Z-scale:
Tiny 32-bit RISC-V Systems
With Updates to the Rocket Chip Generator

Yunsup Lee, Albert Ou, Albert Magyar

2nd RISC-V Workshop

yunsup@eecs.berkeley.edu

6/30/2015

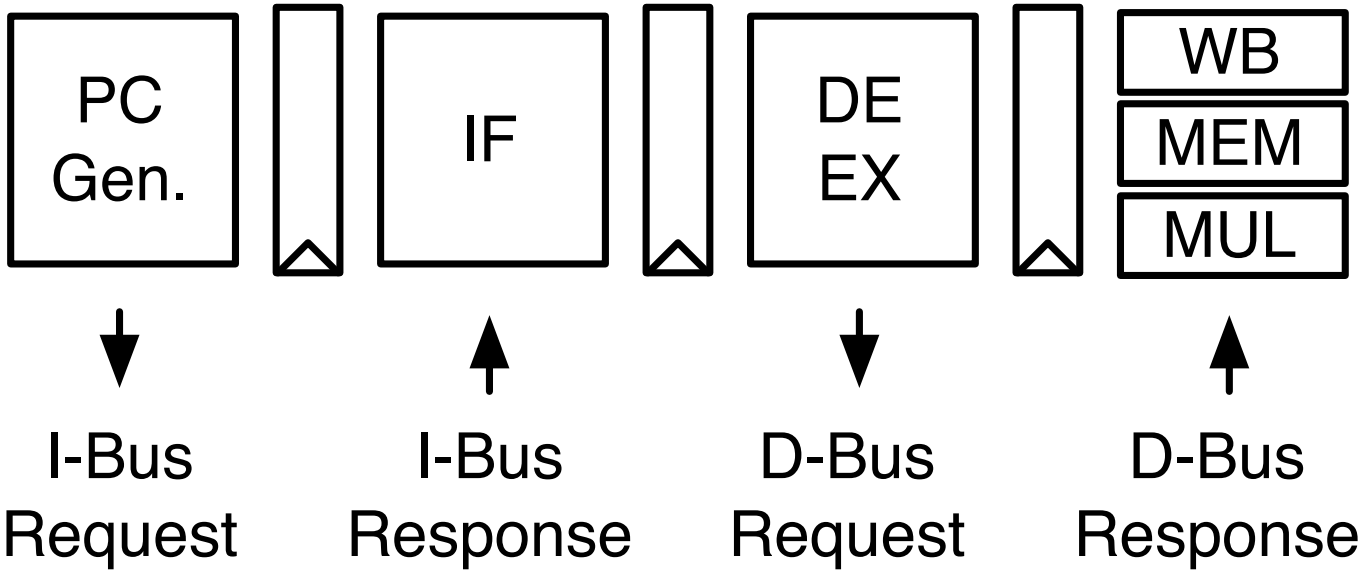
What is Z-scale? Why?

- Z-scale is a tiny 32-bit RISC-V core generator suited for microcontrollers and embedded systems
- Over the past months, external users have expressed great interest in small RISC-V cores
  - So we have listened to your feedback!
- Z-scale is designed to talk to AHB-Lite buses
  - plug-compatible with ARM Cortex-M series
- Z-scale generator also generates the interconnect between core and devices
  - Includes buses, slave muxes, and crossbars

Berkeley's RISC-V Core Generators

- **Z-scale**: Family of Tiny Cores
  - Similar in spirit to *ARM Cortex M0/M0+/M3/M4*
  - Integrates with AHB-Lite interconnect
- **Rocket**: Family of In-order Cores
  - Currently 64-bit single-issue only
  - Plans to work on dual-issue, 32-bit options
  - Similar in spirit to *ARM Cortex A5/A7/A53*
  - Will integrate with AXI4 interconnect
- **BOOM**: Family of Out-of-Order Cores
  - Supports 64-bit single-, dual-, quad-issue
  - Similar in spirit to *ARM Cortex A9/A15/A57*
  - Will integrate with AXI4 interconnect
  - BOOM talk right after this one

Z-scale Pipeline



- 32-bit 3-stage single-issue in-order pipe
- Executes RV32IM ISA, has M/U privilege modes
- I-bus and D-bus are AHB-Lite and 32-bits wide
- Interrupts are supported
- Will publish a “microarchitecture specification”

