Generating Rocket/BOOM SoCs with Rocket Chip

Howard Mao, Jerry Zhao

UC Berkeley Architecture Research

Hot Chips 2019

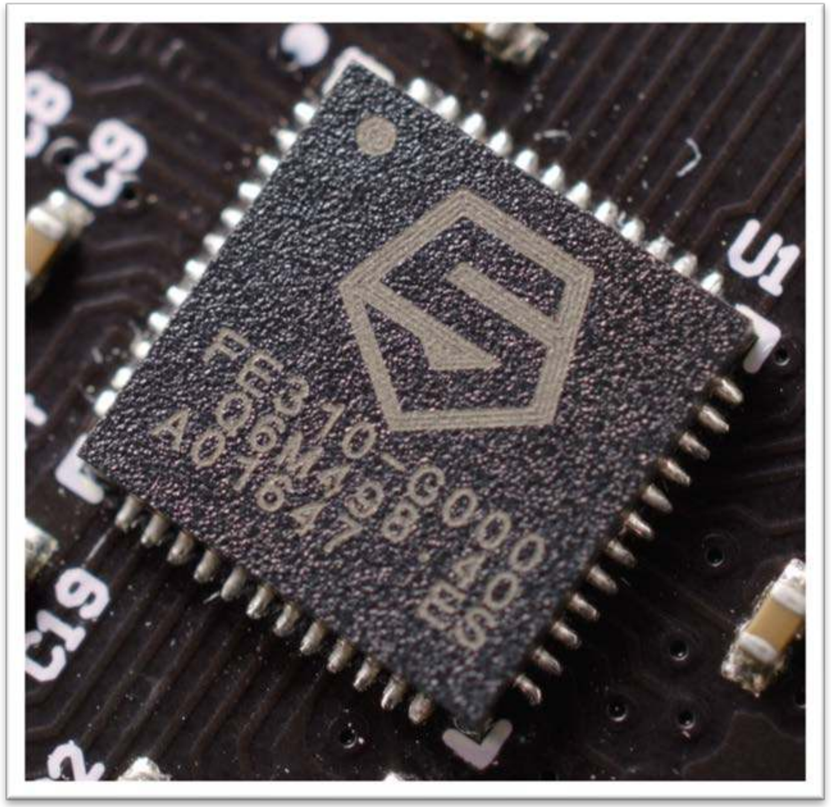
Berkeley Architecture Research

What is Rocket Chip?

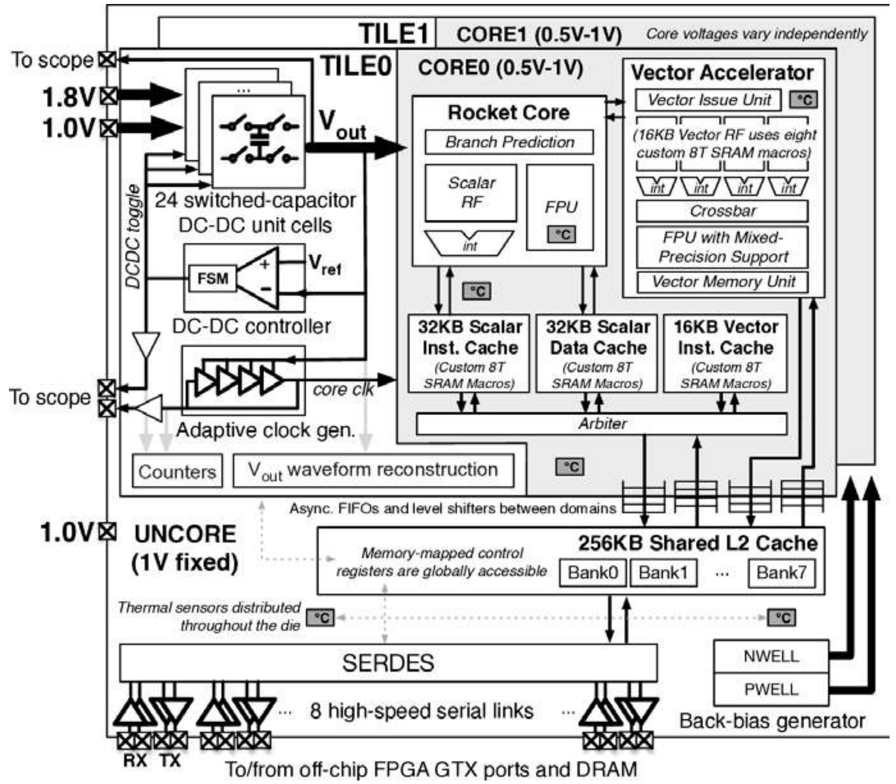
- A highly parameterizable SoC generator
  - Replace default Rocket core w/ your own core
  - Add your own coprocessor
  - Add your own SoC IP to uncore
- A library of reusable SoC components
  - Memory protocol converters
  - Arbiters and Crossbar generators
  - Clock-crossings and asynchronous queues
- The largest open-source Chisel codebase
  - Scala allows advanced generator features
- Developed at Berkeley, now maintained by many
  - SiFive, ChipsAlliance, Berkeley

Generating Varied SoCs

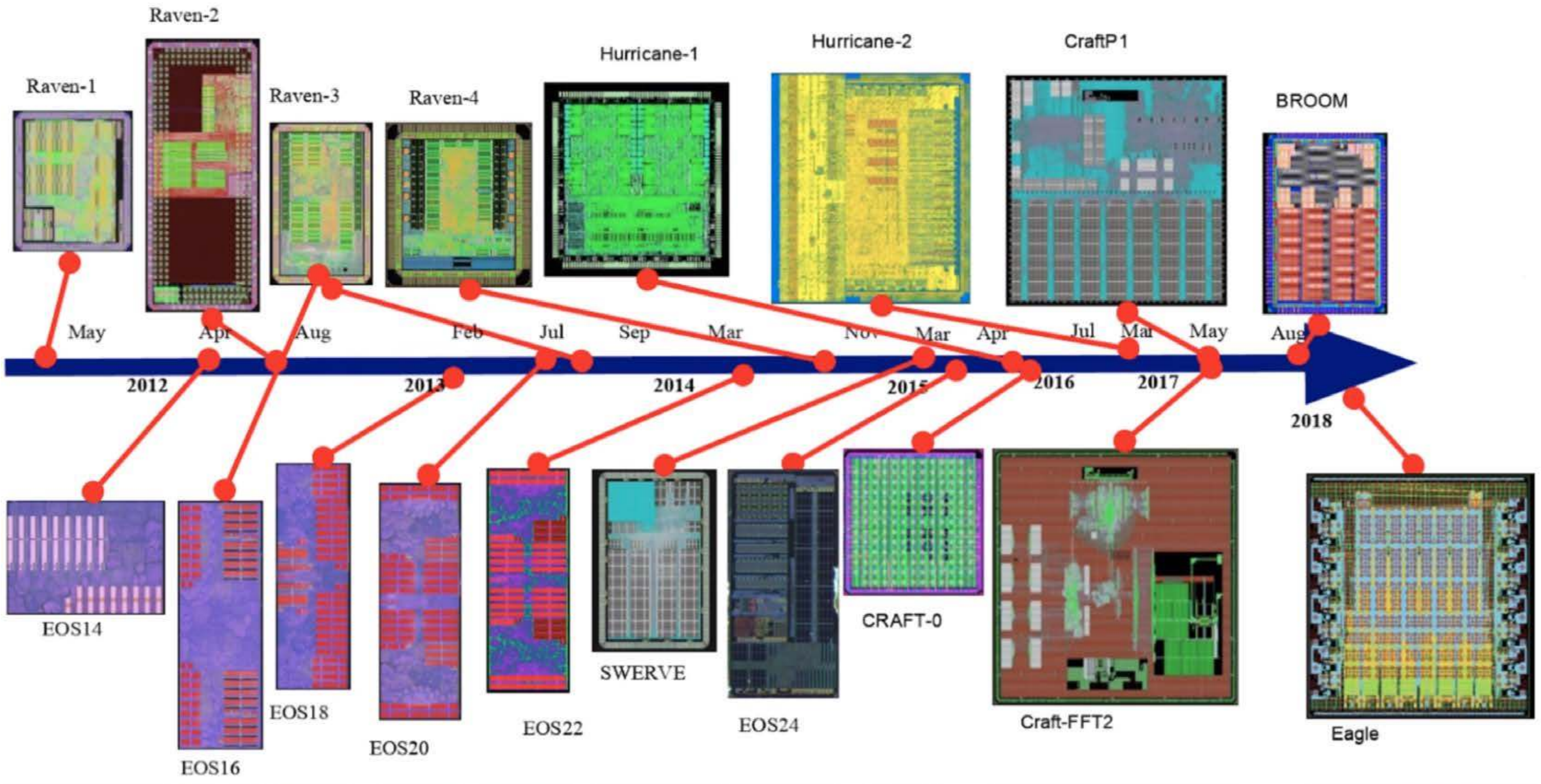
In industry: **SiFive Freedom E310**



In academia: **UCB Hurricane-1**


TILE1
CORE1 (0.5V-1V)
Core voltages vary independently
To scope
TILE0
CORE0 (0.5V-1V)
Vector Accelerator
Vector Issue Unit
1.8V
1.0V
Vout
Rocket Core
Branch Prediction
(16KB Vector RF uses eight custom 8T SRAM macros)
Scalar RF
FPU
int
Crossbar
FPU with Mixed-Precision Support
Vector Memory Unit
24 switched-capacitor DC-DC unit cells
DCDC toggle
FSM
Vref
DC-DC controller
32KB Scalar Inst. Cache (Custom 8T SRAM Macros)
32KB Scalar Data Cache (Custom 8T SRAM Macros)
16KB Vector Inst. Cache (Custom 8T SRAM Macros)
core clk
Arbiter
Adaptive clock gen.
Counters
Vout waveform reconstruction
°C
Async. FIFOs and level shifters between domains
1.0V
UNCORE (1V fixed)
Memory-mapped control registers are globally accessible
256KB Shared L2 Cache
Bank0
Bank1
...
Bank7
Thermal sensors distributed throughout the die
SERDES
NWELL
PWELL
8 high-speed serial links
Back-bias generator
RX
TX
To/from off-chip FPGA GTX ports and DRAM

