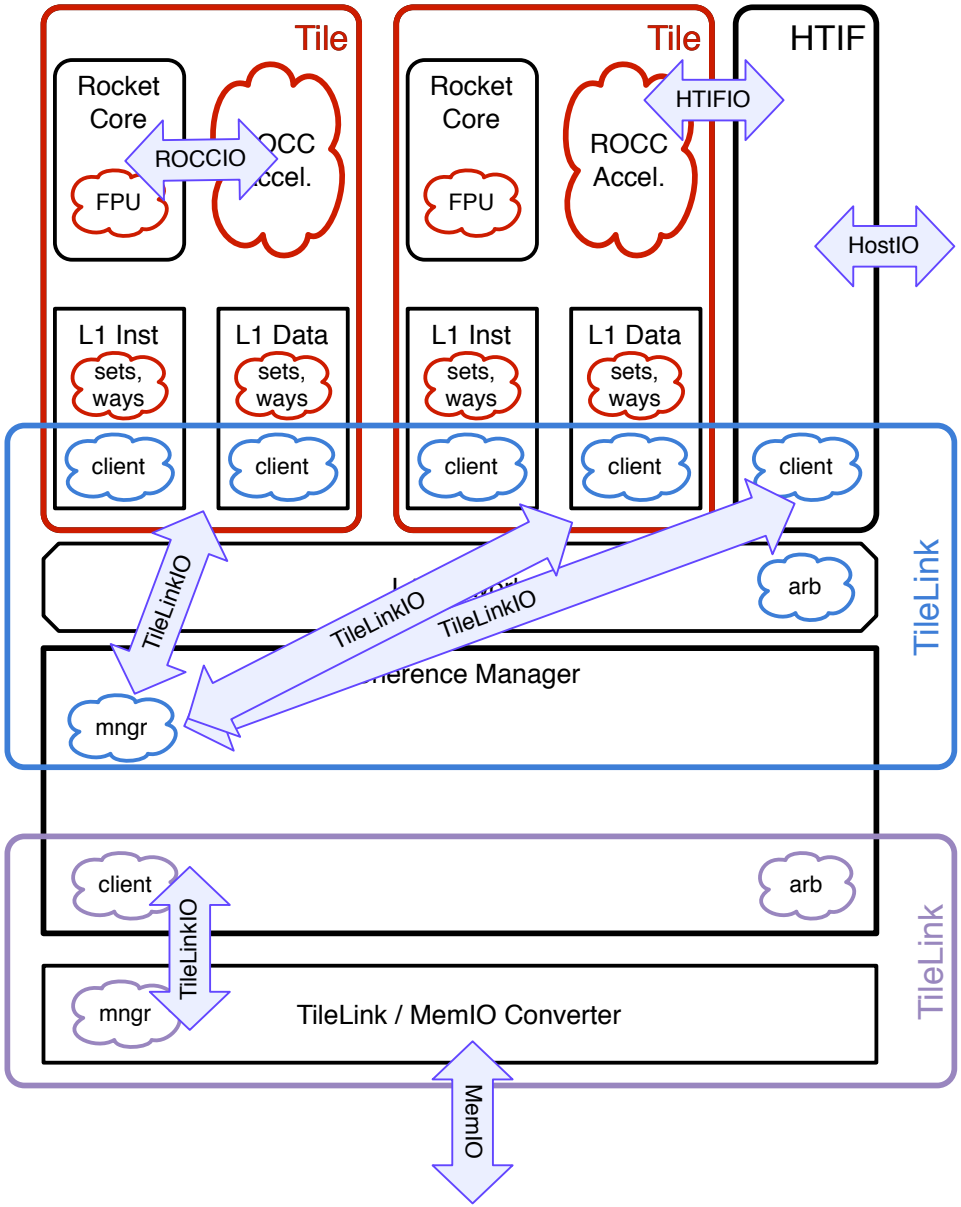
Category	ARM Cortex-A5	RISC-V Rocket
ISA	32-bit ARM v7	64-bit RISC-V v2
Architecture	Single-Issue In-Order	Single-Issue In-Order 5-stage
Performance	1.57 DMIPS/MHz	1.72 DMIPS/MHz
Process	TSMC 40GPLUS	TSMC 40GPLUS
Area w/o Caches	0.27 mm^2	0.14 mm^2
Area with 16K Caches	0.53 mm^2	0.39 mm^2
Area Efficiency	2.96 DMIPS/MHz/mm^2	4.41 DMIPS/MHz/mm^2
Frequency	>1GHz	>1GHz
Dynamic Power	<0.08 mW/MHz	0.034 mW/MHz

- PPA reporting conditions
  - 85% utilization, use Dhrystone for benchmark, frequency/power at TT 0.9V 25C, all regular VT transistors
- 10% higher in DMIPS/MHz, 49% more area-efficient

