Generating Rocket/BOOM SoCs with Rocket Chip

Howard Mao, Jerry Zhao

UC Berkeley Architecture Research

Hot Chips 2019
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**
Used in Many Tapeouts
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

```scala
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
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

Front End

ICache TLB*
ICache Tags*
BOOM Core ("Mega" configuration)
BTB* (1-cycle redirect)
GShare* BPU (3-cycle redirect)
Return Address Stack (RAS)

Instruction Fetch & PreDecode (4 cycles)
(16* Byte window)
Fetch Buffer (32* entries)
4*-Wide Decode
Decoder

Instruction Fetch & PreDecode (16 B window)
4-Way Decode
Stack Engine (16)

16 Bytes/cycle
Inst
Inst
Inst
Inst
Inst
Inst
Inst

128 bit/cycle
L1 Cache
L2 Cache
L2 TLB*
L1 Data Cache
Memory Subsystem
Execution Engine
EUs

Intel SandyBridge*
ARM A76*

*Block diagram from WikiChip
## Core Comparisons

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

- **RoCC**: Rocket Chip Coprocessor
- Execute custom RISC-V instructions for a custom extension
- RoCC decoupled interface for connecting accelerators
- Examples of RoCC accelerators
  - Hwacha vector accelerators
  - Memcpy accelerator
  - Machine-learning accelerators
  - Java GC accelerator
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)
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))
class MyCustomConfig extends Config(
    new WithExtMemSize(((1L << 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)

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)

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)

TestHarness

Top

Tile 0

SysBus

MemBus

GPIOs

BootROM

JTAG

Tile 1

3-w BOOM

Hwacha

L1$I$

L1$D$

Tile 2

3-w BOOM

SHA3

L1$I$

L1$D$

JTAG

SimBlockDevice

SimAXIMem

RV32Rocket

ConvNN

L2
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)
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 WithMultiRoCCWacha(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

- Chisel
  - Chisel elaboration
    - Memory Verilog
    - Top Verilog
    - Harness Verilog
      - VCS/Verilator
        - SW Simulation

- 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

Chisel

Chisel elaboration

SRAM Macros

Top Verilog

VLSI Flow

GDS

MyCustomConfig.scala

MyCustomConfig.top.v

MyCustomConfig.mems.v

MyCustomConfig.gds
Using Rocket Chip for Software

- Chisel
  - Chisel elaboration
    - Core Config
    - Memory Map
    - Device Tree
  - Developer
    - Software

- MyCustomConfig.scala
- MyCustomConfig.core.config
- MyCustomConfig.memmap.json
- MyCustomConfig.dts
- MyCustomSoftware.c
Using Rocket Chip for Everything

Chisel

Chisel-generated Artifacts

- Core Config
- Memory Map
- Top Verilog
- Harness Verilog
- Memory Conf
- FIRRTL
- Device Tree

- SW Sim
- FPGA Sim
- VLSI
- Software
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/


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