Used in Many Tapeouts

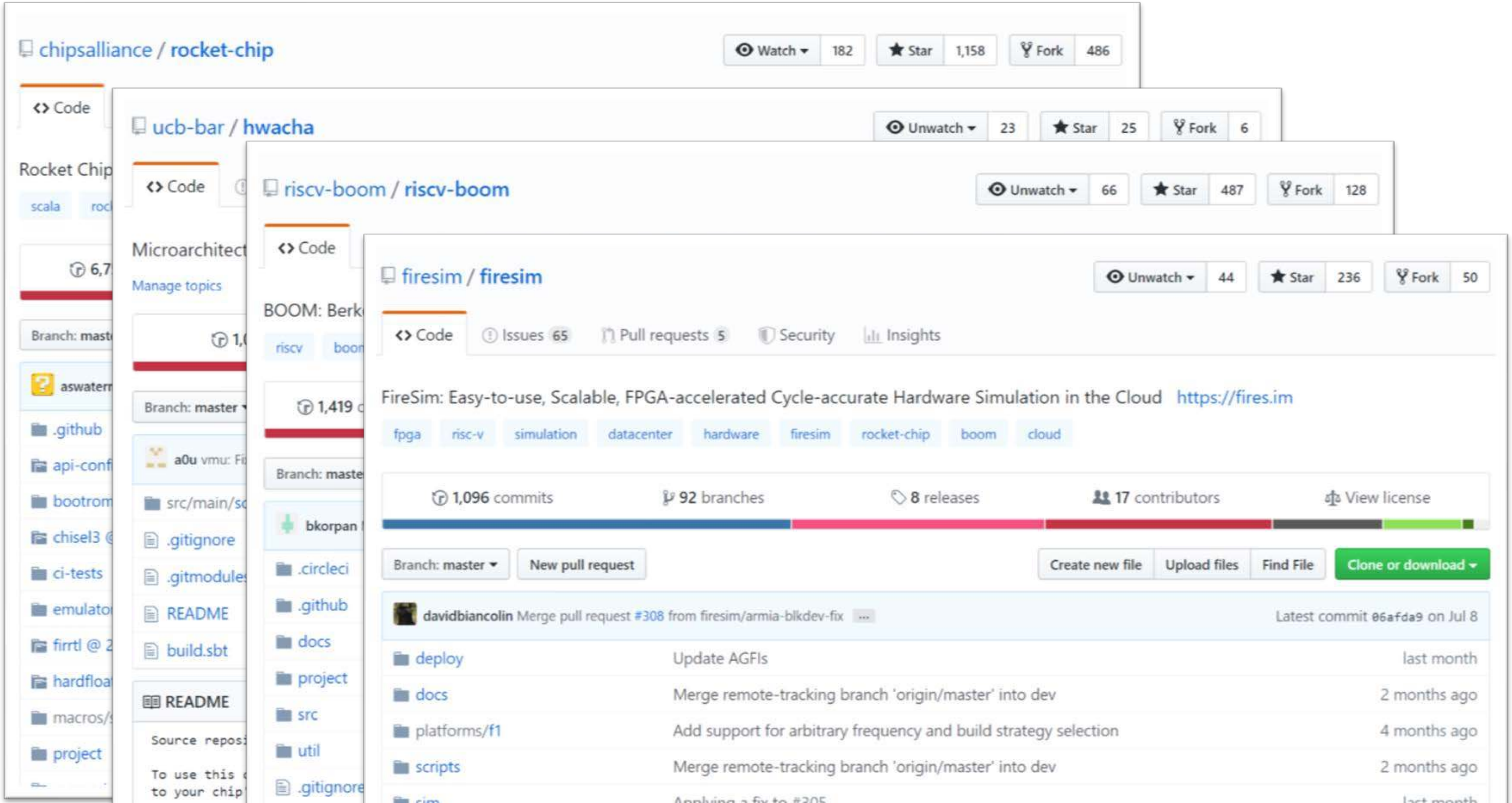


Built with Chisel

- Chisel is a hardware construction DSL built on top of Scala
- Allows description of RTL in a more programmable way
  - Utilize OOP/Functional programming paradigms
  - NOT Scala-to-Gates / HLS in Scala
- Use Scala features to build complex parameterized generators

```
class TreeAdderPipeline(n: Int) extends Module {
  val io = IO(new Bundle {
    val in = Input(Vec(n, UInt(32.W)))
    val out = Output(UInt(32.W))
  })
  val nStages = log2Ceil(n)
  val stages = Seq.tabulate(nStages) { i => Reg(Vec(nStages-i-1, UInt(32.W))) }
  for (i <- 1 until nStages) {
    for (j <- 0 until stages(i).size) {
      stages(i)(j) := stages(i-1)(2*j) + stages(i-1)(2*j+1)
    }
  }
  io.out := stages(nStages-1)(0)
}
```

Fully Open-Source



chipsalliance / rocket-chip

Watch 182 | Star 1,158 | Fork 486

Code

Rocket Chip

scala roc

6,7

Branch: mast

aswaterr

.github
api-conf
bootrom
chisel3 @
ci-tests
emulato
firrtl @ 2
hardfloa
macros/
project

ucb-bar / hwacha

Unwatch 23 | Star 25 | Fork 6

Code

Microarchitect

Manage topics

1,

Branch: master

a0u vmu: Fi

src/main/sc
.gitignore
.gitmodules
README
build.sbt

README

```
Source reposi

To use this
to your chip
```

riscv-boom / riscv-boom

Unwatch 66 | Star 487 | Fork 128

Code

BOOM: Berk

riscv boon

1,419 c

Branch: maste

bkorpan

.circleci
.github
docs
project
src
util
.gitignore

firesim / firesim

Unwatch 44 | Star 236 | Fork 50

Code | Issues 65 | Pull requests 5 | Security | Insights

FireSim: Easy-to-use, Scalable, FPGA-accelerated Cycle-accurate Hardware Simulation in the Cloud https://fires.im

fpga risc-v simulation datacenter hardware firesim rocket-chip boom cloud

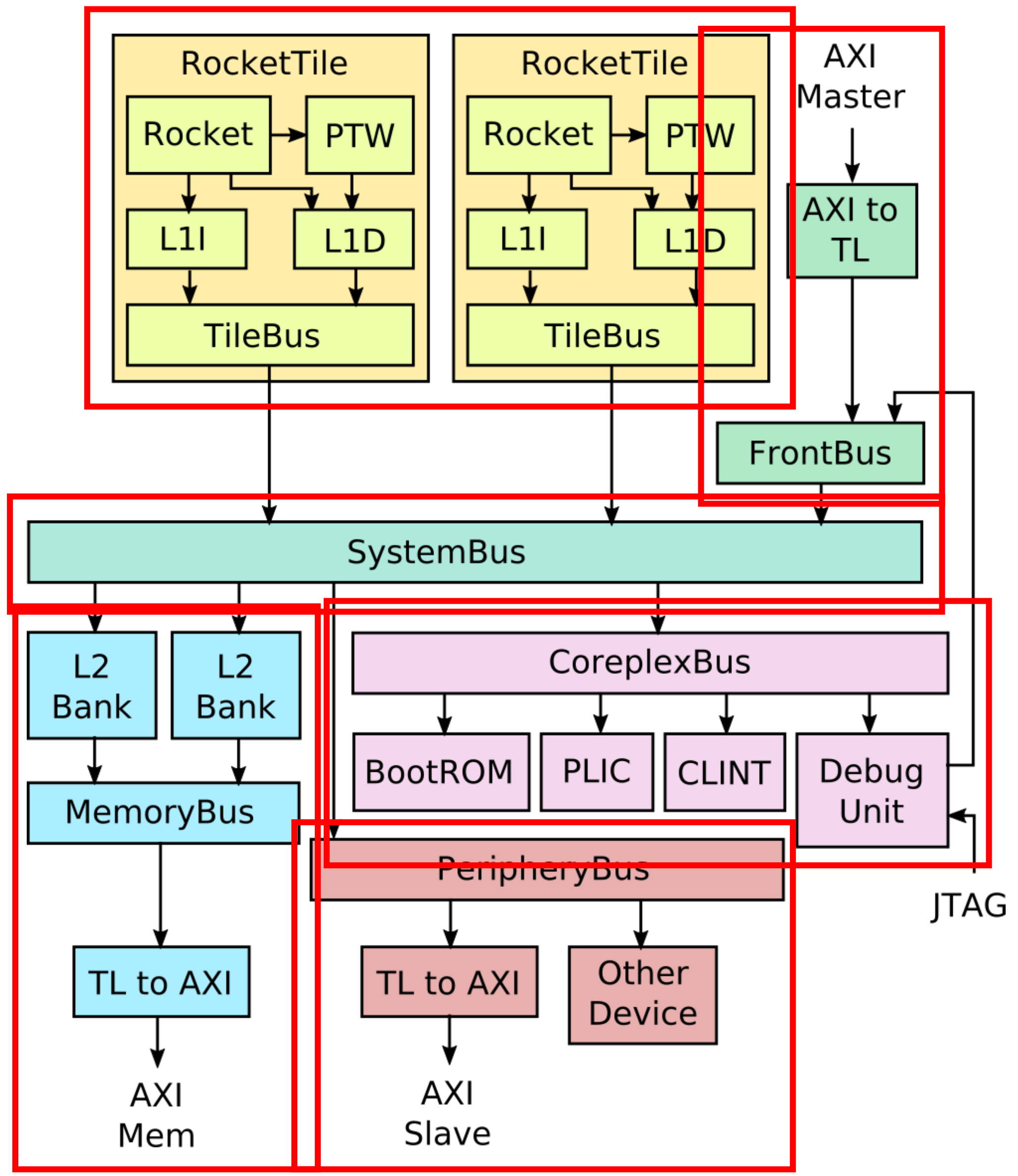
1,096 commits | 92 branches | 8 releases | 17 contributors | View license

Branch: master | New pull request | Create new file | Upload files | Find File | Clone or download

davidbiancolin Merge pull request #308 from firesim/armia-blkdev-fix … Latest commit 06afda9 on Jul 8

deploy	Update AGFIs	last month
docs	Merge remote-tracking branch 'origin/master' into dev	2 months ago
platforms/f1	Add support for arbitrary frequency and build strategy selection	4 months ago
scripts	Merge remote-tracking branch 'origin/master' into dev	2 months ago
sim	Applying a fix to #305	last month

Structure of a Rocket Chip SoC



Tiles: unit of replication for a core

- CPU
- L1 Caches
- Page-table walker

L2 banks:

- Receive memory requests

FrontBus:

- Connects to DMA devices

CoreplexBus:

- Connects to core-complex devices

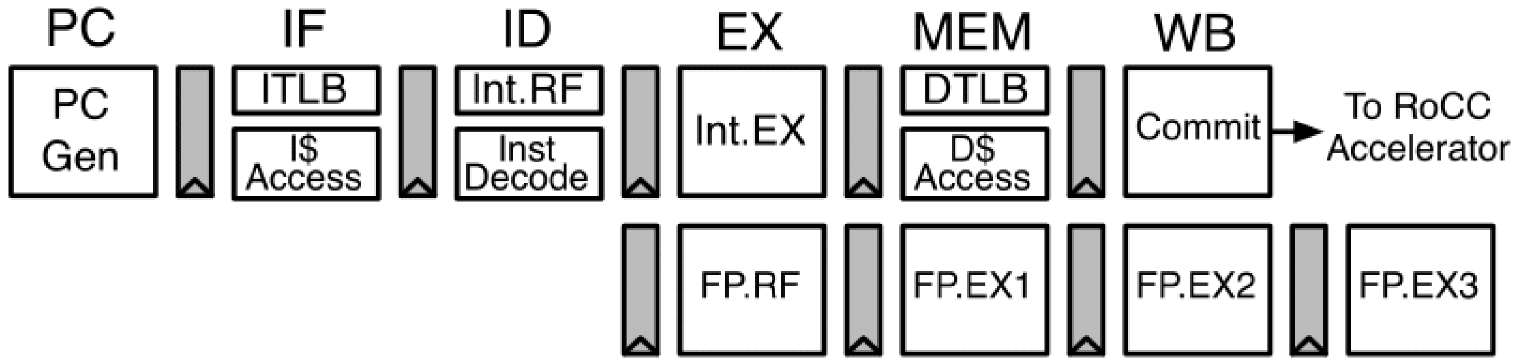
PeripheryBus:

- Connects to other devices

SystemBus:

- Ties everything together

The Rocket In-Order Core



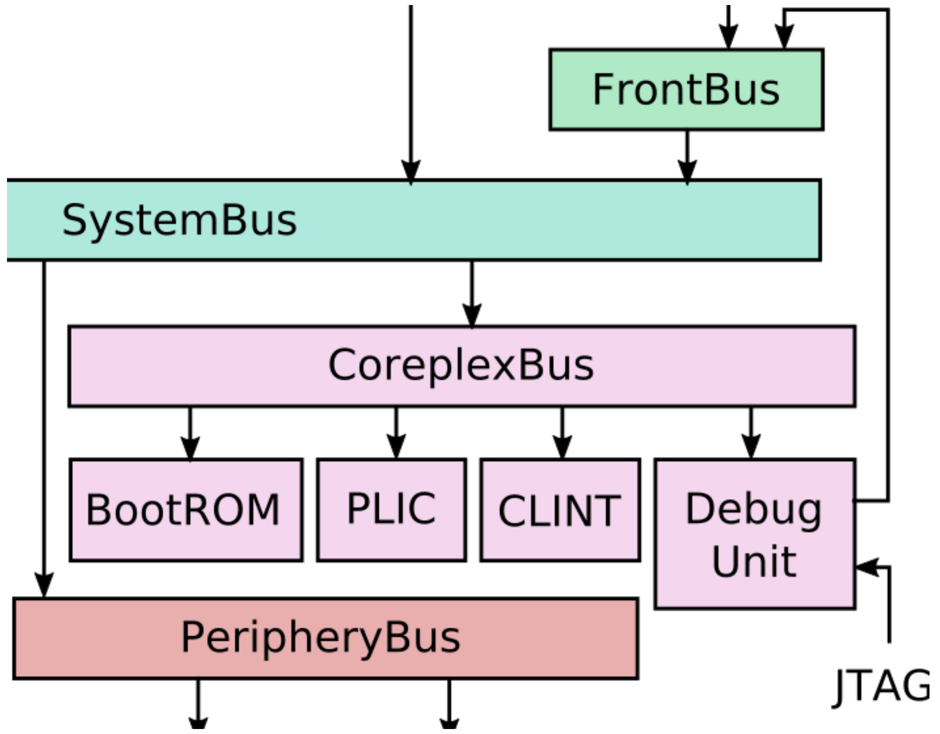
- First open-source RISC-V CPU
- In-order, single-issue RV64GC core
  - Floating-point via Berkeley hardfloat library
  - RISC-V Compressed
  - Physical Memory Protection (PMP) standard
  - Supervisor ISA and Virtual Memory
- Boots Linux
- Supports Rocket Chip Coprocessor (RoCC) interface
- L1 I$ and D$
  - Data cache can be configured as data scratchpad