HTIF: Host-Target Interface

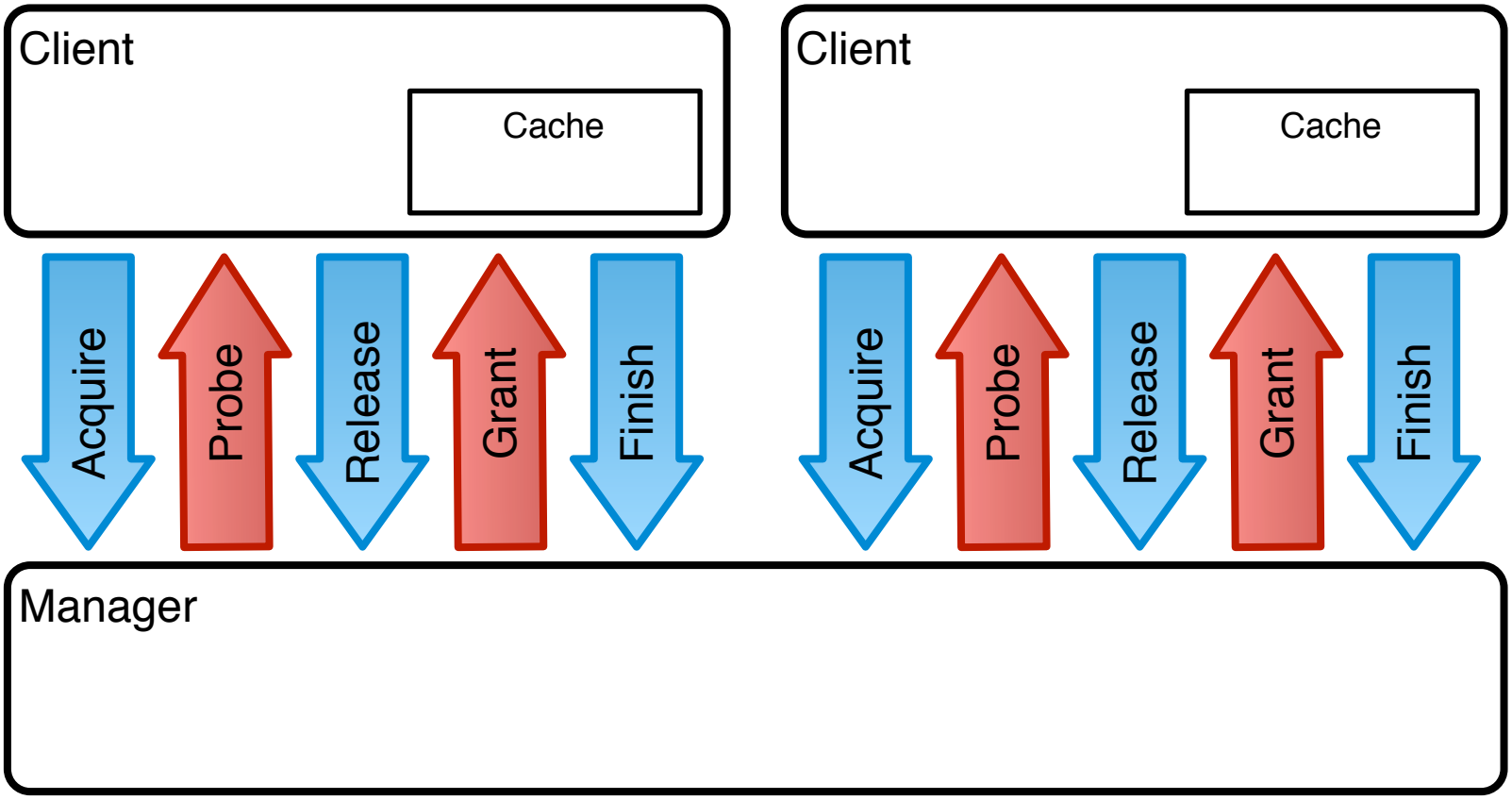
- UC Berkeley specific block mainly used to emulate devices for simple test chips
  - Emulates system calls, console, block devices, frame buffer, network devices
  - No need for this block once the SoC has actual devices on the target machine
- Consider it as a “host DMA engine”
- A port for for host system to read/write
  - Core CSRs (control and status registers)
  - Target Memory

Important Interfaces in the Rocket Chip



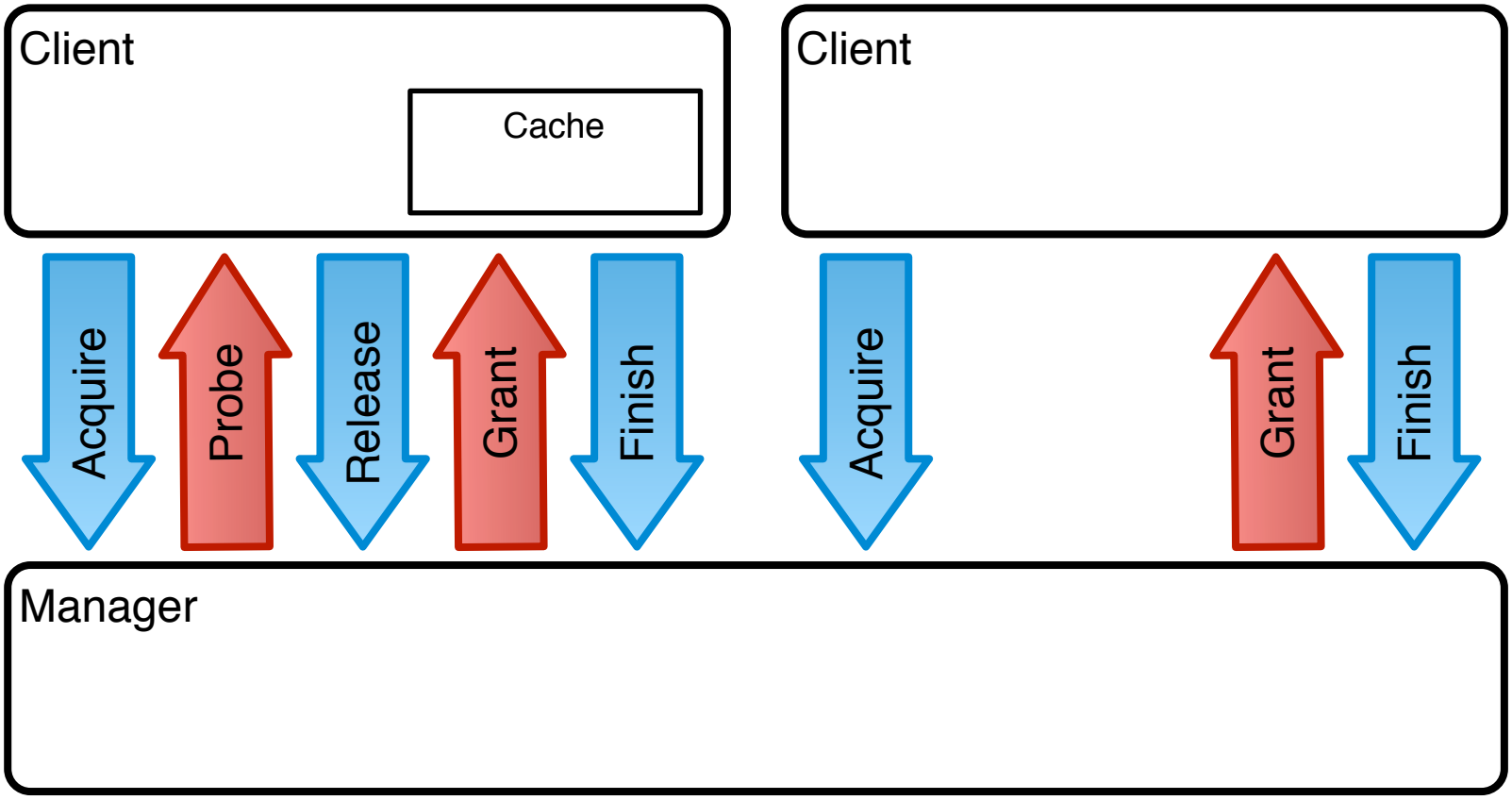
- ROCCIO
  - Interface between Rocket/Accelerator
- HTIFIO
  - Read/Write CSRs
- TileLinkIO
  - Coherence Fabric
- MemIO
  - Simple AXI-like memory interface
- HostIO
  - Host Interface to HTIF