ARM Cortex-M0 vs. Z-scale

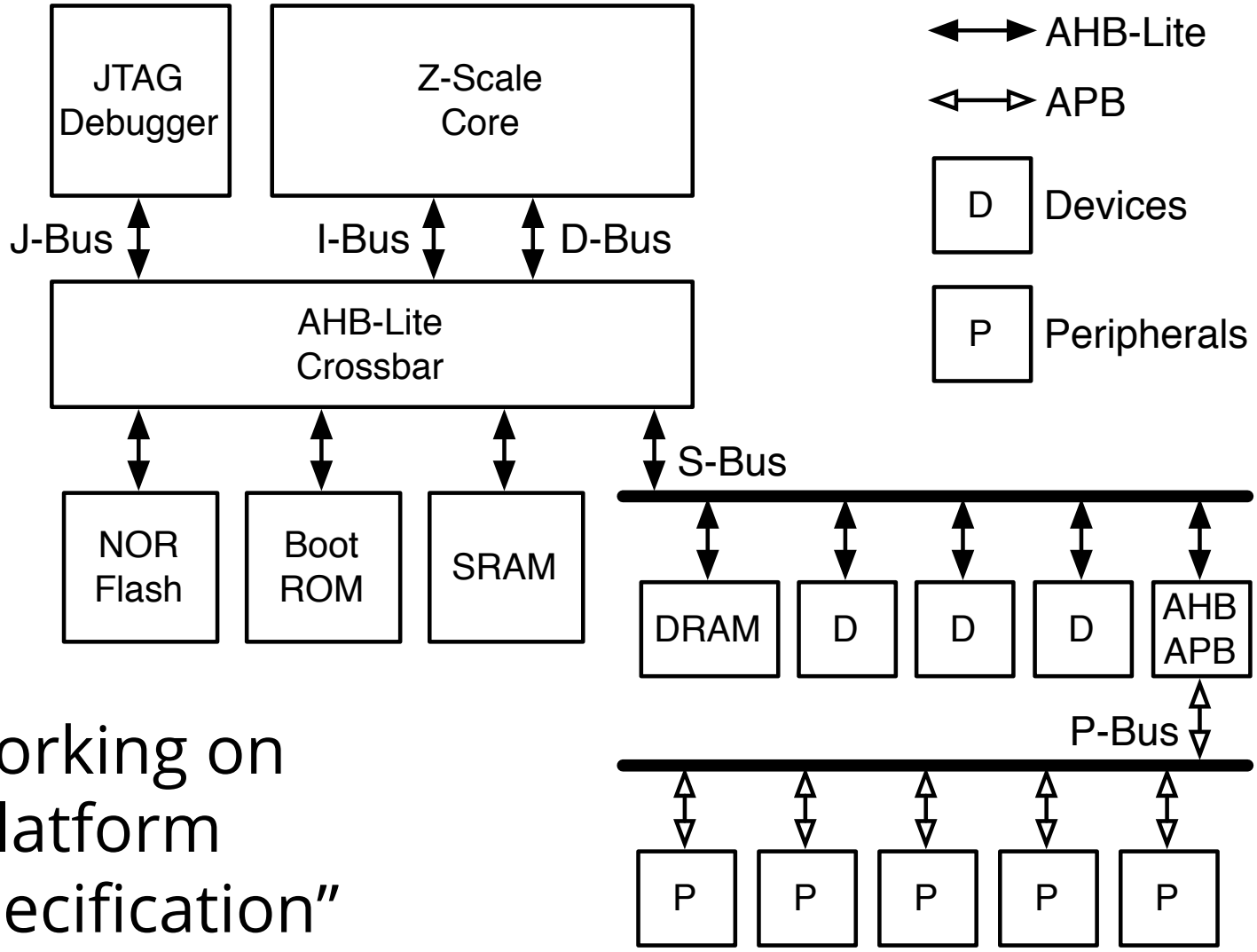
Category	ARM Cortex-M0	RISC-V Zscale
ISA	32-bit ARM v6	32-bit RISC-V (RV32IM)
Architecture	Single-Issue In-Order 3-stage	Single-Issue In-Order 3-stage
Performance	0.87 DMIPS/MHz	1.35 DMIPS/MHz
Process	TSMC 40LP	TSMC 40GPLUS
Area w/o Caches	0.0070 mm^2	0.0098 mm^2
Area Efficiency	124 DMIPS/MHz/mm^2	138 DMIPS/MHz/mm^2
Frequency	≤50 MHz	~500 MHz
Voltage (RTV)	1.1 V	0.99 V
Dynamic Power	5.1 µW/MHz	1.8 µW/MHz

- Note: numbers are very likely to change in the future as we tune the design and add things to the core.