TileLink Interconnect

- Rocket Chip's memory/cache protocol
- Configurable data width and multi-beat transactions
- Three different protocol levels with increasing complexity
  - TL-UL (Uncached Lightweight)
  - TL-UH (Uncached Heavyweight)
  - TL-C (Cached)
- Rocket Chip provides library of reusable TileLink widgets
  - Conversion to/from AXI4, AHB, APB
  - Conversion among TL-UL, TL-UH, TL-C
  - Width / N beats conversion
  - Crossbar generator

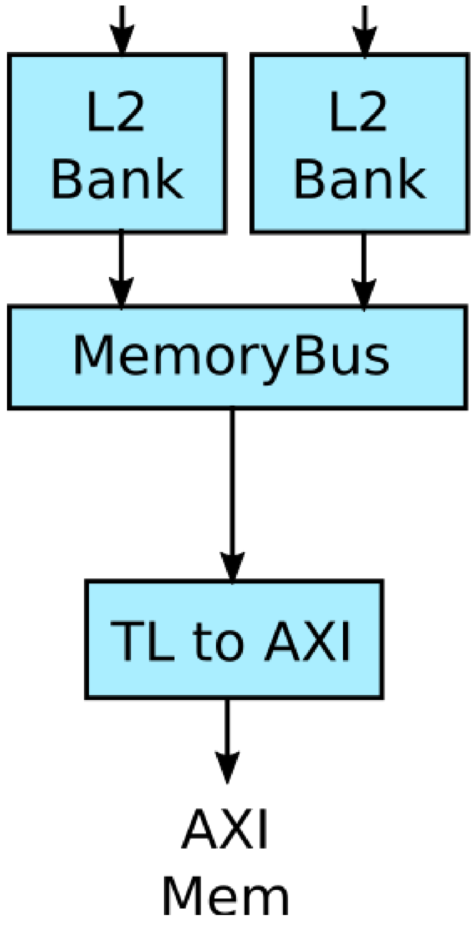
Core Complex Devices

- BootROM
  - Zero-stage bootloader
  - DeviceTree
- PLIC
- CLINT
  - Software interrupts
  - Timer interrupts
- Debug Unit
  - DMI
  - JTAG



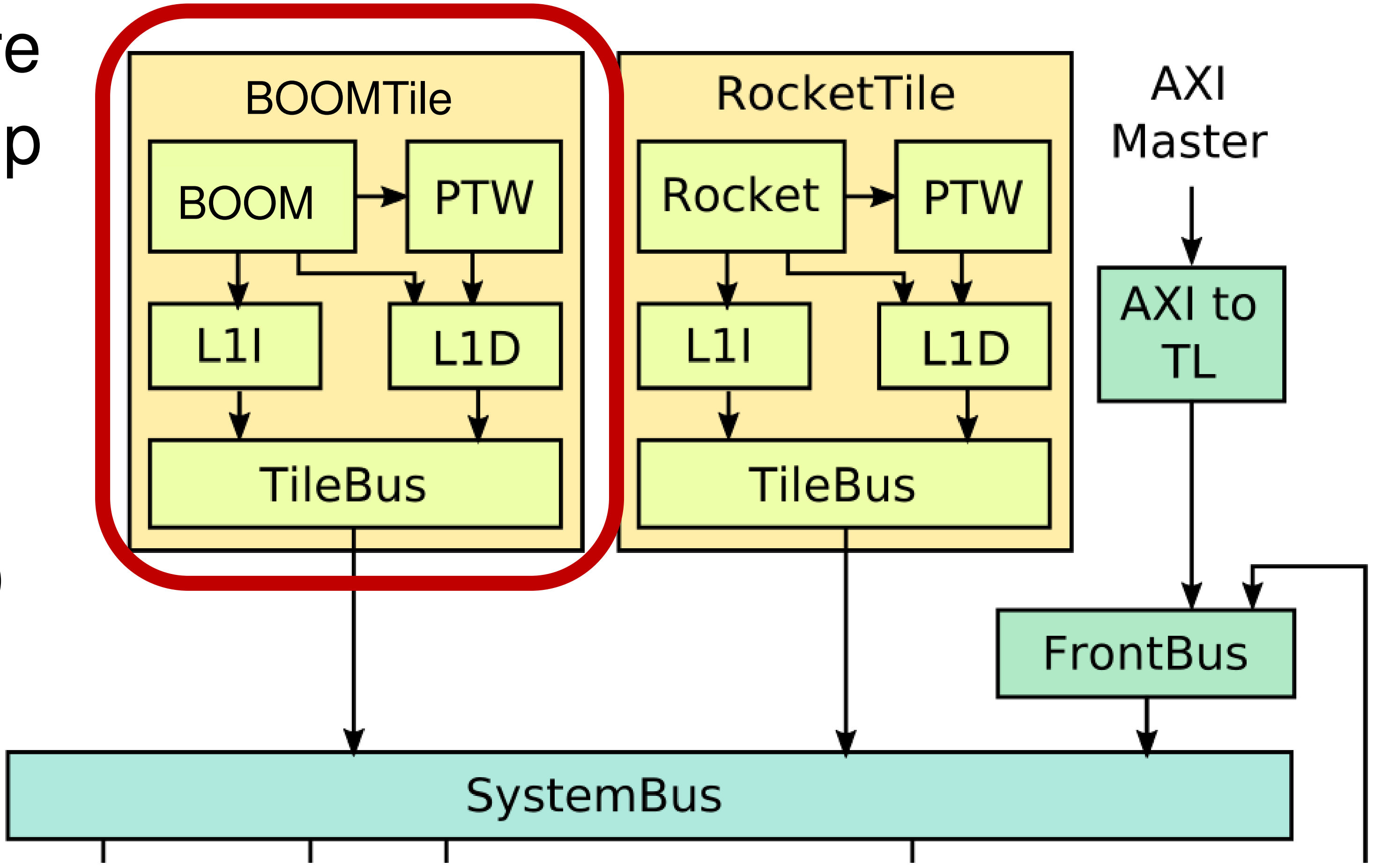
L2 Cache and Memory System

- Multi-bank shared L2
  - SiFive’s open-source IP
  - Fully coherent
  - Configurable size, associativity
  - Supports atomics
- Non-caching L2 Broadcast Hub
  - Coherence w/o caching
  - Bufferless design
- Multi-channel memory system
  - Conversion to AXI4 for compatible DRAM controllers

