
Christopher Celio, Krste Asanovic, David Patterson

2016 Jan

celio@eecs.berkeley.edu
What is BOOM?

- superscalar, out-of-order processor written in Berkeley’s Chisel RTL
- It is synthesizable
- It is parameterizable
- We hope to use it as a platform for architecture research
- It is open-source!

BOOM is a work-in-progress. Results shown in the talk are preliminary and subject to change!
Open-source Berkeley RISC-V Processors

- **Sodor** Collection
  - RV32I - tiny, educational, not synthesizable
- **Rocket-chip** SoC generator
  - Z-scale
    - RV32IM - micro-controller
  - Rocket
    - RV64G - in-order, single-issue application core
- **BOOM**
  - RV64G - out-of-order, superscalar application core
Why Out-of-Order?

- Great for ...
  - tolerating **variable** latencies
  - finding **ILP** in code (instruction-level parallelism)
  - complex method for fine-grain data **prefetching**
  - plays nicely with **poor** compilers and **lazily** written code

- Downsides
  - complicated
  - expensive (area & power)

Performance! (and easy to program!)
OoO is widely used in industry

- Intel Xeon/i-series (10-100W)
- ARM Cortex mobile chips (1W)
- Sun/Oracle Niagra UltraSPARC
- Intel Atom
- Play Station
Goals

- build a prototypical OoO core
  - provide an open-source implementation for education, research, and industry
  - serve as a baseline for micro-architecture research
- enable studies that need an OoO core
  - research memory systems
  - research accelerators
  - research VLSI methodologies
- use BOOM to get detailed performance, area, and power running real programs
  - most research uses sw simulators for performance (100 KIPS), RTL for area
Berkeley Architecture Research Infrastructure

- RISC-V ISA
- Rocket-chip SoC generator
- Chisel HCL (hardware construction language)
- bar.eecs.berkeley.edu
BOOM Implements IMAFD

- “M” (mul/div/rem)
  - imul can be either pipelined or unpipelined
- "A"
  - AMOs+LR/SC
- “FD”
  - single, double-precision floating point
  - IEEE 754-2008 compliant FPU
  - SP, DP FMA with hw support for subnormals
  - parameterized latency
- RV64G + privileged spec (page-based virtual memory)
The RISC-V ISA is easy to implement!

- relaxed memory model
- accrued FP exception flags
- no integer side-effects (e.g., condition codes)
- no cmov or predication
- no implicit register specifiers
  - JAL requires explicit rd
- rs1, rs2, rs3, rd always in same space
  - allows decode, rename to proceed in parallel
Rocket-Chip SoC Generator

- open-source
- taped out 11 times by Berkeley
- runs at 1.6 GHz in IBM 45nm
- makes for a great library of processor components!
- swap out Rocket tile and replace with a BOOM tile
- as Rocket-chip gets better, so does BOOM

*slightly out-of-date uncore (now uses AXI instead of MemIO)
Rocket-Chip Updates

- uncached loads/stores
- memory-mapped IO
- replaced memIO with AXI
- Work in Progress
  - remove HTIF (untether)

*slightly out-of-date uncore (now uses AXI instead of MemIO)*
Chisel

- Hardware Construction Language embedded in Scala
- **not** a high-level synthesis language
- hardware module is a data structure in Scala
- Full power of Scala for writing generators
  - object-oriented programming
  - factory objects, traits, overloading
  - functional programming
  - high-order funs, anonymous funcs, currying
- generated C++ simulator is **1:1** copy of Verilog designs
- **version 3.0 coming soon!**
  - uses an IR called FIRRTL
BOOM Pipeline

Fetch → Decode & Rename → Issue Window → Unified Physical Register File → Functional Unit

in-order front-half
out-of-order back-half
- **PRF**
  - explicit renaming
  - holds speculative and committed data
  - holds both x-regs, f-regs
- **split ROB/issue window design**
- **Unified Issue Window**
  - holds all instructions
Parameterized Superscalar

dual-issue (5r,3w)