RV32E

- New base integer instruction set
  - Reduced version of RV32I designed for embedded systems
- Cut number of integer registers to 16
- Remove counters that are mandatory in RV32I
  - Counter instructions (rdcycle[h], rdtime[h], rdinstret[h]) are not mandatory

Building a Z-scale System



- Working on "platform specification"

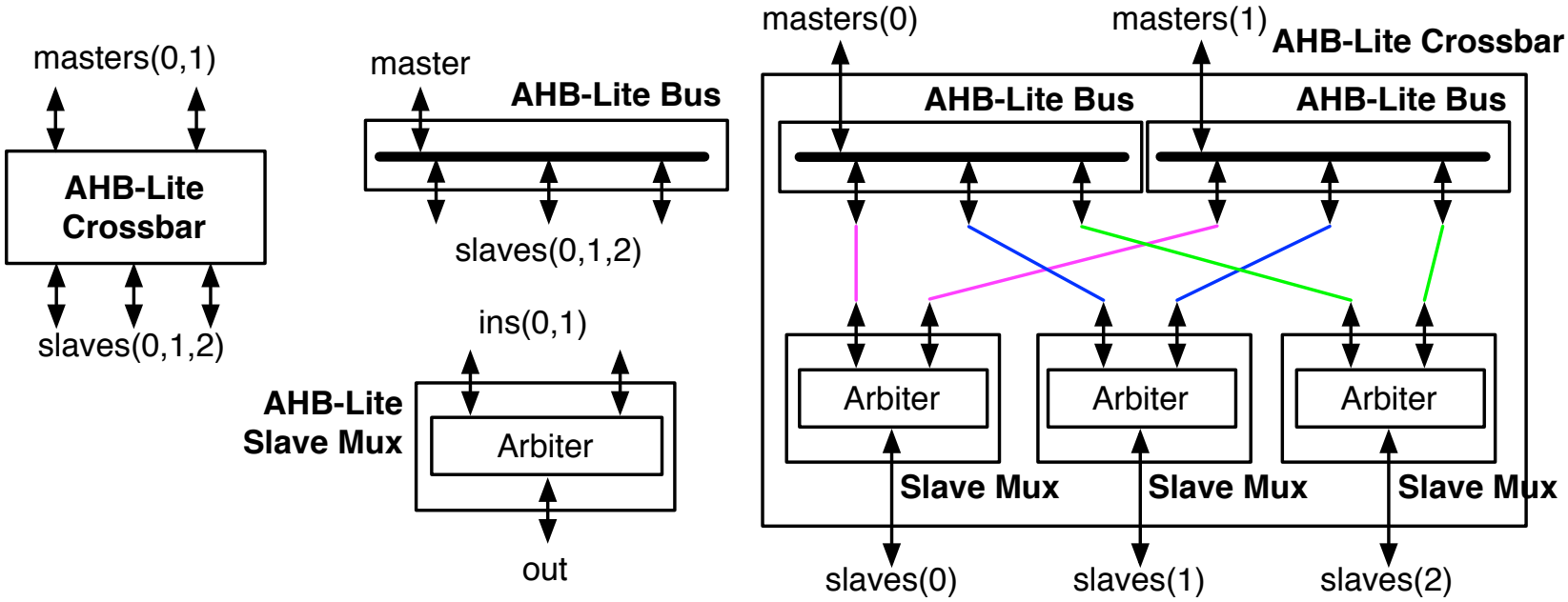
Z-scale Generator is Written in Chisel

- Chisel is a new HDL embedded in Scala
  - Rely on good software engineering practices such as abstraction, reuse, object oriented programming, functional programming
  - Build hardware like software
- 604 Unique LOC written in Chisel
  - Control: 274 lines
  - Datapath: 267 lines (99 lines could be generalized)
  - Top-level: 63 lines
- 983 LOC in Chisel borrowed from Rocket
- Reuse and parameterization in Chisel and Rocket chip actually works!