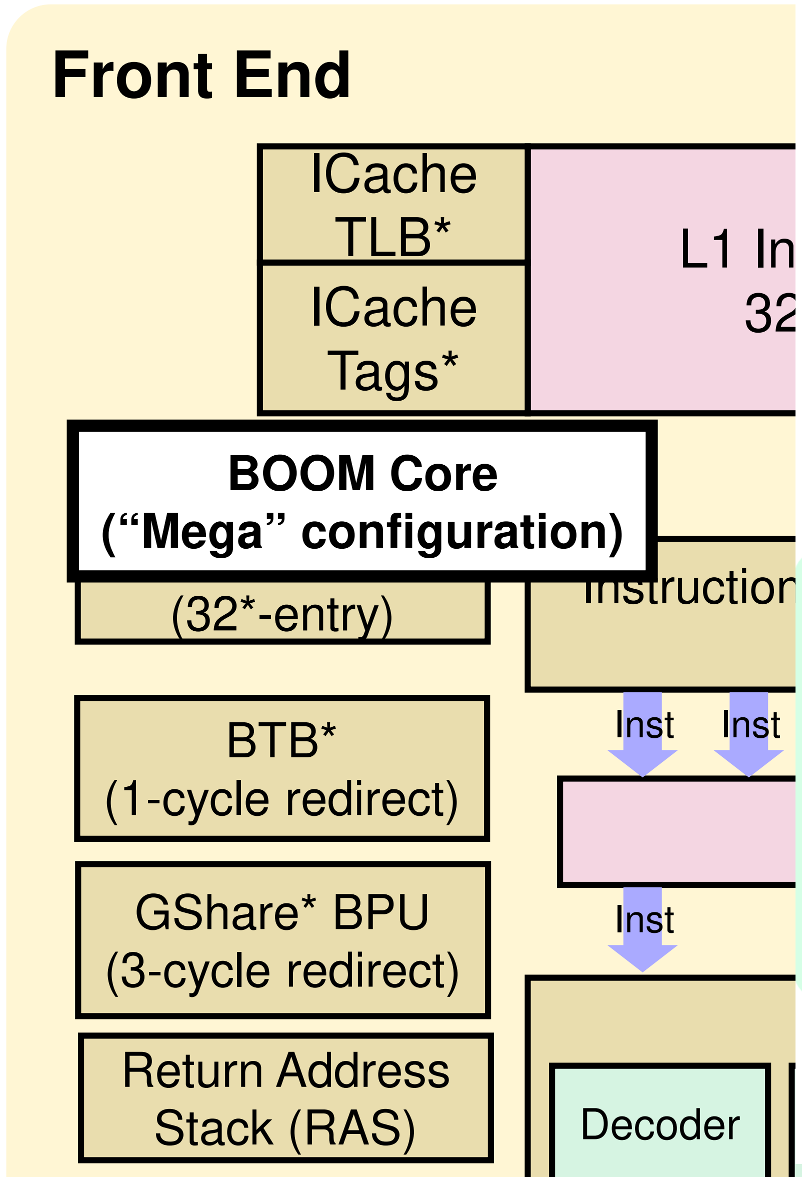


BOOM: The Berkeley Out-of-Order Machine

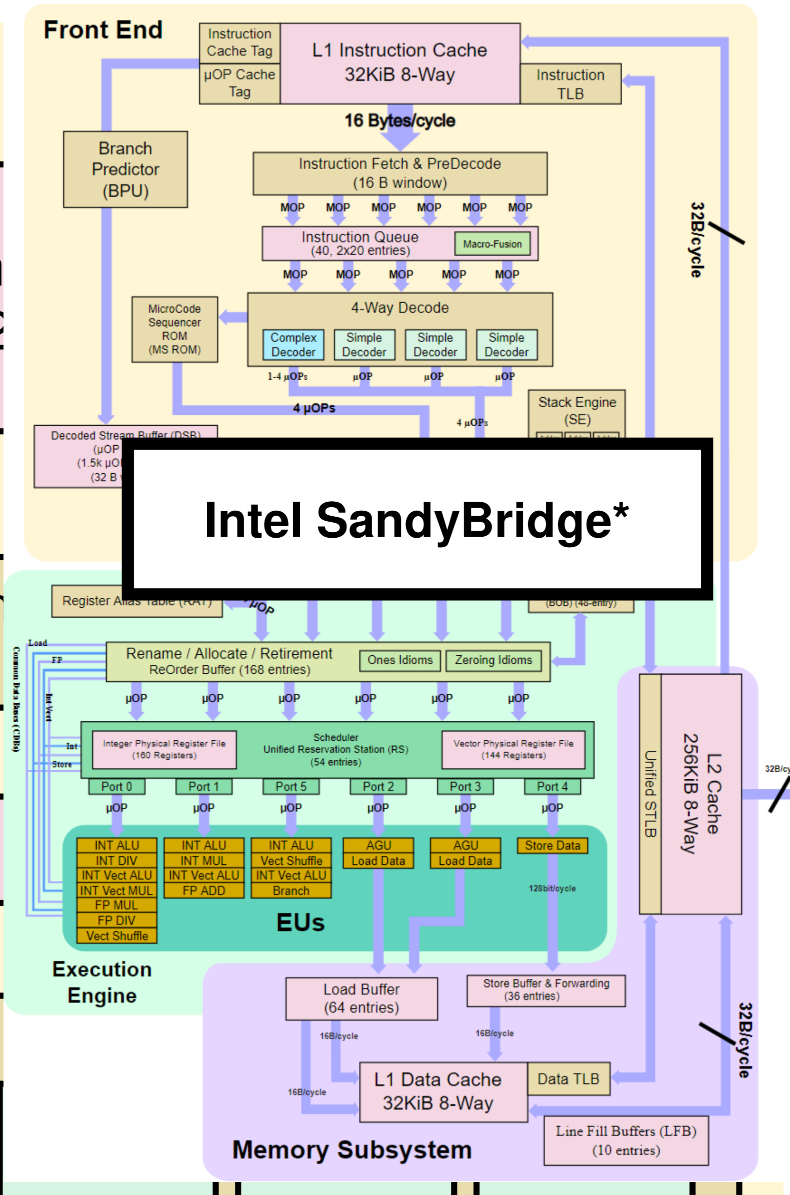
- Superscalar RISC-V OoO core
- Fully integrated in Rocket Chip ecosystem
- Open-source
- Described in Chisel
- Parameterizable generator
- Taped-out (BROOM at HC18)
- Full RV64GC ISA support
    - FP, RVC, Atomics, PMPs, VM, Breakpoints, RoCC
    - Runs real OS's, software
- Drop-in replacement for Rocket



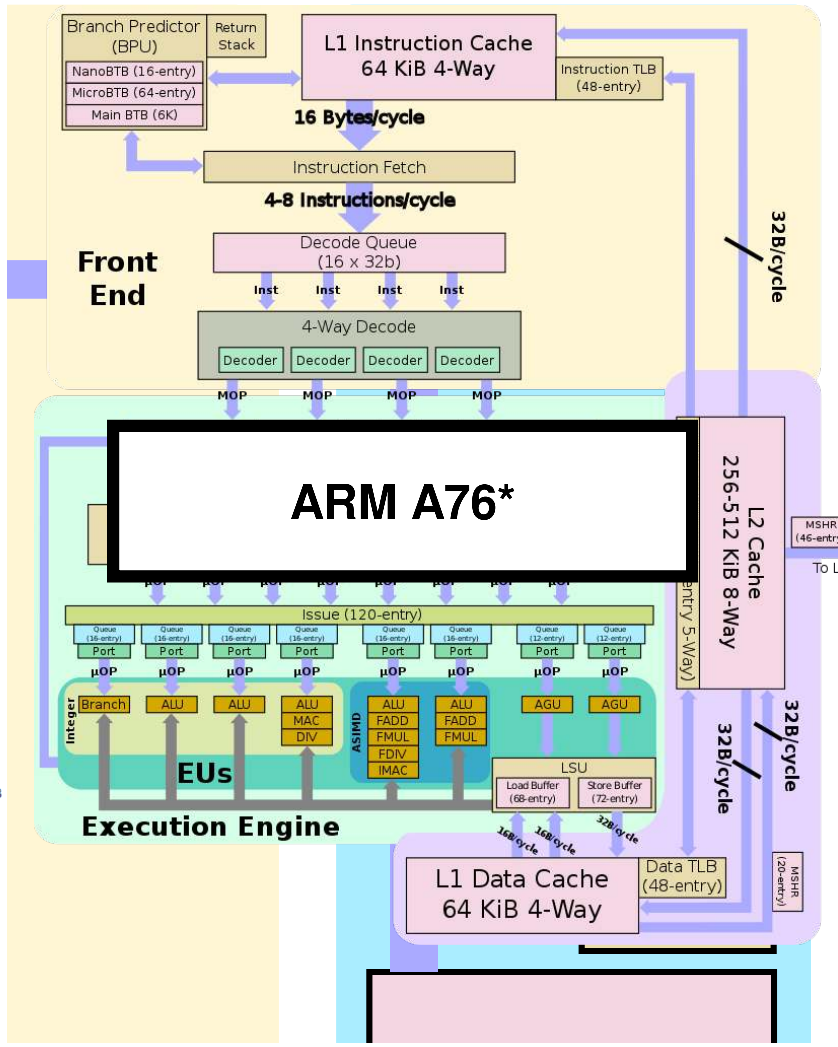
BOOM Microarchitecture



BOOM Core (“Mega” configuration)


Front End
Instruction Cache Tag
µOP Cache Tag
L1 Instruction Cache 32KiB 8-Way
Instruction TLB
16 Bytes/cycle
Branch Predictor (BPU)
Instruction Fetch & PreDecode (16 B window)
MOP
Instruction Queue (40, 2x20 entries)
Macro-Fusion
MicroCode Sequencer ROM (MS ROM)
4-Way Decode
Complex Decoder
Simple Decoder
1-4 µOPs
µOP
4 µOPs
Stack Engine (SE)
Decoded Stream Buffer (DSB)
32B/cycle
Register Alias Table (RAT)
Rename / Allocate / Retirement
ReOrder Buffer (168 entries)
Ones Idioms
Zeroing Idioms
Scheduler
Unified Reservation Station (RS) (54 entries)
Integer Physical Register File (160 Registers)
Vector Physical Register File (144 Registers)
Port 0
Port 1
Port 5
Port 2
Port 3
Port 4
INT ALU
INT DIV
INT Vect ALU
INT Vect MUL
FP MUL
FP DIV
Vect Shuffle
INT MUL
FP ADD
Branch
AGU
Load Data
Store Data
EUs
Execution Engine
Unified STLB
L2 Cache 256KiB 8-Way
To L3
Load Buffer (64 entries)
Store Buffer & Forwarding (36 entries)
16B/cycle
L1 Data Cache 32KiB 8-Way
Data TLB
Line Fill Buffers (LFB) (10 entries)
Memory Subsystem

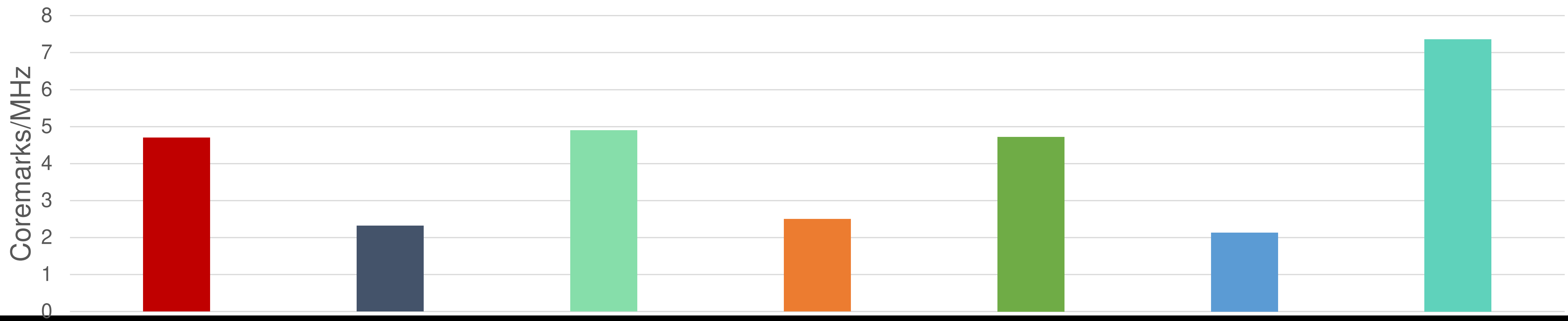
Intel SandyBridge*



ARM A76*

*Block diagram from WikiChip

Core Comparisons



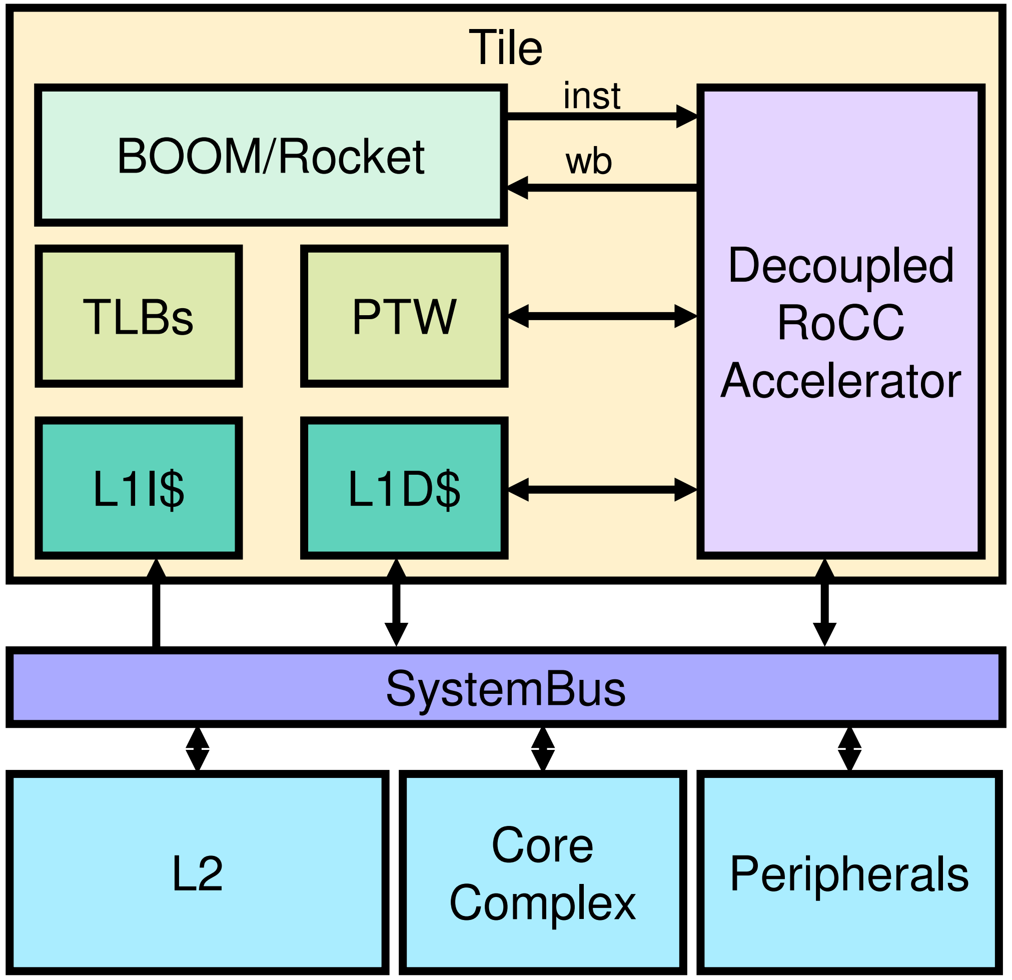
	BOOM	Rocket	WD SWERV	MIPS 74K	CortexA15	CortexA5	Intel Sandy Bridge
Type	4-w out-of-order	5-stage in-order	9-stage dual-issue	15-stage dual-issue	4-w out-of-order	8-stage in-order	6-w out-of-order
Coremarks /MHz	4.70	2.32	4.90	2.50	4.72	2.13	7.36
ISA	RV64GC	RV64GC	RV32IMC	MIPS32	ARMv7	ARMv7	X86-64
Source	10k LOC Chisel	9k LOC Chisel	25k LOC System Verilog	Closed	Closed	Closed	Closed

RoCC Accelerators

- **RoCC:** Rocket Chip Coprocessor
- Execute custom RISC-V instructions for a custom extension