TileLinkIO



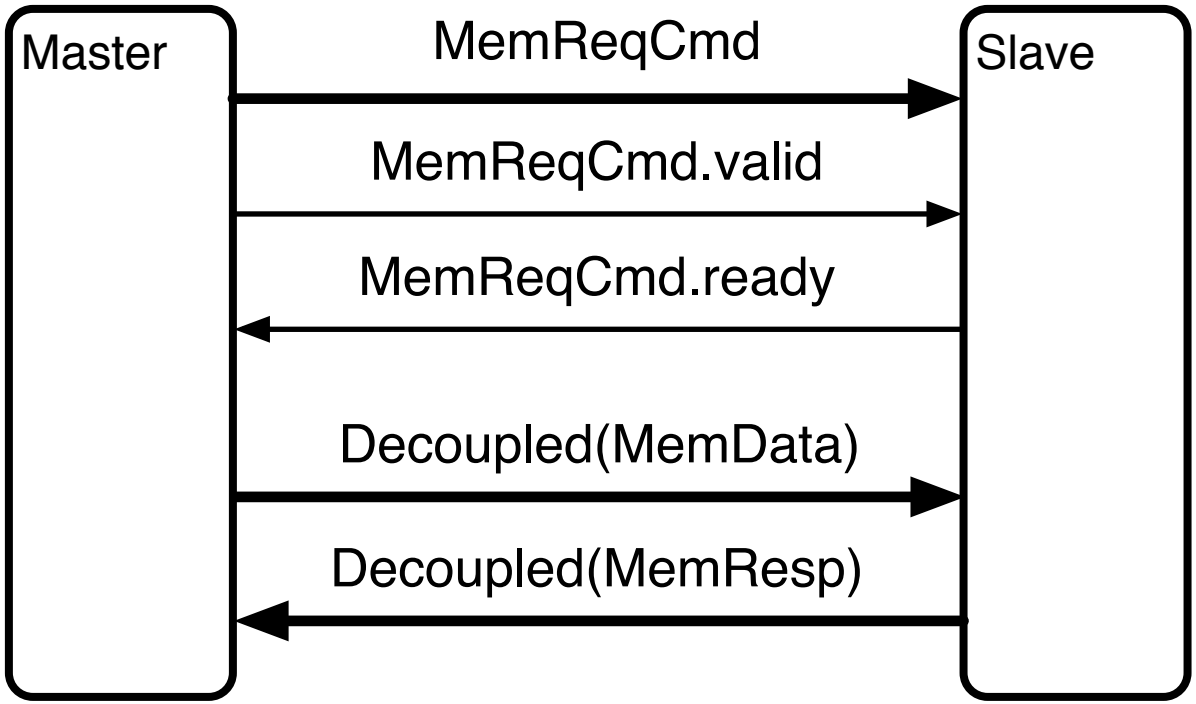
– TileLinkIO consists of Acquire, Probe, Release, Grant, Finish

UncachedTileLinkIO



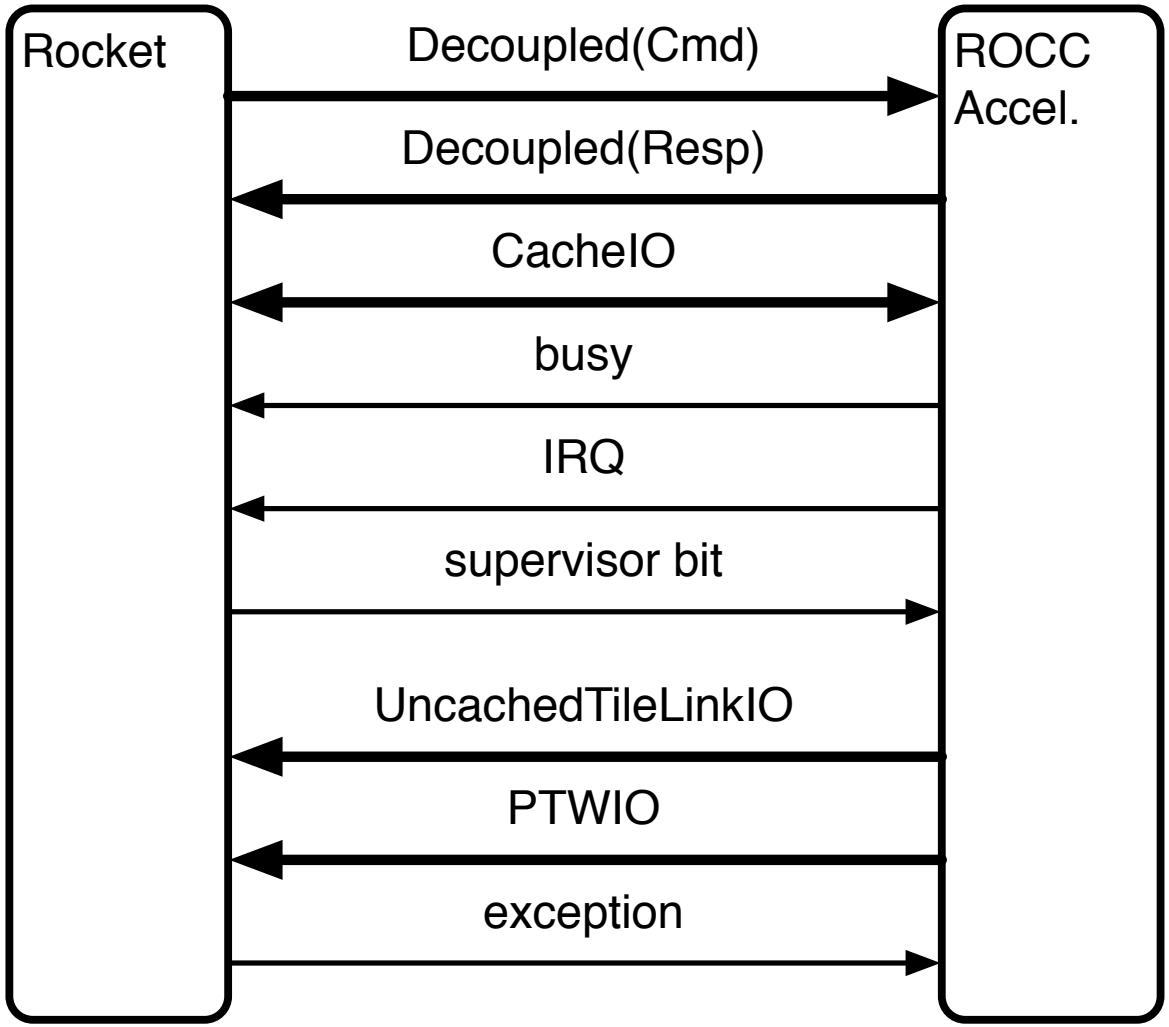
- UncachedTileLinkIO consists of Acquire, Grant, Finish
- Convertors for TileLinkIO/UncachedTileLinkIO in uncore library