Functional Programming 101

- (1, 2, 3) **map** { n => n+1 }
  - (2, 3, 4)
- (1, 2, 3) **map** { _+1 }
- (1, 2, 3) **zip** (a, b, c)
  - ((1, a), (2, b), (3, c))
- (1, a)._1
  - 1
- ((1, a), (2, b), (3, c)) **map** { _._1}
- ((1, a), (2, b), (3, c)) **map** { n => n._1}
  - (1, 2, 3)
- (0 until 3) **foreach** { println(_) }
  - 0/1/2

Functional Programming Example Used in AHB-Lite Crossbar

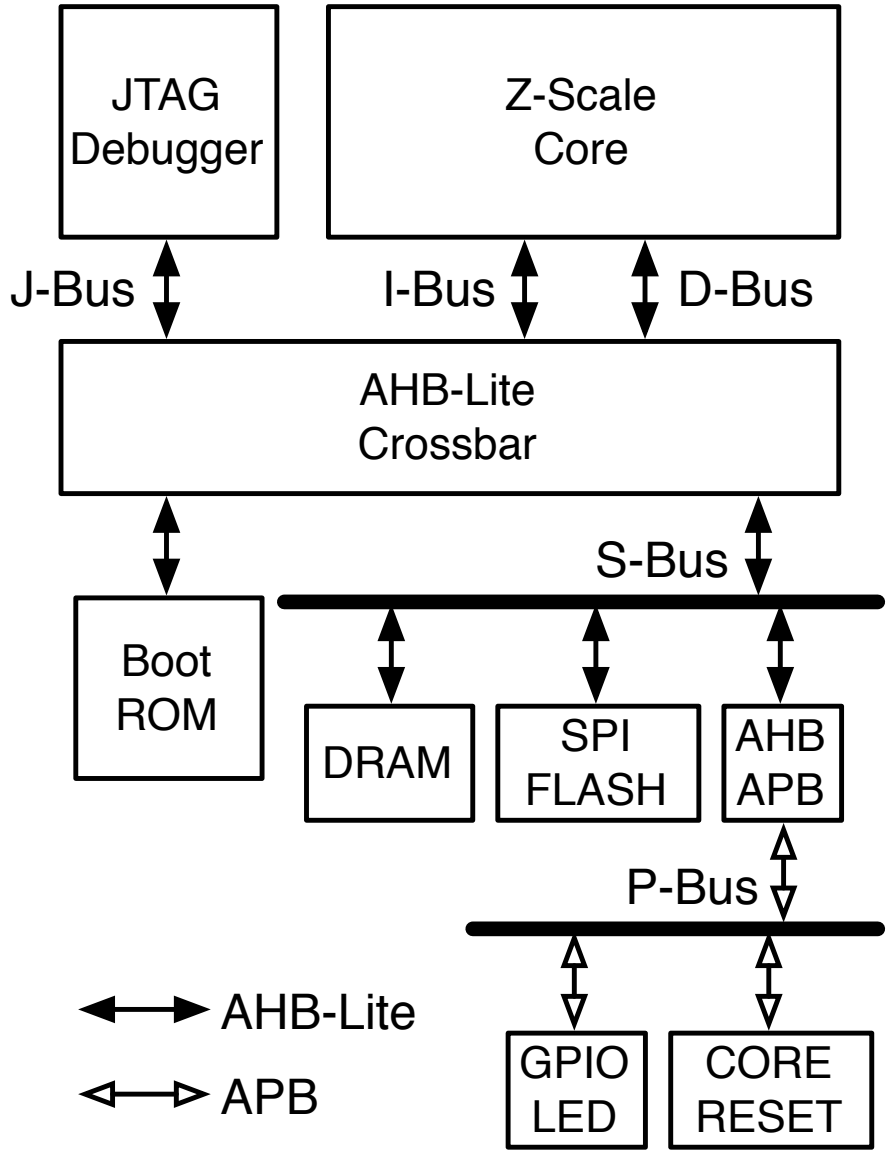


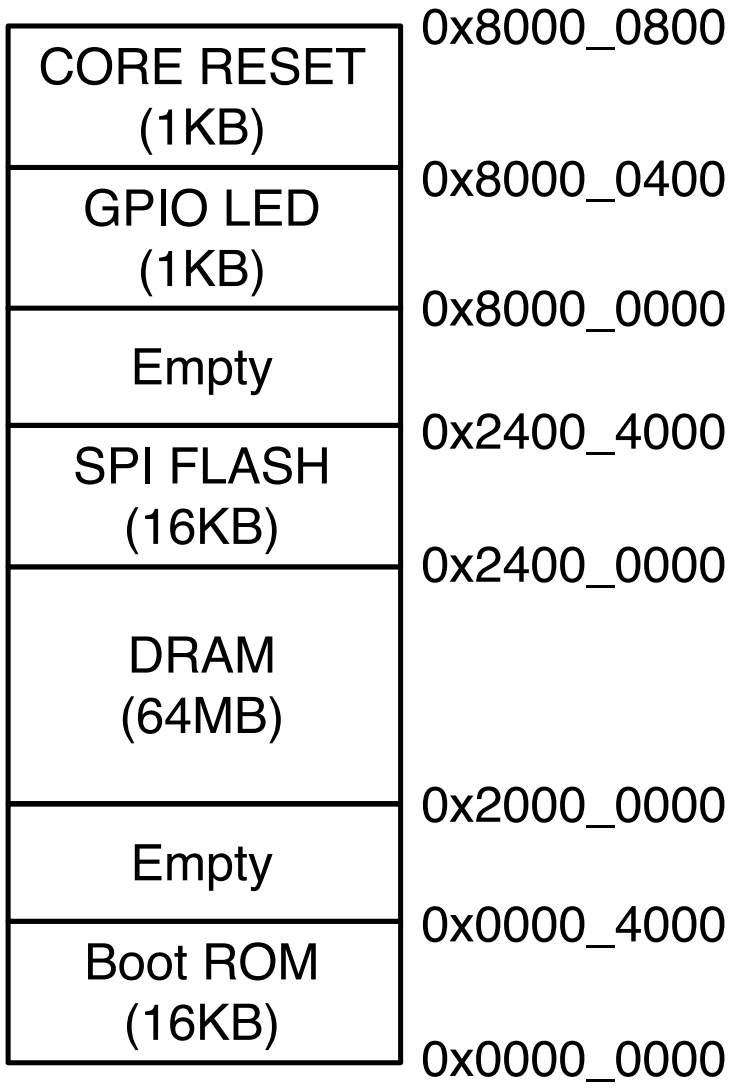

```
class AHBXbar(n: Int, amap: Seq[UInt=>Bool]) extends Module
{
  val io = new Bundle {
    val masters = Vec.fill(n){new AHBMasterIO}.flip
    val slaves = Vec.fill(amap.size){new AHBSlaveIO}.flip
  }

  val buses = io.masters map { m => Module(new AHBBus(amap)).io }
  val muxes = io.slaves map { s => Module(new AHBSlaveMux(n)).io }

  (buses.map(_.master) zip io.masters) foreach { case (b, m) => b <> m }
  (0 until n) foreach { m => (0 until amap.size) foreach { s =>
     buses(m).slaves(s) <> muxes(s).ins(m) } }
  (io.slaves zip muxes.map(_.out)) foreach { case (s, x) => s <> x }
}
```

Z-scale in Verilog

- Talked to many external users, and perhaps the #1 reason why they can't use our stuff is because it's written in Chisel
  - So we have listened to your feedback!
- We have implemented the same Z-scale core in Verilog
- 1215 LOC
- No more excuses for adoption!
  - If there still is any reason why you can't use RISC-V, please do let us know

Z-scale FPGA DEMO System





Z-scale FPGA DEMO System Mapped to Xilinx Spartan6 LX9

Resource	Used	Percentage
Registers	2,329	20%
LUTs	4,328	75%
RAM16	8	25%
RAM8	0	0%