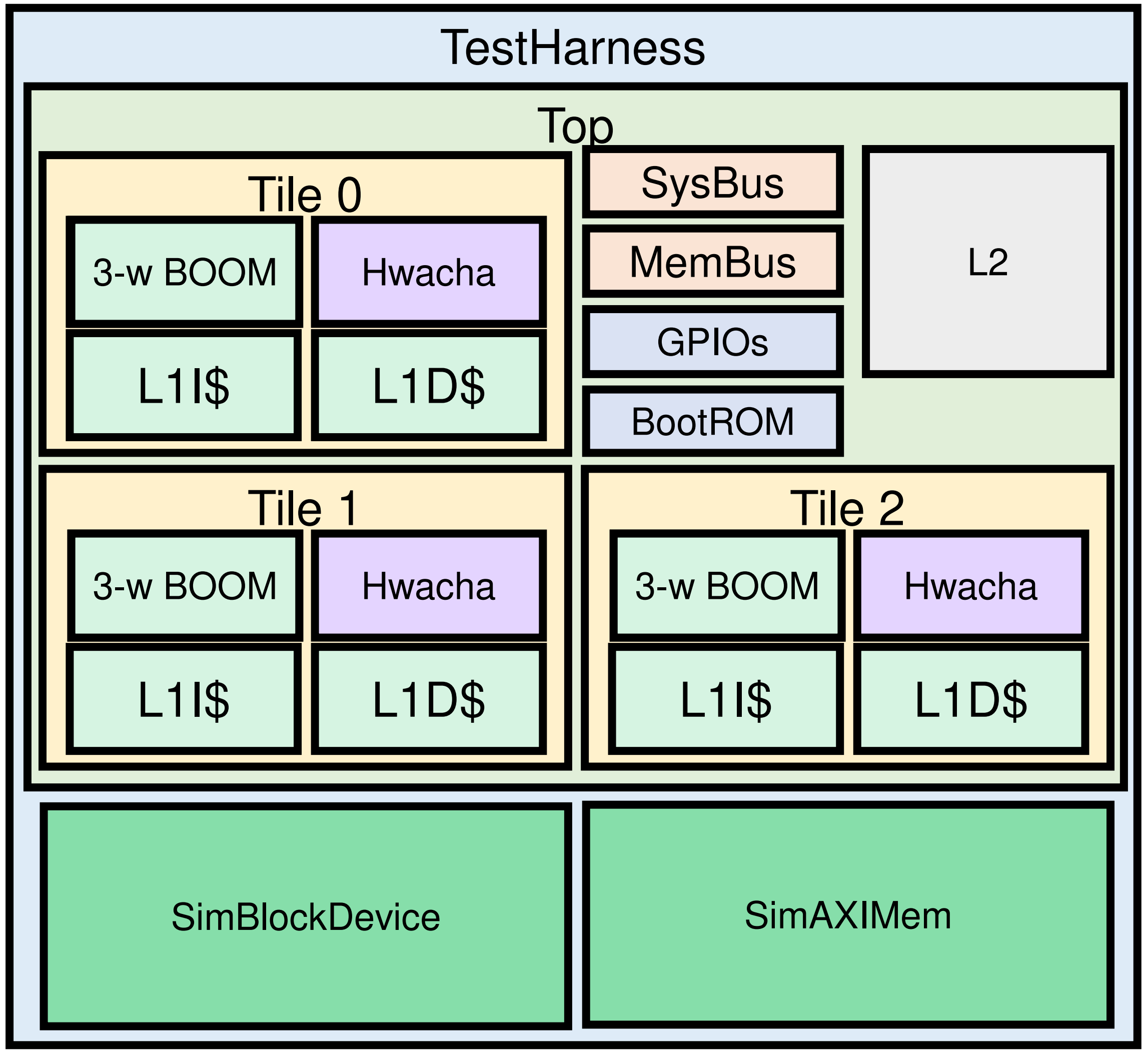
31 25	24 20	19 15	14	13	12	11 7	6 0
funct	rs2	rs1	xd	xs1	xs2	rd	opcode
7	5	5	1	1	1	5	7

- RoCC decoupled interface for connecting accelerators
- Examples of RoCC accelerators
  - Hwacha vector accelerators
  - Memcpy accelerator
  - Machine-learning accelerators
  - Java GC accelerator



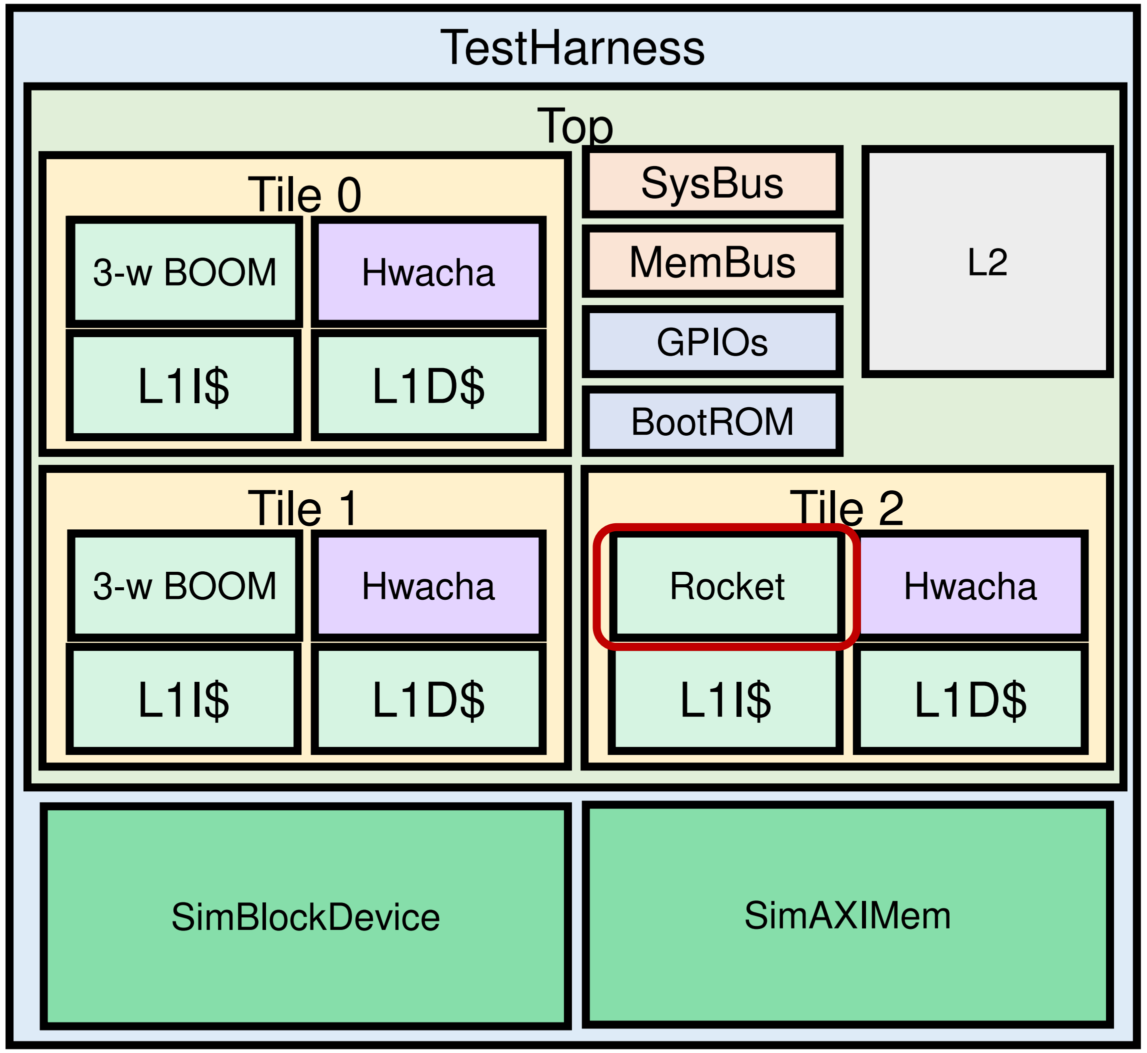
Rocket Chip Configuration

```
class MyCustomConfig extends Config(
  new WithExtMemSize((1<<30) * 2L)           ++
  new WithBlockDevice                        ++
  new WithGPIO                               ++
  new WithBootROM                            ++
  new hwacha.DefaultHwachaConfig             ++
  new WithInclusiveCache(capacityKB=1024)    ++
  new boom.common.WithLargeBooms             ++
  new boom.system.WithNBoomCores(3)          ++
  new WithNormalBoomRocketTop                ++
  new rocketchip.system.BaseConfig)
```



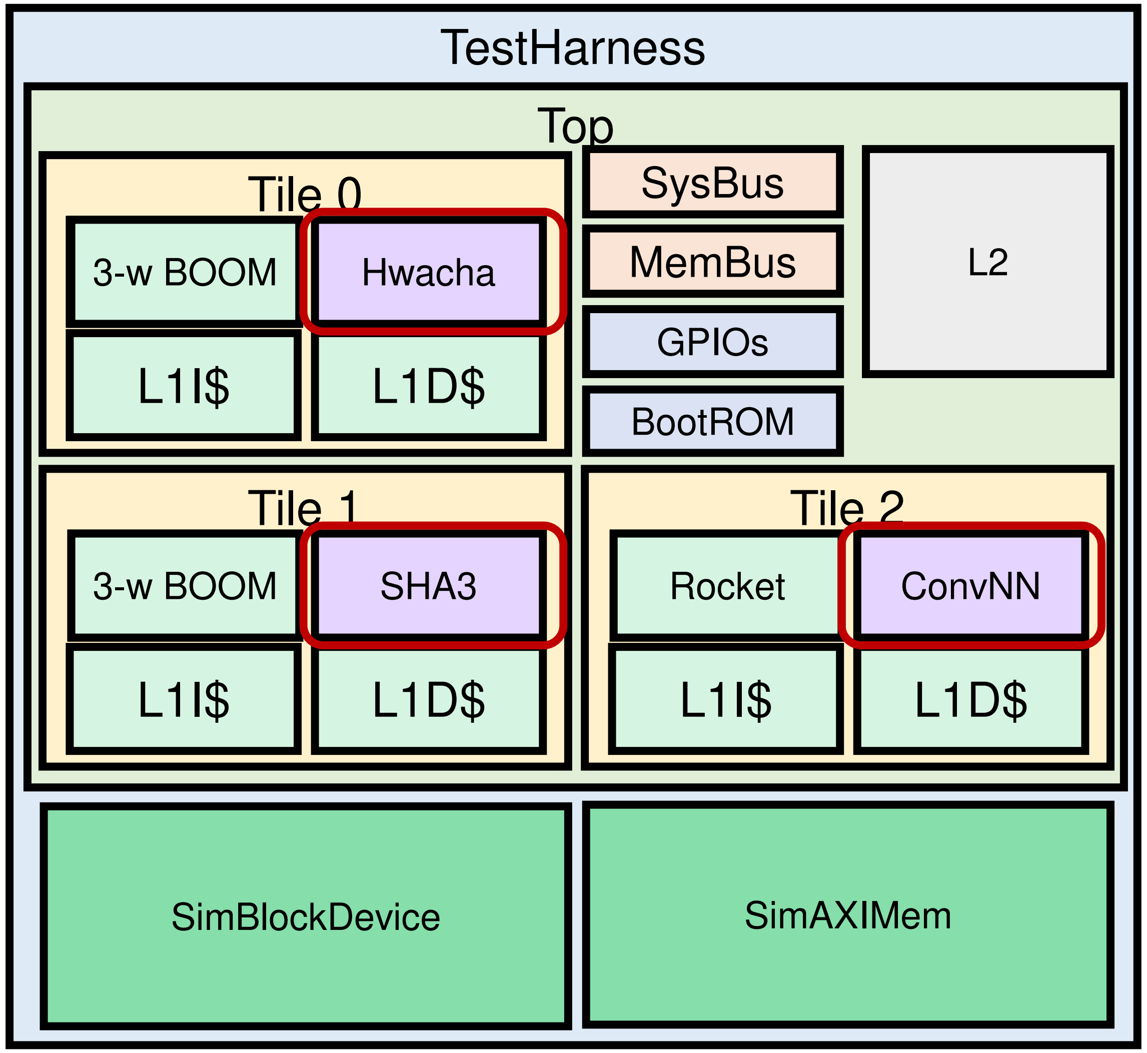
Rocket Chip Configuration

```
class MyCustomConfig extends Config(
  new WithExtMemSize((1<<30) * 2L)            ++
  new WithBlockDevice                         ++
  new WithGPIO                                ++
  new WithBootROM                             ++
  new hwacha.DefaultHwachaConfig              ++
  new WithInclusiveCache(capacityKB=1024)     ++
  new boom.common.WithLargeBooms              ++
  new boom.system.WithNBoomCores(2)           ++
  new rocketchip.subsystem.WithNBigCores(1)++
  new WithNormalBoomRocketTop                 ++
  new rocketchip.system.BaseConfig)
```