OR

Quad-issue (9r,4w)

val exe_units = ArrayBuffer[ExecutionUnit]()
exe_units += Module(new ALUExeUnit(is_branch_unit = true,
                                   has_fpu = true,
                                   has_mul = true))
exe_units += Module(new ALUExeUnit(fp_mem_support = true,
                                   has_div = true))

exe_units += Module(new ALUExeUnit(is_branch_unit = true))
exe_units += Module(new ALUExeUnit(has_fpu = true,
                                   has_mul = true))
exe_units += Module(new ALUExeUnit(has_div = true))
exe_units += Module(new MemExeUnit())

exe_units += Module(new ALUExeUnit(is_branch_unit = true))
exe_units += Module(new ALUExeUnit(has_fpu = true,
                                   has_mul = true))
exe_units += Module(new ALUExeUnit(has_div = true))
exe_units += Module(new MemExeUnit())
A Functional Unit Hierarchy

- **Abstract FunctionalUnit**
  - describes common IO
- **Pipelined/Unpipelined**
  - handles storing uop metadata, branch resolution, branch kills
- **Concrete Subclasses**
  - instantiates the actual expert-written FU
  - no modifications required to get FU working with speculative OoO
  - allows easy “stealing” of external code

- Pipelined (req.ready == true)
- UnPipelined

  - iMul
  - iMul/iDiv/iRem
  - fDiv/fSqrt

  - ALU (w/ optional Br Unit)
  - MemAddrCalc
  - FPU
    - ALU
    - DFMA
    - SFMA
    - FPTo FP
    - FPTo Int
    - IntTo FP
    - FPTo FP
Some of the Parameters

- fetch/decode/commit width (1,2,4)
- issue width
  - functional unit mix
- un-ordered vs age-ordered issue scheduler (R10k vs R12k)
- enable commit map table (R10k roll-back vs. Alpha 21264 single-cycle flush)
- rob size
- issue window size
- lsu size
- number of physical registers
- max inflight branches
- FPU latencies, iMul latencies, unpipelined mul/div bits per stage
- fetch buffer size
- flow-through fetch buffer
- enable backing branch predictor (and it's set of parameters...)
- enable uarch counters
- RAS size, BTB size
- L1 cache sets, ways, mshrs
- enable L2 (and L2 parameters...)
- Targeting ASIC
- Runs on FPGA
  - Zynq zc706

2-wide BOOM, 16kB L1 caches
1.2 mm²

preliminary results
How fast can BOOM be clocked?

- depends on parameters
- +1.5 GHz for two-wide
- (currently) designed for single-cycle SRAM access as critical path
- upcoming tape-out will keep us honest!
- ... and 50 MHz on FPGA
  - bottleneck is FPGA tools which can't register-retime the FPU
Full Branch Speculation Support

- next-line predictor (NLP)
  - BTB, BHT, RAS
  - combinational
- backing predictor (BPD)
  - global history predictor
  - SRAM (1 r/w port)
Load/Store Unit

- talks to Rocket's non-blocking data cache
- load/store queue with store ordering
  - loads execute fully out-of-order wrt stores, other loads
    - (based on current understanding of the RISC-V memory model)
  - store-data forwarded to loads as required
## Feature Summary

<table>
<thead>
<tr>
<th>Feature</th>
<th>BOOM</th>
</tr>
</thead>
<tbody>
<tr>
<td>ISA</td>
<td>RISC-V (RV64G)</td>
</tr>
<tr>
<td>Synthesizable</td>
<td>√</td>
</tr>
<tr>
<td>FPGA</td>
<td>√</td>
</tr>
<tr>
<td>Parameterized</td>
<td>√</td>
</tr>
<tr>
<td>floating point</td>
<td>√</td>
</tr>
<tr>
<td>AMOs+LR/SC</td>
<td>√</td>
</tr>
<tr>
<td>caches</td>
<td>√</td>
</tr>
<tr>
<td>VM</td>
<td>√</td>
</tr>
<tr>
<td>Boots Linux</td>
<td>√</td>
</tr>
<tr>
<td>Multi-core</td>
<td>√</td>
</tr>
<tr>
<td>lines of code</td>
<td>9k + 11k</td>
</tr>
<tr>
<td>Category</td>
<td>ARM Cortex-A9</td>
</tr>
<tr>
<td>-------------------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td>ISA</td>
<td>32-bit ARM v7</td>
</tr>
<tr>
<td>Architecture</td>
<td>2 wide, 3+1 issue Out-of-Order 8-stage</td>
</tr>
<tr>
<td>Performance</td>
<td>3.59 CoreMarks/MHz</td>
</tr>
<tr>
<td>Process</td>
<td>TSMC 40GPLUS</td>
</tr>
<tr>
<td>Area with 32K caches</td>
<td>~2.5 mm²</td>
</tr>
<tr>
<td>Area efficiency</td>
<td>1.4 CoreMarks/MHz/mm²</td>
</tr>
<tr>
<td>Frequency</td>
<td>1.4 GHz</td>
</tr>
<tr>
<td>Power</td>
<td>0.5-1.9 W (2 cores + L2)</td>
</tr>
<tr>
<td></td>
<td>@ TSMC 40nm, 0.8-2.0 GHz</td>
</tr>
</tbody>
</table>