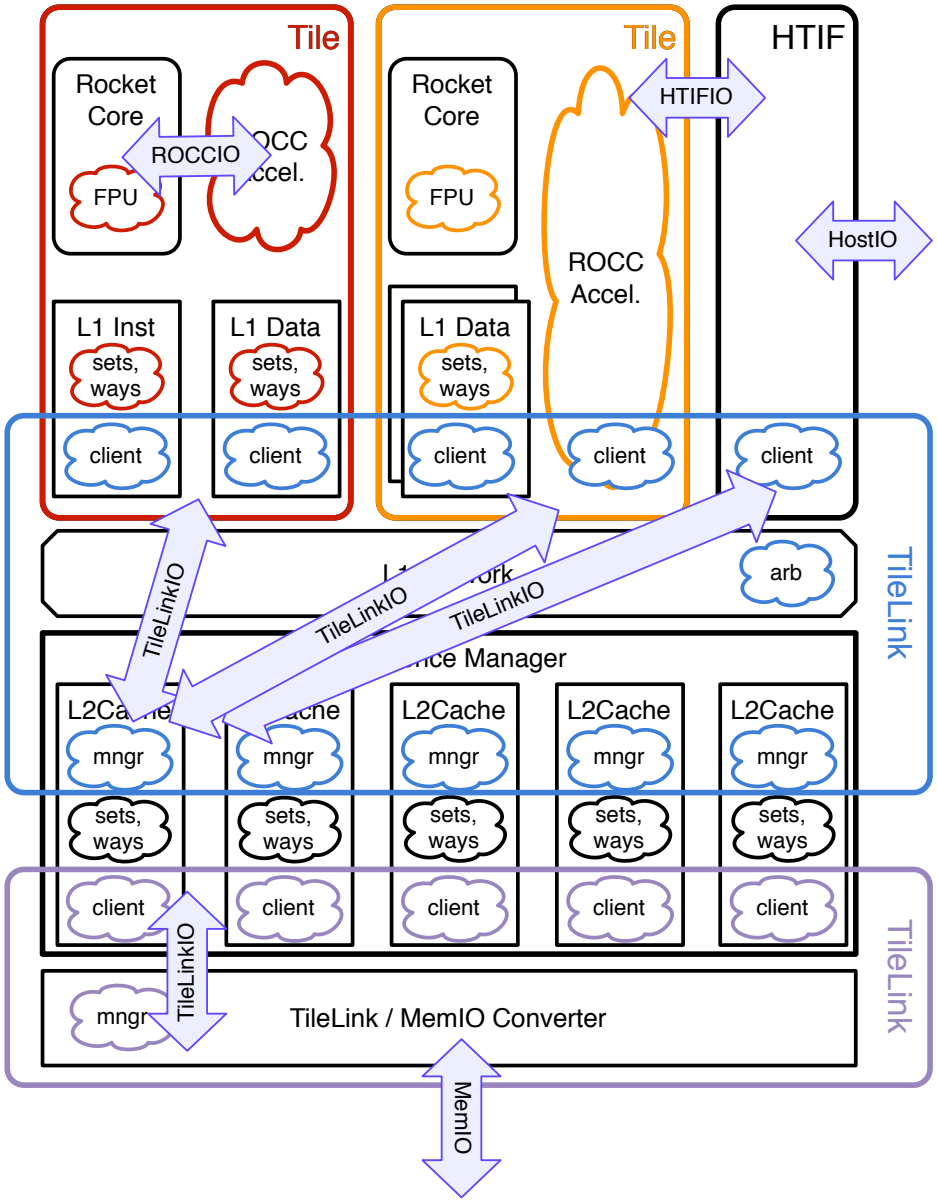
Test program is stored in bootrom. It is a memory test program, which writes 32-bit words generated from an LFSR to 64MB of DRAM, and checks it by reading 64MB of data, and toggles LED if it succeeds.

- Avnet LX9 Microboard
  - $89
  - Xilinx Spartan6 LX9
  - 64MB LPDDR RAM
  - 16MB SPI FLASH
  - 10/100 Ethernet
  - USB-to-UART
  - USB-to-JTAG
  - 2x Pmod headers
  - 4x LEDs
  - 4x DIP switches
  - RESET/PROG buttons
- 4 boards for raffle!