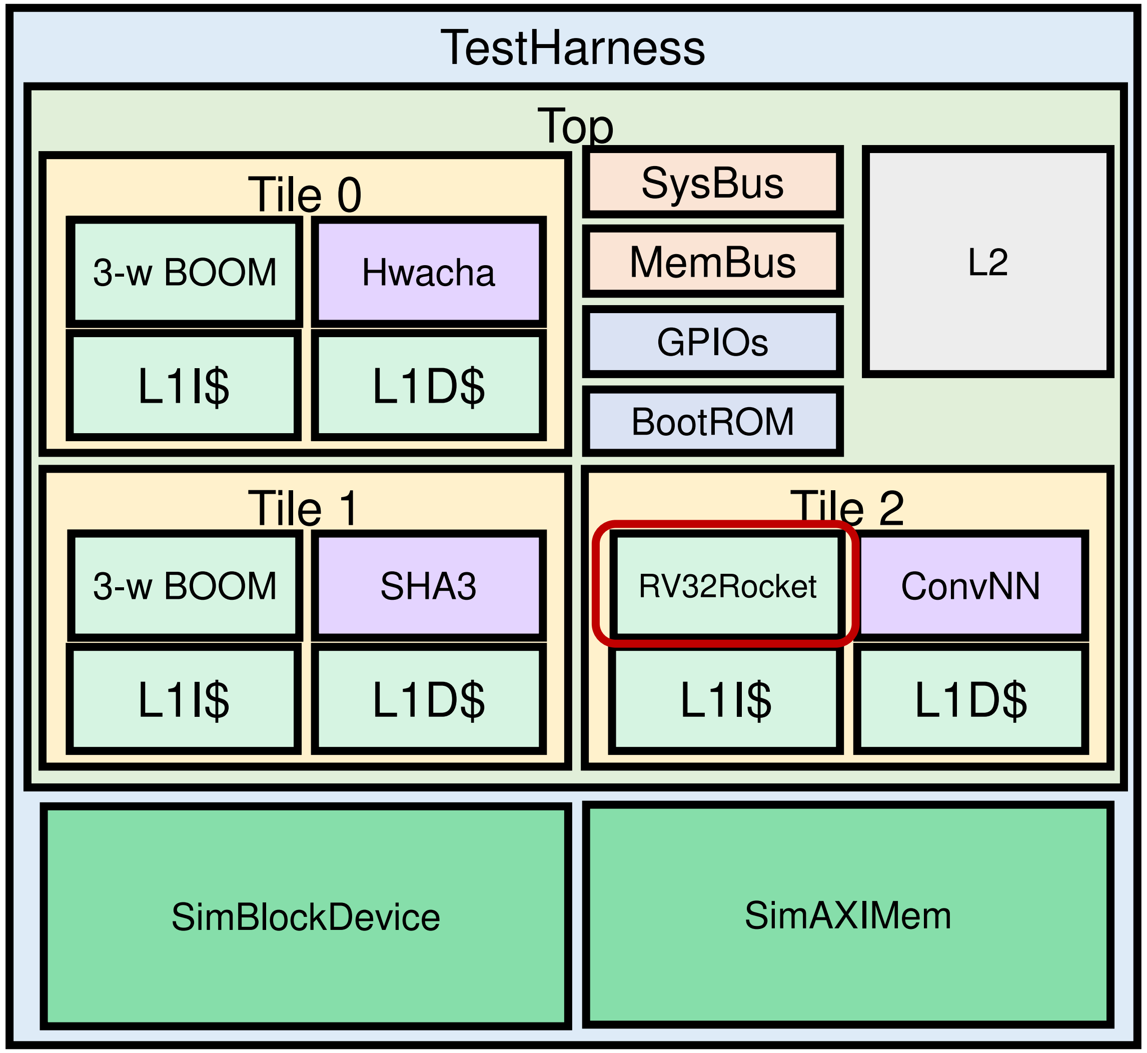
Rocket Chip Configuration

```
class MyCustomConfig extends Config(
  new WithExtMemSize((1<<30) * 2L)                   ++
  new WithBlockDevice                                ++
  new WithGPIO                                       ++
  new WithBootROM                                    ++
  new WithMultiRoCCConvAccel(2)                      ++
  new WithMultiRoCCSha3(1)                           ++
  new WithMultiRoCCHwacha(0)                         ++
  new WithInclusiveCache(capacityKB=1024)            ++
  new boom.common.WithLargeBooms                     ++
  new boom.system.WithNBoomCores(2)                  ++
  new rocketchip.subsystem.WithNBigCores(1)++
  new WithNormalBoomRocketTop                        ++
  new rocketchip.system.BaseConfig)
```



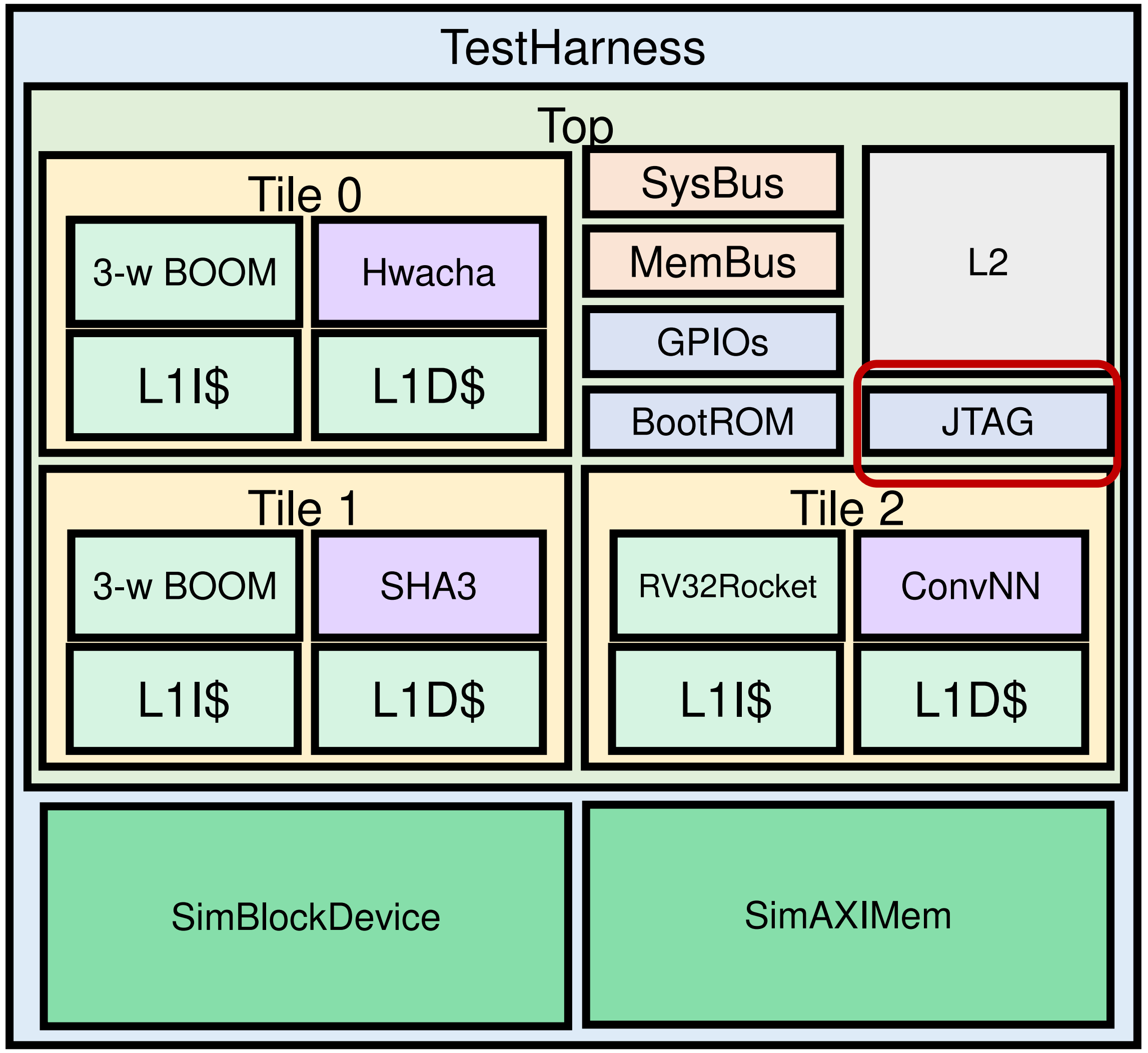
Rocket Chip Configuration

```
class MyCustomConfig extends Config(
  new WithExtMemSize((1<<30) * 2L)           ++
  new WithBlockDevice                        ++
  new WithGPIO                               ++
  new WithBootROM                            ++
  new WithMultiRoCCConvAccel(2)              ++
  new WithMultiRoCCSha3(1)                   ++
  new WithMultiRoCCHwacha(0)                 ++
  new WithInclusiveCache(capacityKB=1024)    ++
  new boom.common.WithLargeBooms             ++
  new boom.system.WithNBoomCores(2)          ++
  new rocketchip.subsystem.WithRV32          ++
  new rocketchip.subsystem.WithNBigCores(1)++
  new WithNormalBoomRocketTop                ++
  new rocketchip.system.BaseConfig)
```



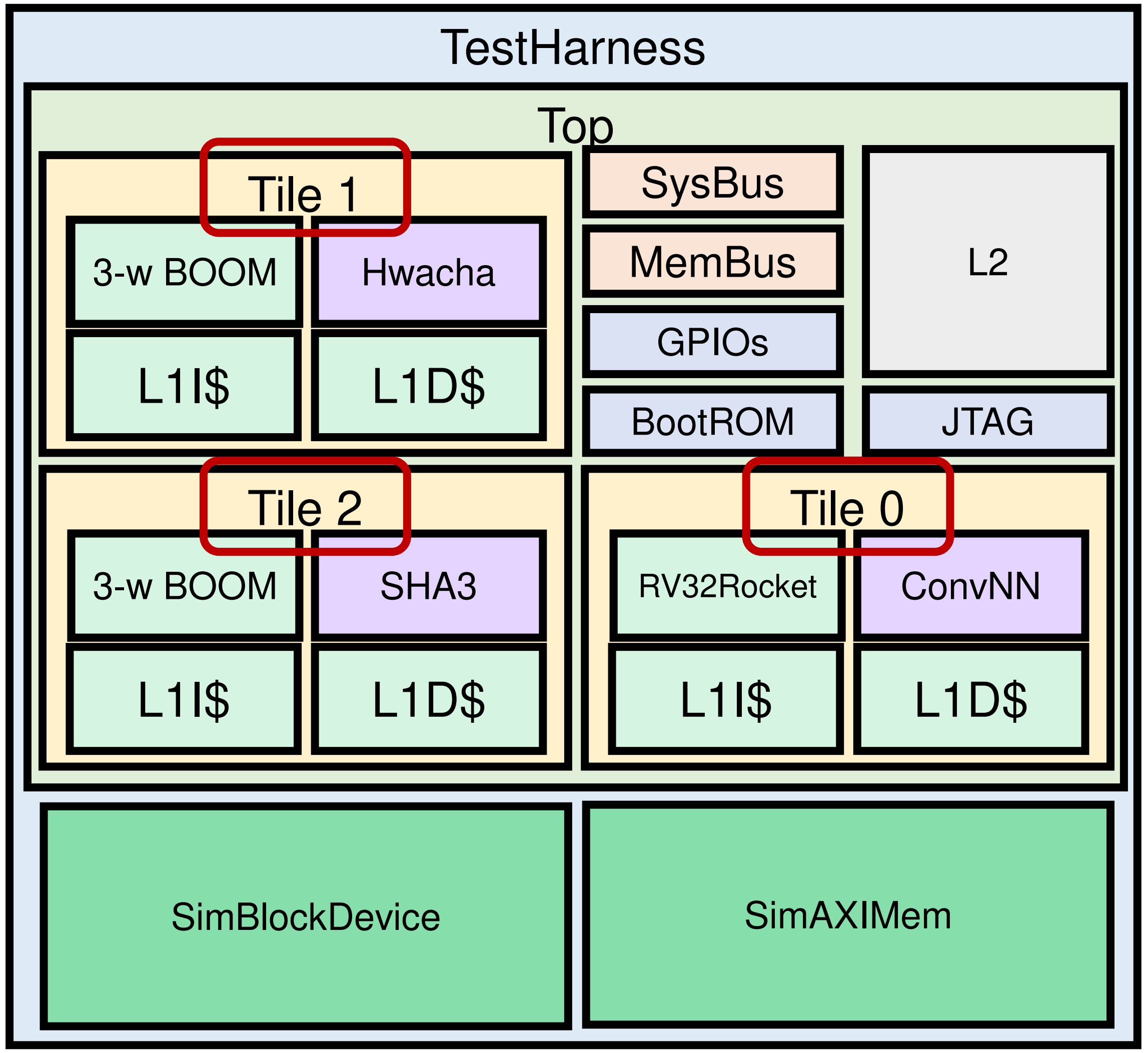
Rocket Chip Configuration

```
class MyCustomConfig extends Config(
  new WithExtMemSize((1<<30) * 2L)          ++
  new WithBlockDevice                       ++
  new WithGPIO                              ++
  new WithJtagDTM                           ++
  new WithBootROM                           ++
  new WithMultiRoCCConvAccel(2)             ++
  new WithMultiRoCCSha3(1)                  ++
  new WithMultiRoCCHwacha(0)                ++
  new WithInclusiveCache(capacityKB=1024)   ++
  new boom.common.WithLargeBooms            ++
  new boom.system.WithNBoomCores(2)         ++
  new rocketchip.subsystem.WithRV32         ++
  new rocketchip.subsystem.WithNBigCores(1)++
  new WithNormalBoomRocketTop               ++
  new rocketchip.system.BaseConfig)
```