MemIO



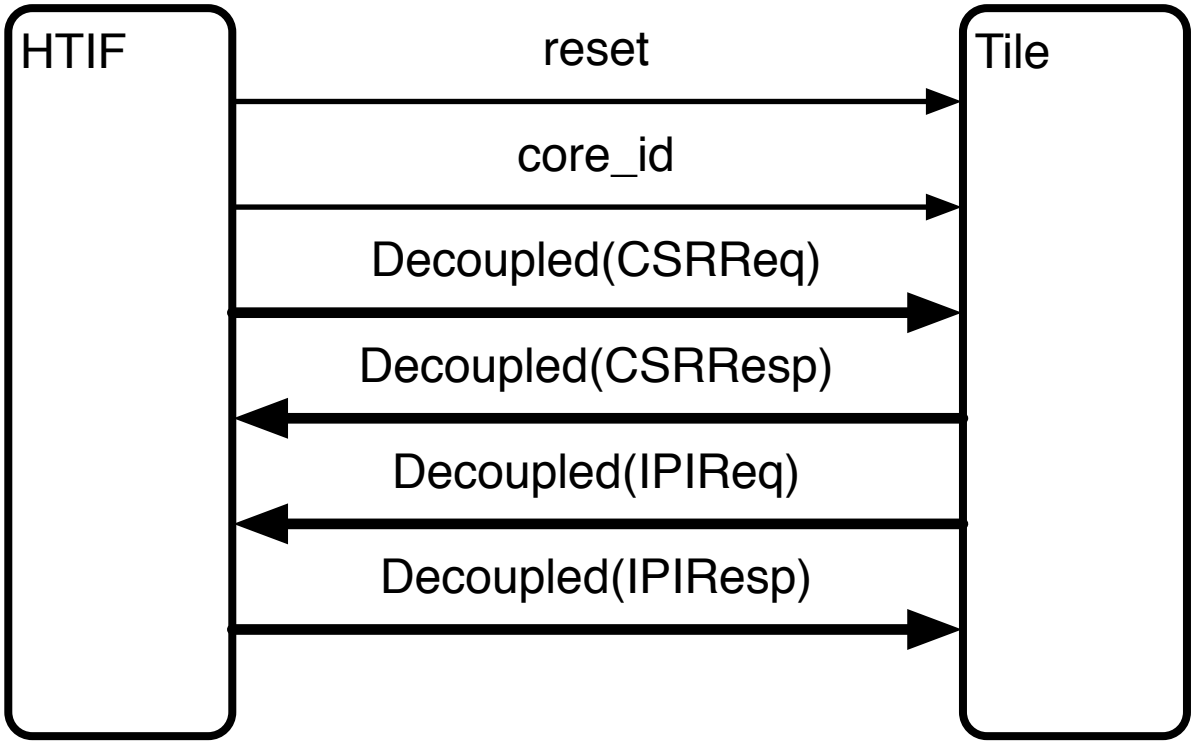
- MemReqCmd consists of addr, rw (write=true), tag
- MemData consists of 128 bit data payload
- MemResp consists of 128 bit data payload, tag
- Decoupled(interface) means an interface with ready/valid signals