Z-scale Use Cases

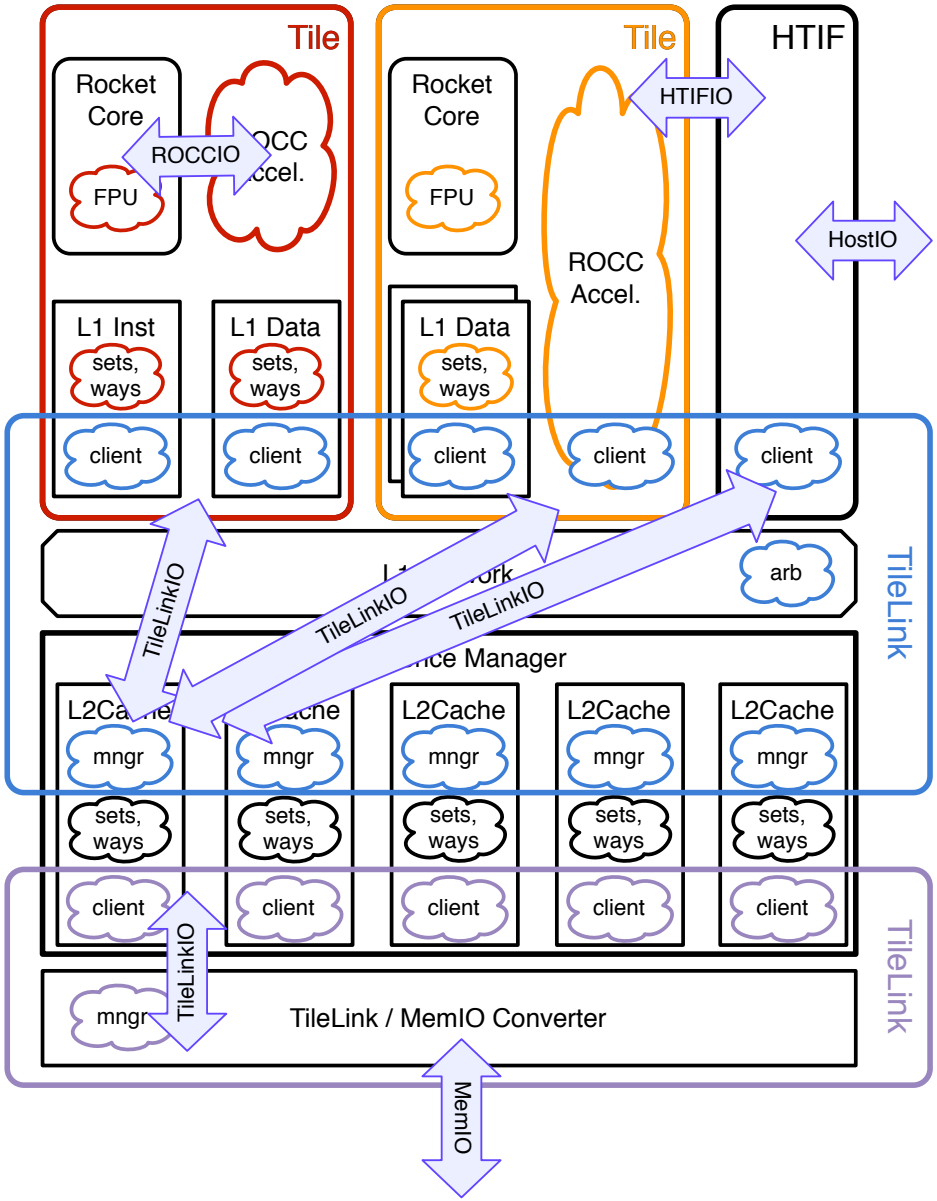
- Microcontrollers
  - Implement your simple control loops
  - If code density matters
- Embedded Systems
  - Build your system around Z-scale
- Validation of Tiny 32-bit RISC-V Systems
  - You don't need to use our code, just consider Z-scale as an existence proof and implement your own RV32I core
- Both Chisel and Verilog versions of Z-scale is open-sourced under the BSD license
  - https://github.com/ucb-bar/zscale
  - https://github.com/ucb-bar/fpga-spartan6

What is the Rocket Chip Generator?



- Parameterized SoC generator written in Chisel
- Generates n Tiles
  - (Rocket) Core
  - RoCC Accelerator
  - L1 I$
  - L1 D$
- Generates Uncore
  - L1 Crossbar
  - Coherence Manager
  - Shared L2$ with directory bits
  - Exports a simple memory interface