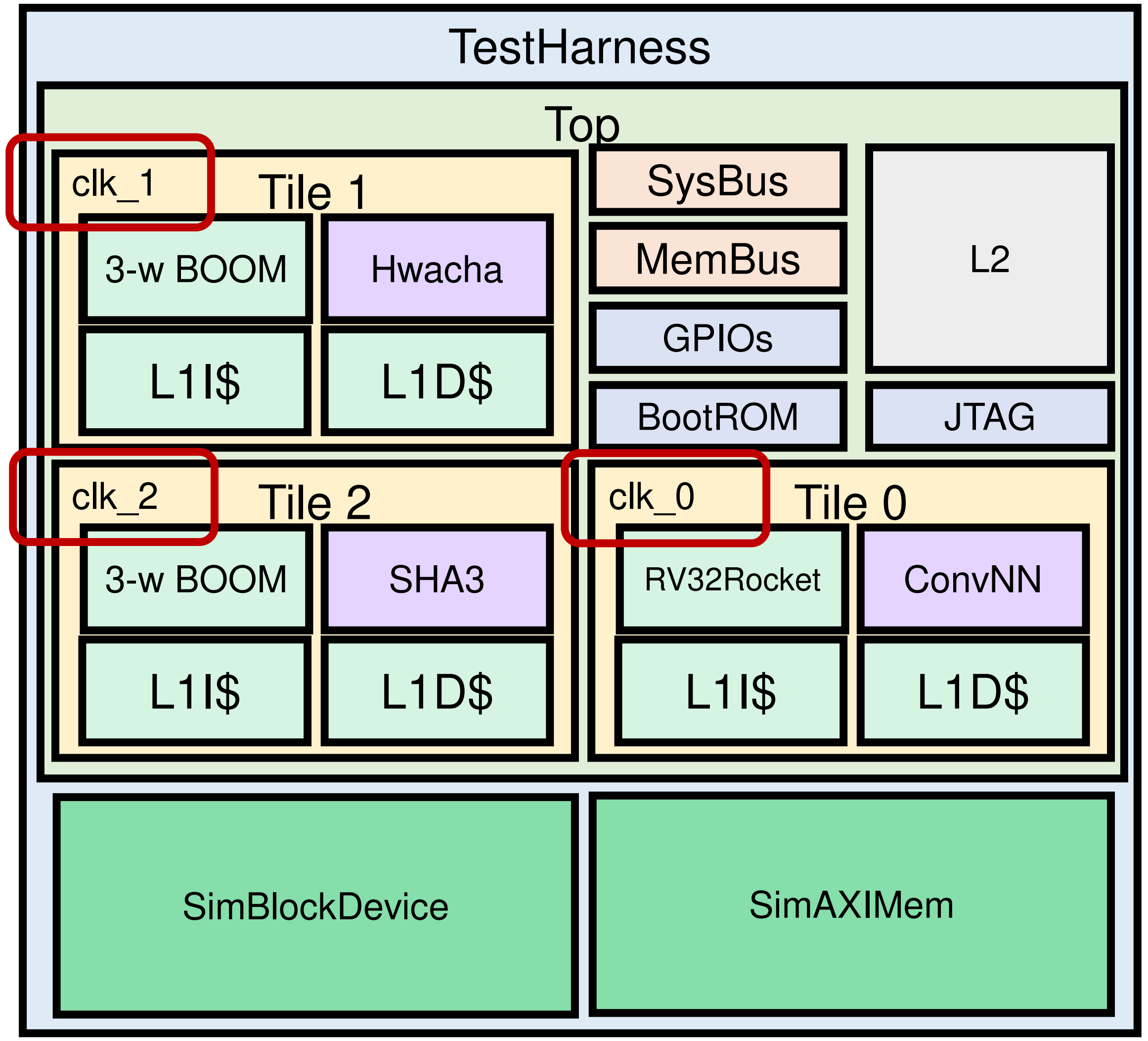
Rocket Chip Configuration

```
class MyCustomConfig extends Config(
  new WithExtMemSize((1<<30) * 2L)        ++
  new WithBlockDevice                     ++
  new WithGPIO                            ++
  new WithJtagDTM                         ++
  new WithBootROM                         ++
  new WithRenumberHarts(rocketFirst=true) ++
  new WithMultiRoCCConvAccel(2)           ++
  new WithMultiRoCCSha3(1)                ++
  new WithMultiRoCCHwacha(0)              ++
  new WithInclusiveCache(capacityKB=1024) ++
  new boom.common.WithLargeBooms          ++
  new boom.system.WithNBoomCores(2)       ++
  new rocketchip.subsystem.WithRV32       ++
  new rocketchip.subsystem.WithNBigCores(1)++
  new WithNormalBoomRocketTop             ++
  new rocketchip.system.BaseConfig)
```

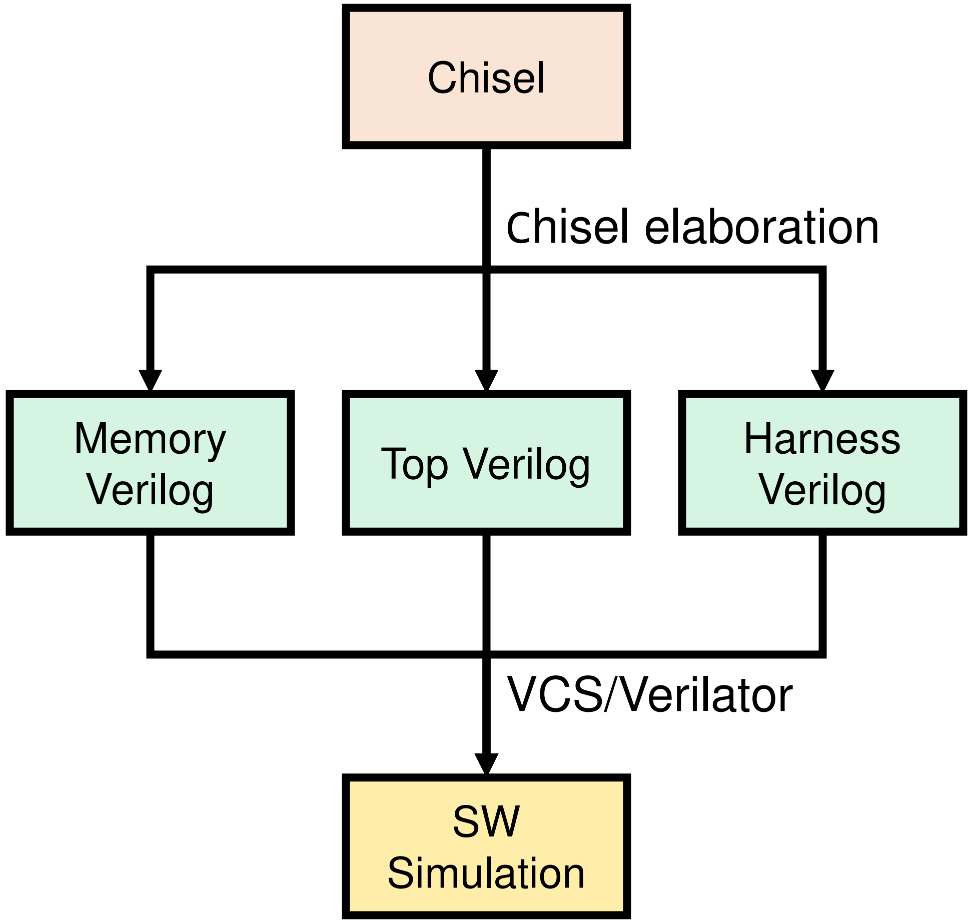


Rocket Chip Configuration

```
class MyCustomConfig extends Config(
  new WithExtMemSize((1<<30) * 2L)          ++
  new WithBlockDevice                       ++
  new WithGPIO                              ++
  new WithJtagDTM                           ++
  new WithBootROM                           ++
  new WithRenumberHarts(rocketFirst=true)   ++
  new WithRationalBoomTiles                 ++
  new WithRationalRocketTiles               ++
  new WithMultiRoCCConvAccel(2)             ++
  new WithMultiRoCCSha3(1)                  ++
  new WithMultiRoCCHwacha(0)                ++
  new WithInclusiveCache(capacityKB=1024)   ++
  new boom.common.WithLargeBooms            ++
  new boom.system.WithNBoomCores(2)         ++
  new rocketchip.subsystem.WithRV32         ++
  new rocketchip.subsystem.WithNBigCores(1)++
  new WithNormalBoomRocketTop               ++
  new rocketchip.system.BaseConfig)
```