+9%!

note: not to scale
Challenges executing SPECint

- 12 benchmarks in CINT2006
  - 35 workloads written in C or C++ (--reference)
  - average 2 trillion instructions
  - ~24T instructions total
    - (2 hours for a 3 GHz Intel processor at CPI=1)
    - (7 years for a sw simulator @ 100 KIPS)

- RISC-V
  - requires libc (riscv64-unknown-linux-gnu*)
  - run on Linux
  - https://github.com/ccelio/Speckle
    - useful for generating portable SPEC directories
Preliminary Data

gcc.0.boom2w-l2 (reference)
- TODO :(
- need more DRAM
- need DRAM emulation
- will take ~1 day with cluster

**SPEC Score**

- astar.1
- h264.1
- bzip2.2
How to use BOOM...
Repositories

- **BOOM**
  - [https://github.com/ucb-bar/riscv-boom](https://github.com/ucb-bar/riscv-boom)
  - just the BOOM core code

- **Rocket-chip**
  - [https://github.com/ucb-bar/rocket-chip](https://github.com/ucb-bar/rocket-chip)
  - the rest of the SoC
  - chisel, rocket, uncore, junctions, riscv-tools, fpga-zynq, etc.
  - generates C++ emulator, verilog for FPGA
  (for more information)
BOOM Quick-start

- "BOOM-chip" is (currently) a branch of rocket-chip
- "make run" builds and runs riscv-tests suite
- there are many different CONFIGs available!
  - rocket-chip/src/main/scala/PrivateConfigs.scala
  - rocket-chip/boom/src/main/scala/configs.scala
  - slightly different configs for different targets (CPP, FPGA, VLSI)

$ export ROCKETCHIP_ADDONS="boom"
$ git clone https://github.com/ucb-bar/rocket-chip.git
$ cd rocket-chip
$ git checkout boom
$ git submodule update --init
$ cd riscv-tools
$ git submodule update --init --recursive riscv-tests
$ cd ../emulator; make run CONFIG=BOOMCPPConfig
How do you verify and debug BOOM?

- parameterization makes this tough!
- tested with:
  - riscv-tests
    - (assembly functional tests + bare-metal micro-benchmarks)
  - CoreMark + riscv-pk
  - SPEC + Linux
  - riscv-torture
riscv-torture

- now open-source!
  - https://github.com/ucb-bar/riscv-torture
- torture generates random tests to stress the core pipeline
- runs test on Spike and your processor
  - architectural register state dumped to memory on program termination
  - diff state
  - if error found, finds smallest version of test that exhibits an error
riscv-torture quick-start

Run Rocket-chip (BOOM-chip?)

```bash
$ git clone https://github.com/ucb-bar/rocket-chip.git
$ cd rocket-chip
$ git submodule update --init
$ cd riscv-tools
$ git submodule update --init --recursive riscv-tests
$ cd ../emulator; make run CONFIG=BOOMCPPConfig
```

Run Torture

```bash
$ cd rocket-chip
$ git clone https://github.com/ucb-bar/riscv-torture.git
$ cd riscv-torture
$ git submodule update --init
$ vim Makefile # change RTL_CONFIG=BOOMCPPConfig
$ make igentest # test that torture works, gen a single test
$ make cnight # run C++ emulator overnight
```
Commit Logging

- BOOM can generate commit logs
  - (priv level, PC, inst, wb raddr, wb data)
- Spike supports generating commit logs!
  - configure Spike with "--enable-commitlog" flag
  - outputs to stderr
- only semi-automated
  - valid differences from Spike and hw
  - e.g., spinning on LR+SC

```
# modify the build.sh in riscv-tools
build_project riscv-isa-sim --prefix=$RISCV --with-fesvr=$RISCV --enable-commitlog
```
What documentation is available?