ROCCIO



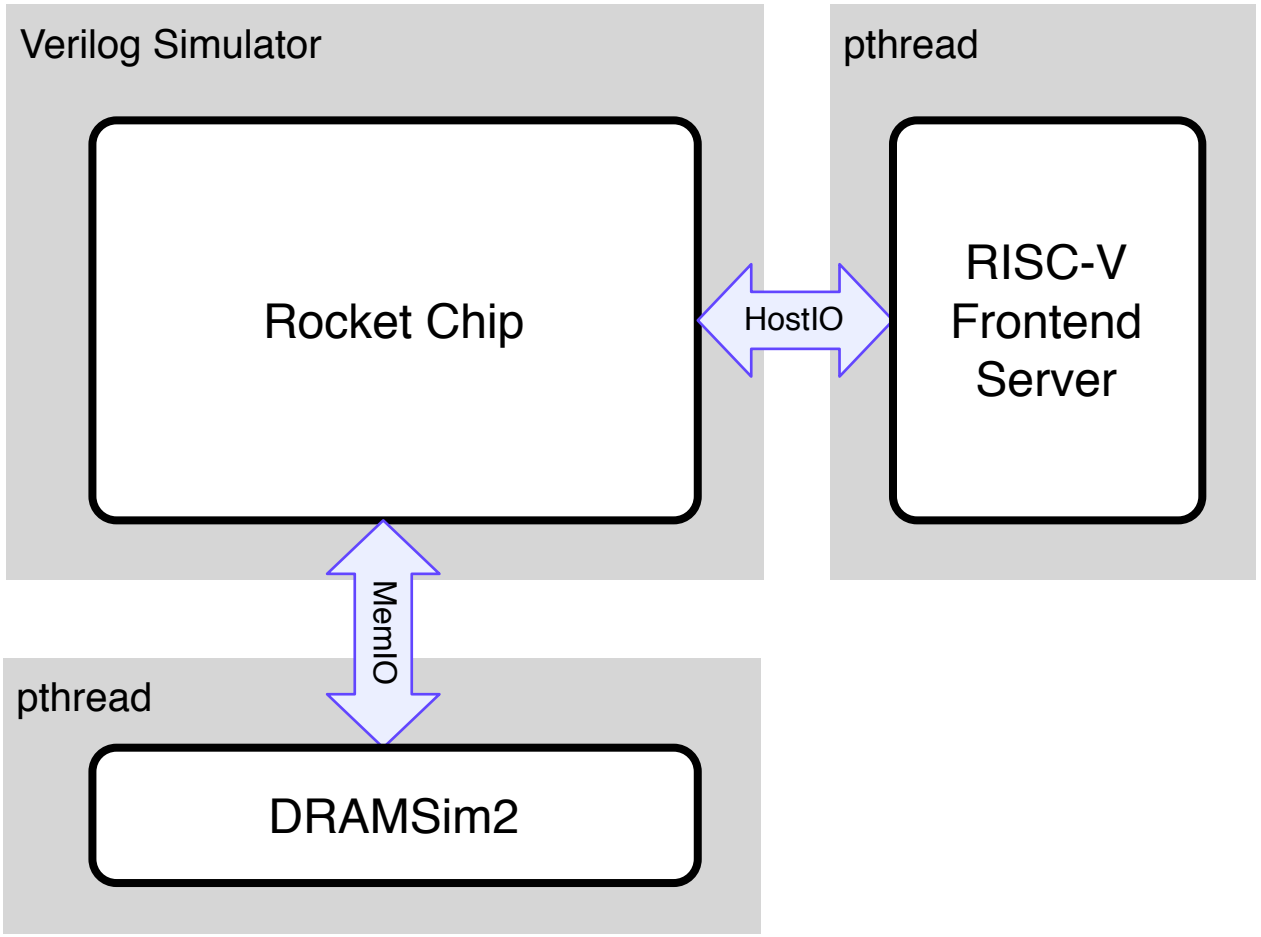
- Rocket sends coprocessor instruction via the Cmd interface
- Accelerator responds through Resp interface
- Accelerator sends memory requests to L1D$ via CacheIO
- busy bit for fences
- IRQ, S, exception bit used for virtualization
- UncachedTileLinkIO for instruction cache on accelerator
- PTWIO for page-table walker ports on accelerator

HTIFIO

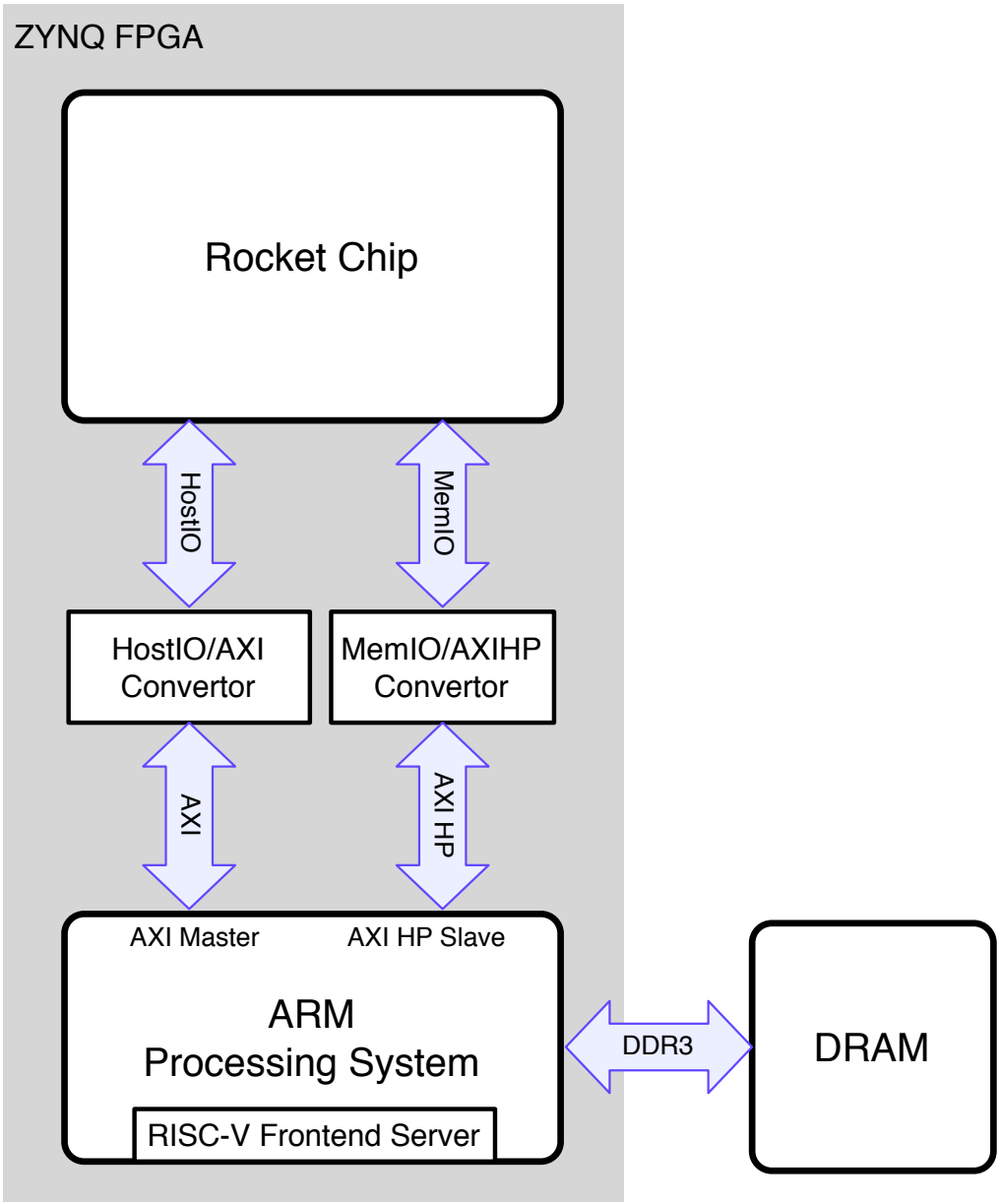


- reset signal and core_id routed from HTIF (historical reasons nothing technical)
- CSR Read/Write requests go through CSRReq/CSRResp
- IPI Requests go through IPIReq/IPIResp
- HTIFIO likely to be modified in the near future