Rocket Chip Generator Updates Since the 1st RISC-V Workshop

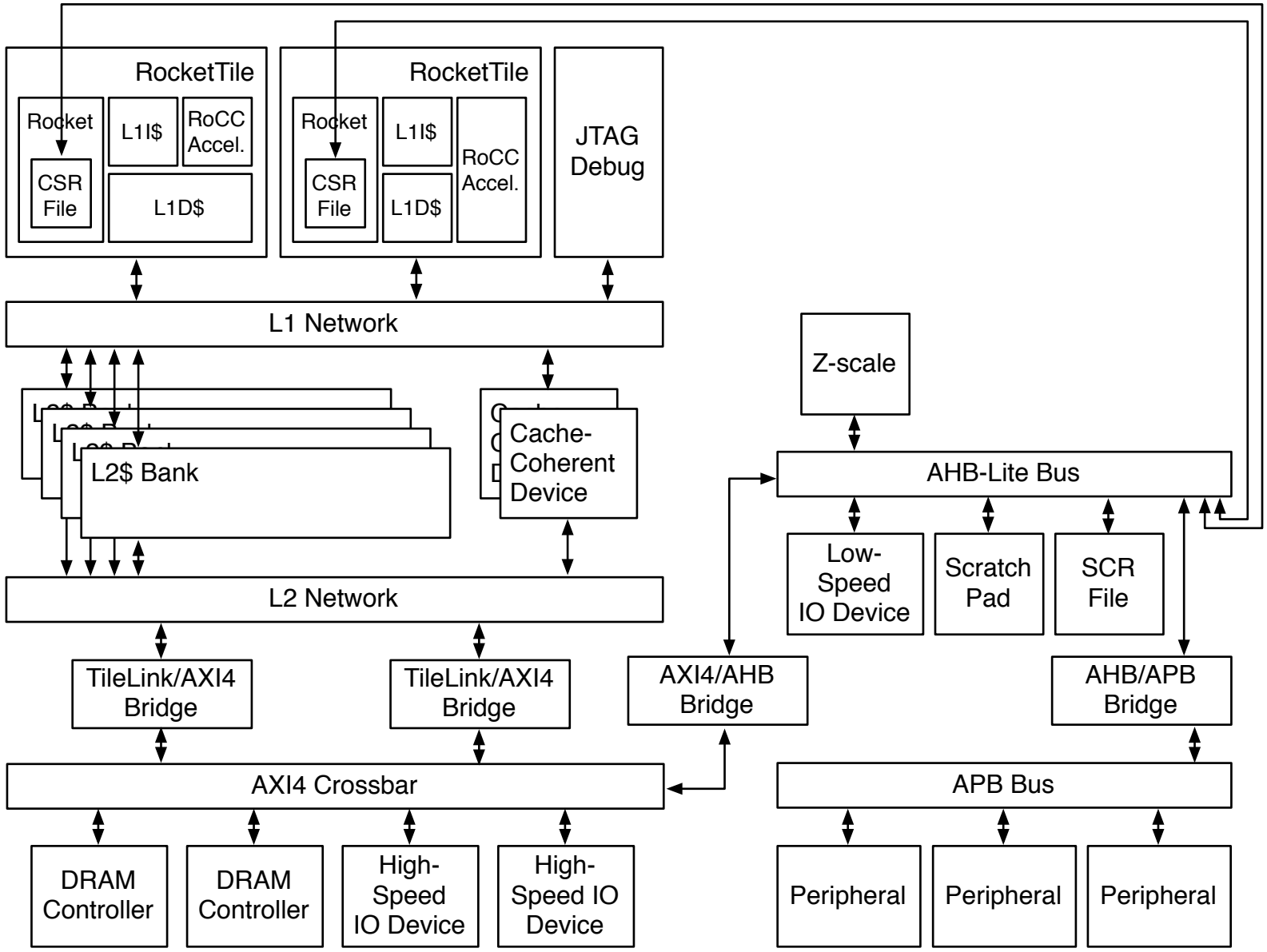


- Implemented L2$ with directory bits
- RoCC coprocessor has a memory port directly into the L2$
- Main development will happen on the rocket-chip repository
- Moving towards standardized memory interfaces

Important Memory Interfaces

- TileLink
  - Our cache-coherent interconnect
  - For more details, watch my talk from last workshop
- NASTI (pronounced nasty)
  - Not A STandard Interface
  - Our implementation of the AXI4 standard
- HASTI (pronounced hasty)
  - Highly Advanced System Transport Interface
  - Our implementation of the AHB-Lite standard
- POCI (pronounced pokey)
  - Peripheral Oriented Connection Interface
  - Our implementation of the APB standard

Rocket Chip Generator Grand Plan with Z-scale



Conclusion, Future Work, and Raffle

- Z-scale is a RISC-V tiny core generator suited for microcontrollers and embedded systems
- Z-scale
  - Microarchitecture document will be released first
  - Improve performance
  - Implement “C” extension as an option
  - Add MMU option to boot Linux
  - More devices on the LX9 board to come
- Rocket Chip Generator
  - JTAG debug interface (get rid of HTIF)
  - Move to standardized interfaces (NASTI/HASTI/POCI)
  - Add Z-scale option
- Raffle time!