Using Rocket Chip for SW Sim

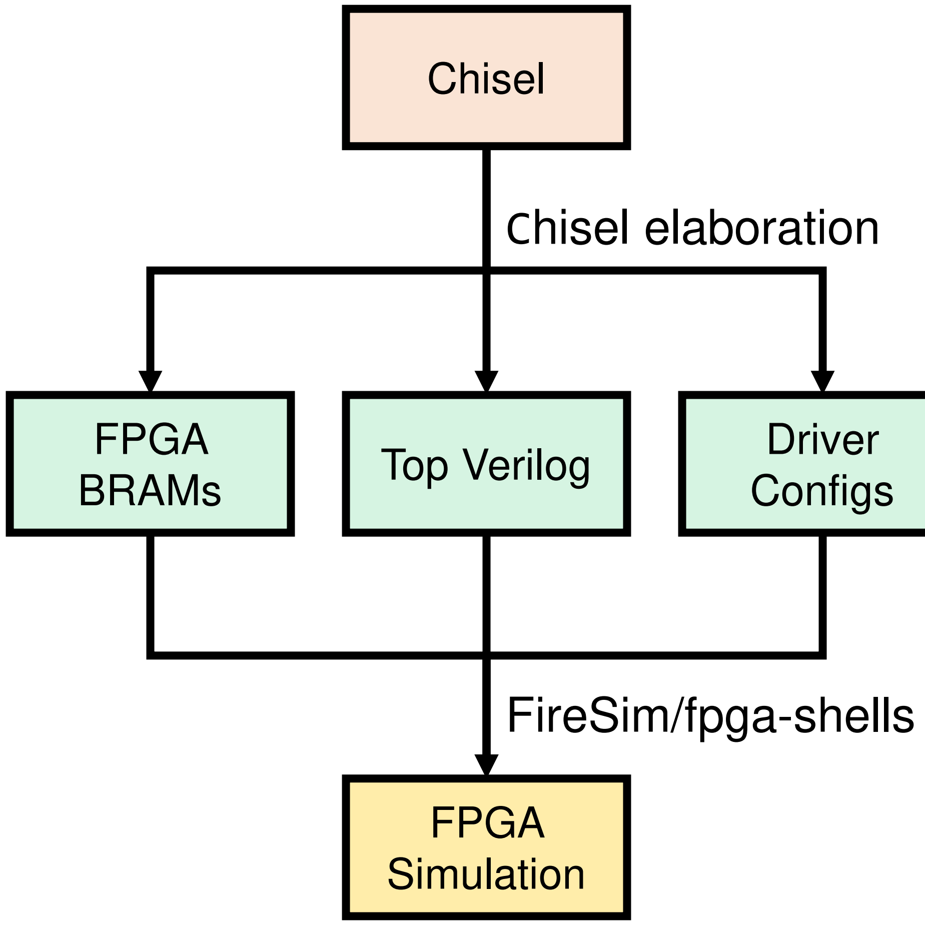


```
MyCustomConfig.scala
```

```
MyCustomConfig.top.v
MyCustomConfig.harness.v
MyCustomConfig.mems.v
```

```
./simv-MyCustomConfig
```

Using Rocket Chip for FPGA Sim

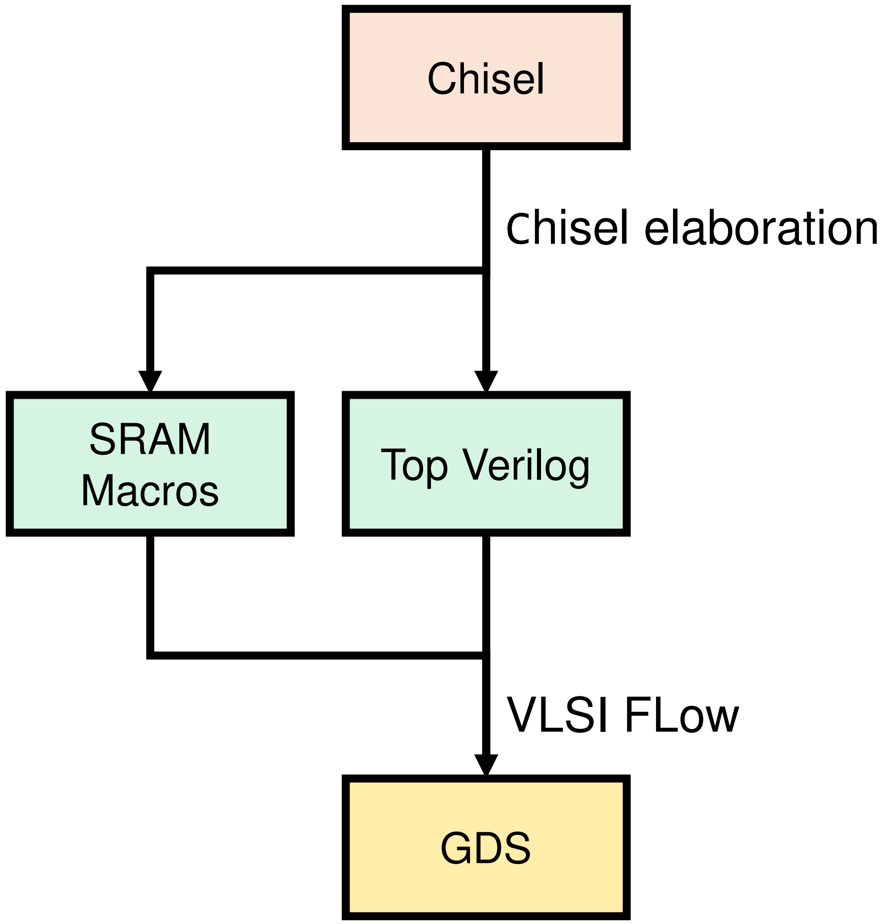


```
MyCustomConfig.scala
```

```
FPGATop.v
MyCustomConfig.mems.v
runtime.conf
FireSim-const.h
```

```
FPGA Bitstream
```

Using Rocket Chip for VLSI

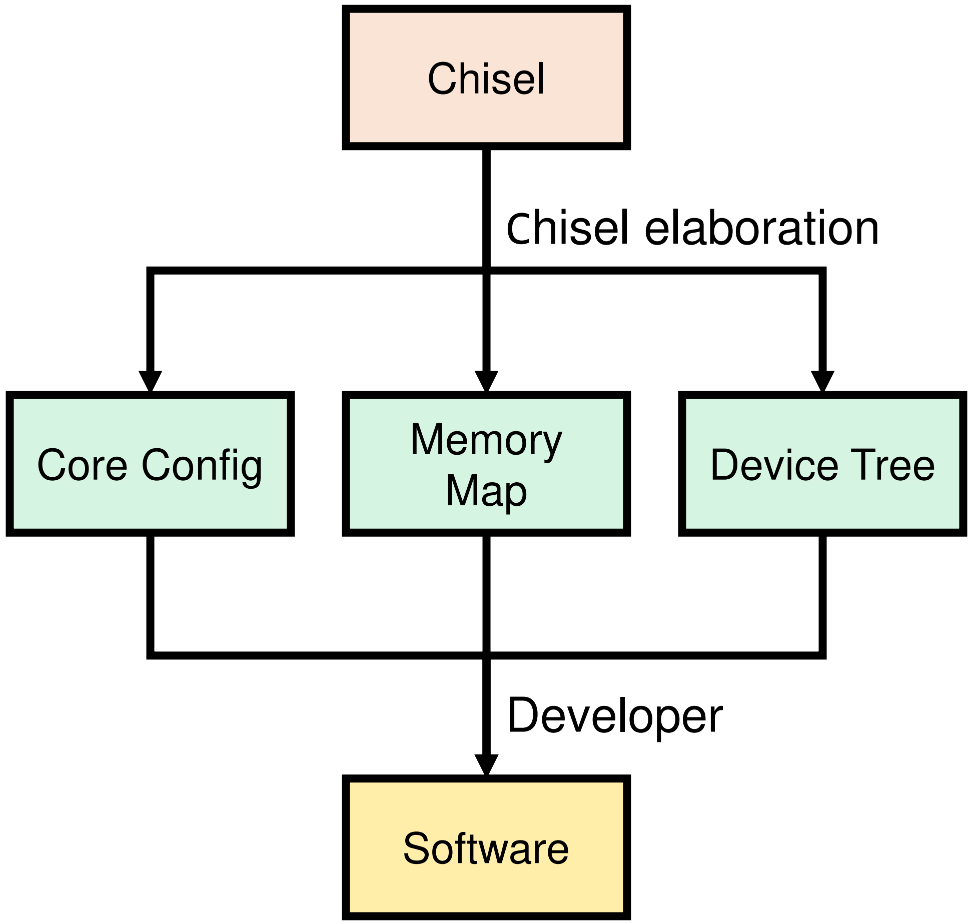


MyCustomConfig.scala

MyCustomConfig.top.v
MyCustomConfig.mems.v

MyCustomConfig.gds

Using Rocket Chip for Software

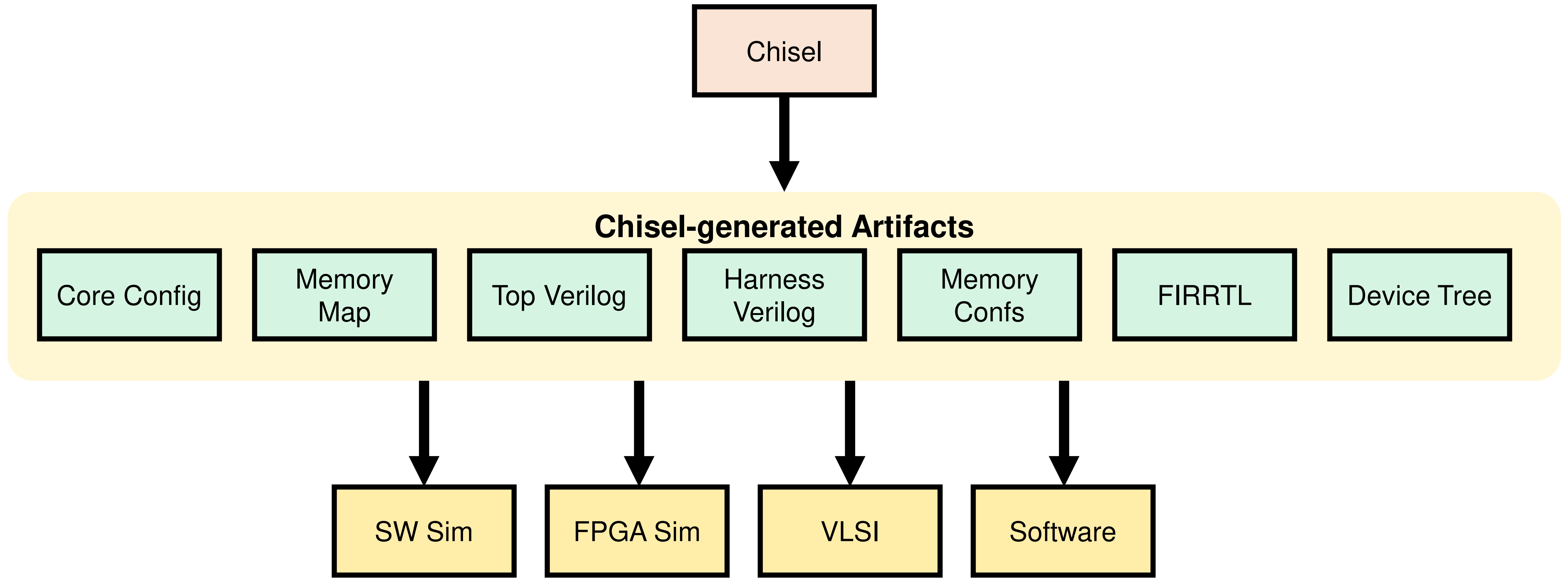


```
MyCustomConfig.scala

MyCustomConfig.core.config
MyCustomConfig.memmap.json
MyCustomConfig.dts

MyCustomSoftware.c
```

Using Rocket Chip for Everything



Research Applications

- Numerous academic tapeouts (Hurricane, BROOM, Raven, etc.)
- Designing/evaluating accelerators (Vector, GC, memcpy)
- Out-of-order core design (BOOM)
- FPGA-accelerated simulation (FireSim, MIDAS)
- Debugging methodologies (DESSERT)
- Power modeling (Strober)
- Security (Keystone)

Active Projects

- Develop more open-source components for the Rocket Chip ecosystem
- **Chipyard**: end-to-end hardware design template for Rocket Chip
- **FireSim**: FPGA simulation/debugging/profiling technologies
- **HAMMER:** Automated VLSI flows
- **BOOM:** improving performance/security, adding more features
- **Hwacha:** multi-dimensional vector execution
- **SiFive Federation:** modularize Rocket Chip
- Educational content using Rocket Chip

Links

Rocket Chip: https://github.com/chipsalliance/rocket-chip

Chipyard (Pre-release): A unified design template for Rocket Chip SoCs

- Link: https://github.com/ucb-bar/chipyard
- Docs: https://chipyard.readthedocs.io/en/dev/
- See our tutorial at MICRO 2019! https://fires.im/micro-2019-tutorial/

BOOM (Out-of-Order core): https://github.com/riscv-boom/riscv-boom

FireSim (FPGA-accelerated simulation): https://github.com/firesim/firesim

HAMMER (automated VLSI flows): https://github.com/ucb-bar/hammer

Hwacha (vector accelerator): https://github.com/ucb-bar/hwacha