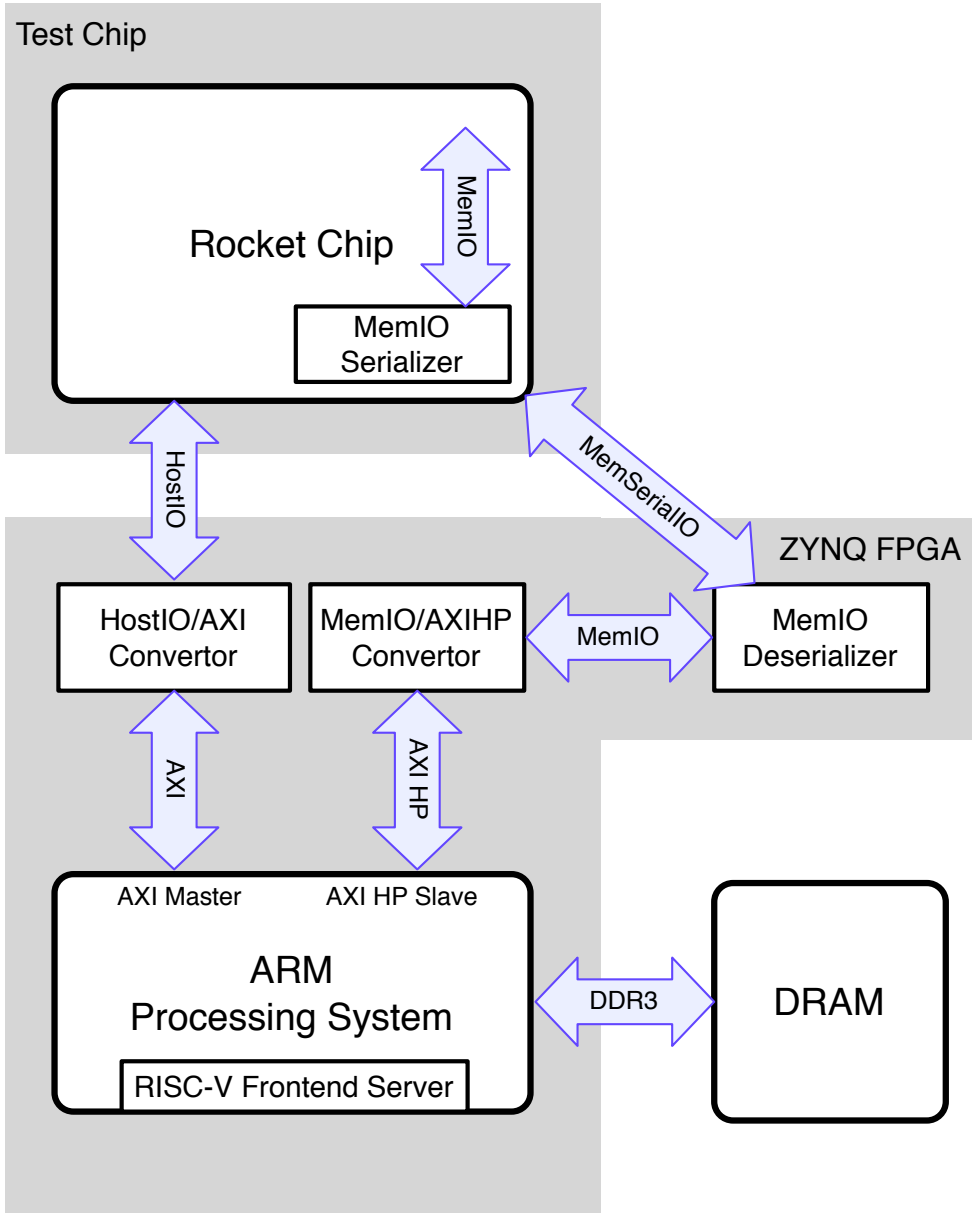
Rocket Chip C++ Emulator Setup



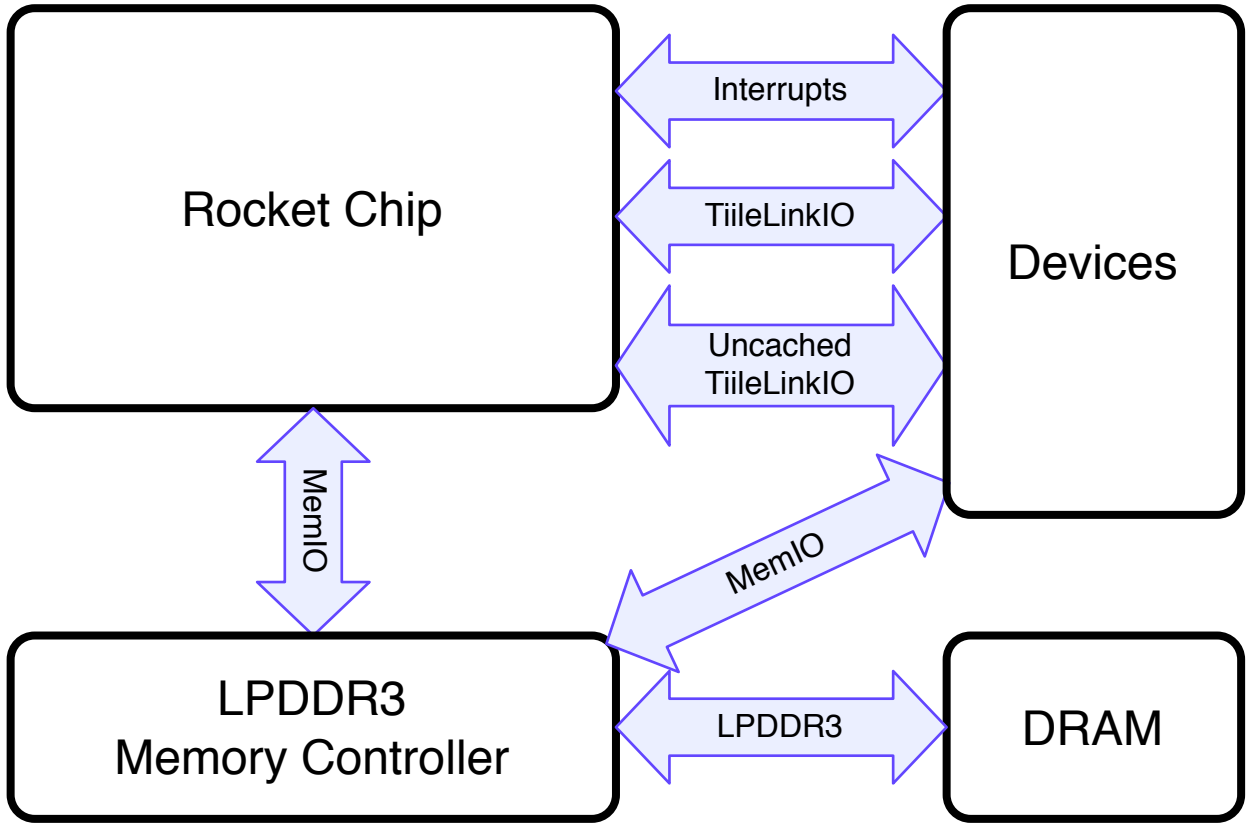
Rocket Chip FPGA Setup



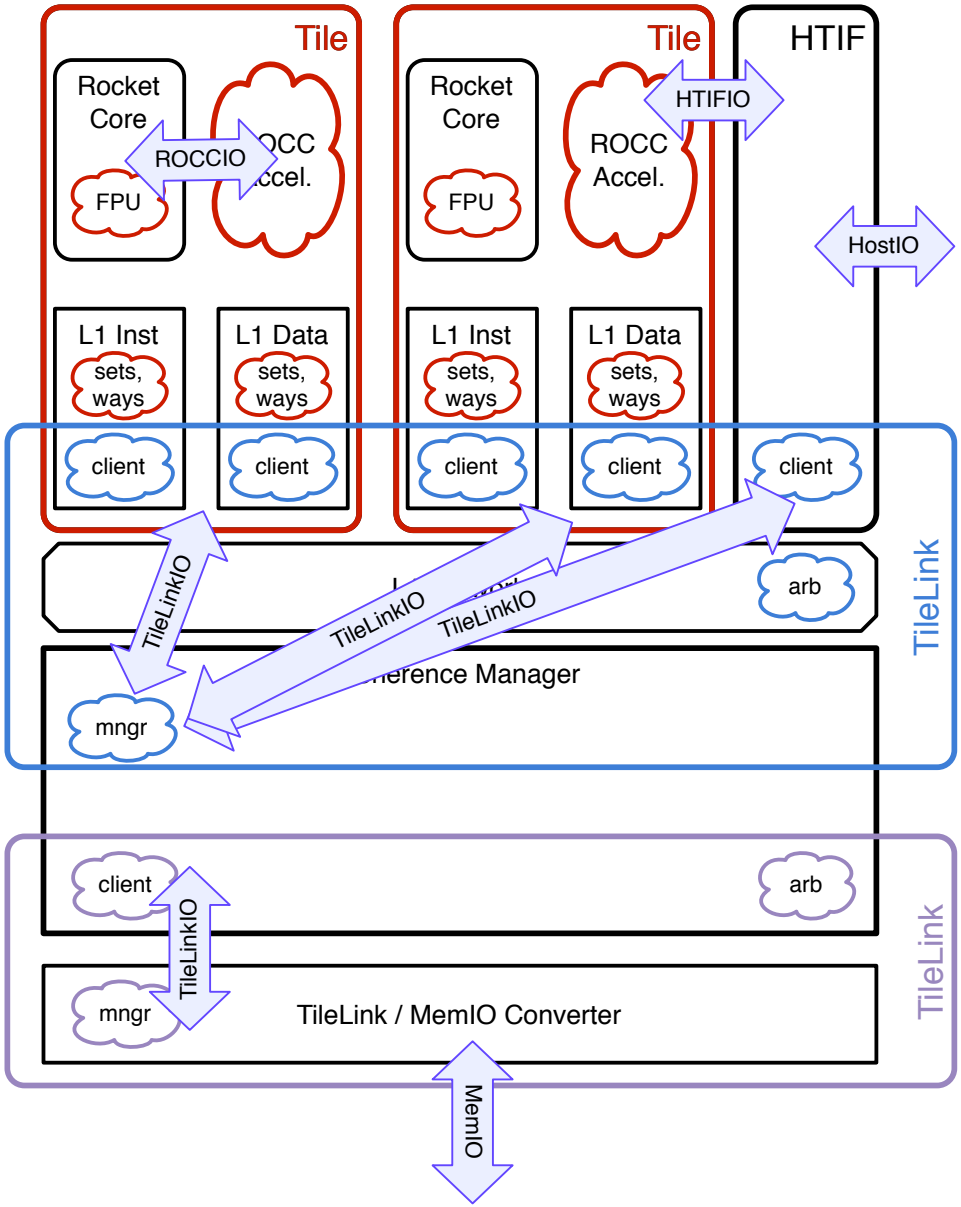
Rocket Chip Berkeley Test Chip Setup



Rocket Chip "SoC" Setup

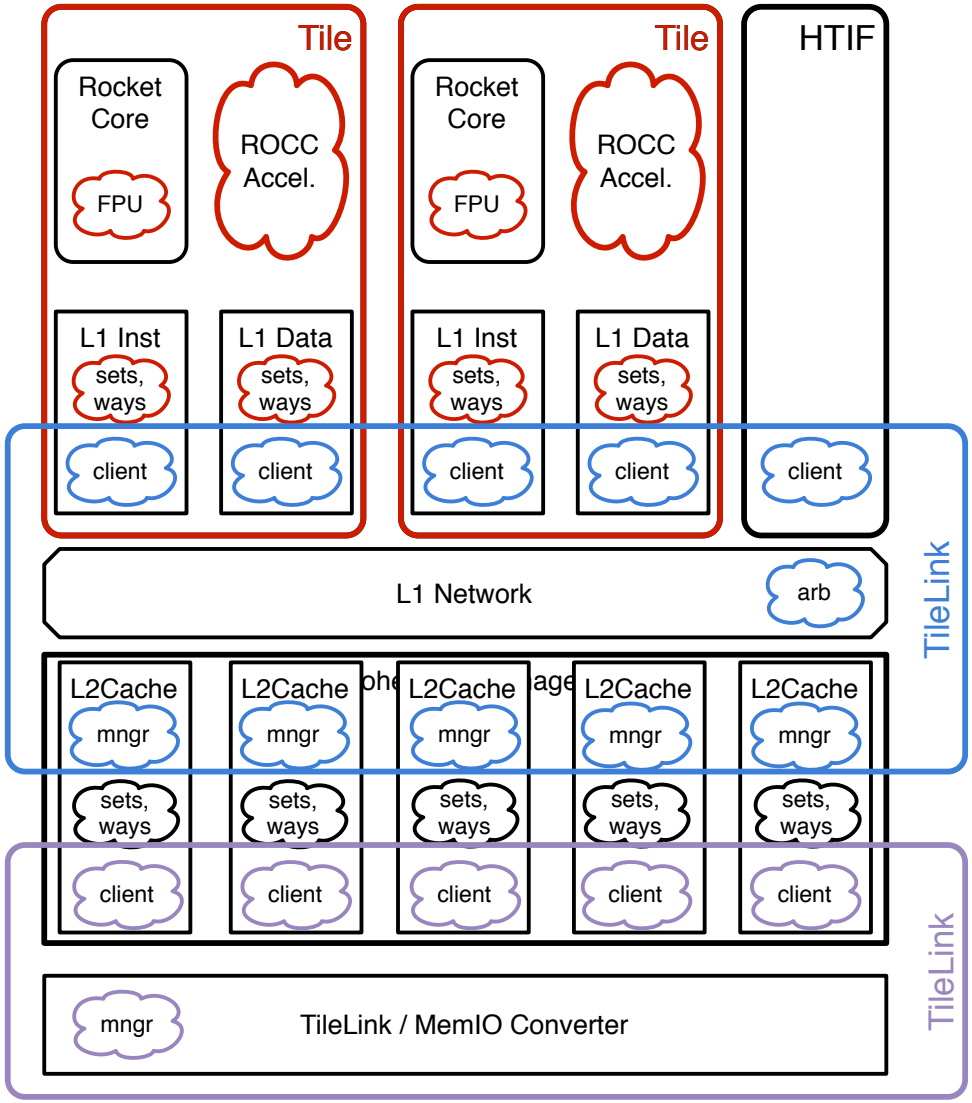


Who should use the Rocket Chip Generator? People who would like to develop …