- A design document is in progress
  - https://github.com/cCELio/riscv-boom-doc
- Wiki
  - https://github.com/UCB-Bar/riscv-boom/wiki
Future Plans

- tape-out this year
- uncached loads/stores
- update to Chisel 3.0
- SPEC score
  - improve branch prediction
  - memory ordering speculation
- finish documentation
- build a community of Baby Boomers?
Conclusion

- BOOM is RV64G, runs SPEC on Linux on an FPGA
- BOOM is ~10k loc and 4 person-years of work
- excellent platform for prototyping new ideas
- now open-source!
- we'll continue to support and improve BOOM
Questions?
Funding Acknowledgements

- Research partially funded by DARPA Award Number HR0011-12-2-0016, the Center for Future Architecture Research, a member of STARnet, a Semiconductor Research Corporation program sponsored by MARCO and DARPA, and ASPIRE Lab industrial sponsors and affiliates Intel, Google, Huawei, Nokia, NVIDIA, Oracle, and Samsung.

- Approved for public release; distribution is unlimited. The content of this presentation does not necessarily reflect the position or the policy of the US government and no official endorsement should be inferred.

- Any opinions, findings, conclusions, or recommendations in this paper are solely those of the authors and does not necessarily reflect the position or the policy of the sponsors.
Synthesis Results

Core Area (um^2)

- BOOM
  - Other
  - Imul
  - FPU
  - FetchBuffer
  - BusyTable
  - Freelist
  - Br Predictor
  - LSU
  - ROB
  - Register File
  - RRd Stage (bypasses)
  - Rename Stage (maptables)
  - Issue Unit

- Rocket
  - Issue Unit

Preliminary results
Synthesis Results

Core Area (um^2)

Tile Area (um^2)

Issue Unit
RRd Stage (bypasses)
ROB
Br Predictor
BusyTable
FPU
Other

Rename Stage (maptables)
Register File
LSU
Freelist
FetchBuffer
Imul

preliminary results
Industry Comparisons

CoreMark/MHz

out-of-order processors

Ivy Bridge
Cortex-A15
BOOM-4w
BOOM-2w
Cortex-A9
MIPS74k
Cortex-A8
Rocket
Cortex-A5

in-order processors

preliminary results
### Industry Comparisons

<table>
<thead>
<tr>
<th>Processor</th>
<th>Core Area</th>
<th>CoreMark/ MHz</th>
<th>Freq (MHz)</th>
<th>IPC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intel Xeon E5 2668 (Ivy)</td>
<td>~12 mm²@22nm</td>
<td>5.6</td>
<td>3,300</td>
<td>1.96</td>
</tr>
<tr>
<td>ARM Cortex-A15</td>
<td>2.8 mm²@28nm</td>
<td>4.72</td>
<td>2,116</td>
<td>1.5</td>
</tr>
<tr>
<td>BOOM-4wide</td>
<td>1.1 mm²@45nm</td>
<td>4.7</td>
<td>1,000</td>
<td>1.5</td>
</tr>
<tr>
<td>BOOM-2wide</td>
<td>0.8 mm²@45nm</td>
<td>3.91</td>
<td>1,500</td>
<td>1.26</td>
</tr>
<tr>
<td>ARM Cortex-A9</td>
<td>2.5 mm²@40nm</td>
<td>3.59</td>
<td>1,400</td>
<td>1.27</td>
</tr>
<tr>
<td>MIPS 74K</td>
<td>2.5 mm²@65nm</td>
<td>2.5</td>
<td>1,600</td>
<td>-</td>
</tr>
<tr>
<td>Rocket (RV64G)</td>
<td>0.5 mm²@45nm</td>
<td>2.32</td>
<td>1,500</td>
<td>0.76</td>
</tr>
<tr>
<td>ARM Cortex-A5</td>
<td>0.5 mm²@40nm</td>
<td>2.13</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

48x preliminary results
Ivy Bridge Tile Comparison

Ivy Bridge-EP Tile
(32kB/32kB + 256kB caches)
~12nm @ 22nm

BOOM-2w Chip
(32kB/32kB + 256kB caches)
1.7mm² @ 45nm

BOOM-2w Chip
scaled to 0.4mm² @ 22nm
preliminary results