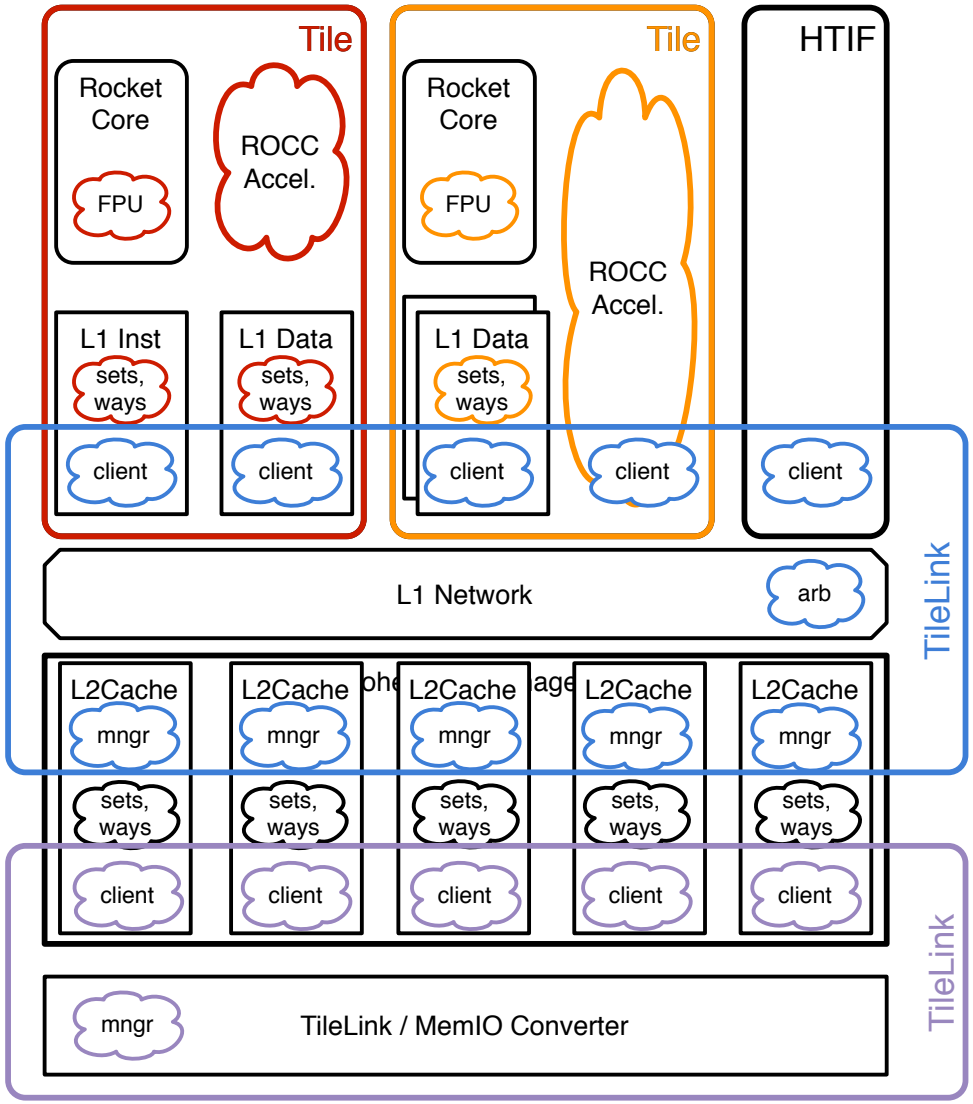
- A RISC-V SoC
  - Look into Chisel parameters
- New Accelerators
  - Drop in at ROCCIO level
- Own RISC-V Core
  - Drop in at TileLinkIO level or MemIO level
- Own Device
  - Drop in at TileLinkIO or UncachedTileLinkIO

New Features: L2$ with Directory Bits



- Shared L2$ with multiple banks
- Each L2$ will act as a coherence manager with directory bits (snoop filter)
- These caches can be composed to build outer-level caches such as an L3$

New Features: ROCC interfaces with L2$



- ROCC talks directly to the L2$ to address more data

New Features on the Deck

- Dual-issue Rocket Core
- Hwacha Vector Unit (checkout hwacha.org)
- Dump MemIO and use AXI