The Berkeley Out-of-Order Machine (BOOM): An Industry-Competitive, Synthesizable, Parameterized RISC-V Processor

Christopher Celio
David A. Patterson
Krste Asanović

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2015-167
http://www.eecs.berkeley.edu/Pubs/TechRpts/2015/EECS-2015-167.html

June 13, 2015

The Berkeley Out-of-Order Machine (BOOM): An Industry-Competitive, Synthesizable, Parameterized RISC-V Processor

Christopher Celio, David Patterson, and Krste Asanović
University of California, Berkeley, California 94720–1770
celio@eecs.berkeley.edu

BOOM is a work-in-progress. Results shown are preliminary and subject to change as of 2015 June.

1. The Berkeley Out-of-Order Machine

BOOM is a synthesizable, parameterized, superscalar out-of-order RISC-V core designed to serve as the prototypical baseline processor for future micro-architectural studies of out-of-order processors. Our goal is to provide a readable, open-source implementation for use in education, research, and industry.

BOOM is written in roughly 9,000 lines of the hardware construction language *Chisel*. We leveraged Berkeley's open-source *Rocket-chip* SoC generator, allowing us to quickly bring up an entire multi-core processor system (including caches and uncore) by replacing the in-order *Rocket* core with an out-of-order *BOOM* core. *BOOM* supports atomics, IEEE 754-2008 floating-point, and page-based virtual memory. We have demonstrated *BOOM* running Linux, SPEC CINT2006, and CoreMark.

BOOM, configured similarly to an *ARM Cortex-A9*, achieves 3.91 CoreMarks/MHz with a core size of 0.47 mm^2 in TSMC 45 nm excluding caches (and 1.1 mm^2 with 32 kB L1 caches). The in-order *Rocket* core has been successfully demonstrated to reach over 1.5 GHz in IBM 45 nm SOI, with the SRAM access being the critical path. As *BOOM* instantiates the same caches as *Rocket*, *BOOM* should be similarly constrained to 1.5 GHz. So far we have not found it necessary to deeply pipeline *BOOM* to keep the logic faster than the SRAM access. With modest resource sizes matching the synthesizable *MIPS32 74K*, the the worst case path for *BOOM*'s logic is ∼2.2 GHz in TSMC 45 nm (∼30 FO4).

2. Leveraging New Infrastructure

The feasibility of *BOOM* is in large part due to the available infrastructure that has been developed in parallel at Berkeley.

BOOM implements the open-source RISC-V ISA, which was designed from the ground-up to enable VLSI-driven computer architecture research. It is clean, realistic, and highly extensible. Available software includes the *GCC* and *LLVM* compilers and a port of the Linux operating system.[6]

BOOM is written in *Chisel*, an open-source hardware construction language developed to enable advanced hardware design using highly parameterized generators. *Chisel* allows designers to utilize concepts such as object orientation, functional programming, parameterized types, and type inference. From a single *Chisel* source, *Chisel* can generate a cycle-accurate C++ simulator, Verilog targeting FPGA designs, and Verilog targeting ASIC tool-flows.[2]

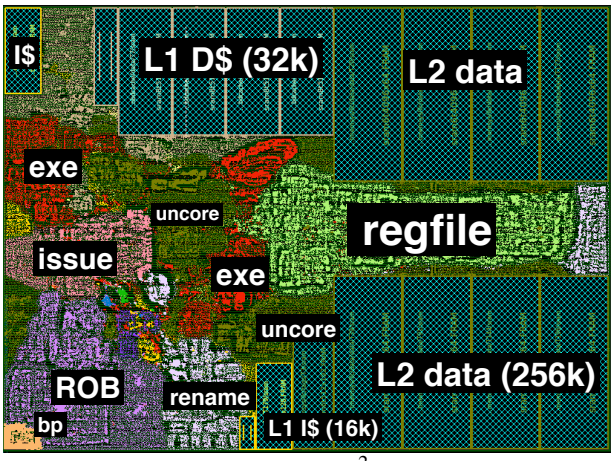

Figure 1: 2-wide *BOOM*, 1.7 mm^2 total in TSMC 45 nm

UC Berkeley also provides the open-source *Rocket-chip* SoC generator, which has been successfully taped out seven times in two different, modern technologies.[6, 10] *BOOM* makes significant use of *Rocket-chip* as a library – the caches, the uncore, and functional units all derive from *Rocket*. In total, over 11,500 lines of code is instantiated by *BOOM*.

3. Methodology: What We Plan to Do

The typical methodology for single-core studies, as gathered from an informal sampling of *ISCA 2014* papers, is to use CPU2006 coupled with a *SimPoints*[12]-inspired methodology to choose the most representative section of the *reference* input set. Each sampling point is typically run for around *10-100 million* instructions of detailed software-based simulation.

The average CPU2006 benchmark is roughly *2.2 trillion* instructions, with many of the benchmarks exhibiting multiple phases of execution.[9] While completely untenable for software simulators, FPGA-based simulators can bring runtimes to within reason – a 50 MHz FPGA simulation can take over 12 hours for a single benchmark. Moreover, we hope to utilize an FPGA cluster to run all the SPEC workloads in parallel.

4. Comparison to Commercial Designs

Table 1 shows preliminary results of *BOOM* and *Rocket* for the CoreMark EEMBC benchmark (we use CoreMark because ARM does not offer SPEC results for the A9 and A15 cores). Our aim is to be competitive in both performance and area against low-power, embedded out-of-order cores.

Table 1: CoreMark results.

Processor	Core Area (core+L1s)	CoreMark/ MHz/Core	Freq (MHz)	CoreMark/ Core	IPC
Intel Xeon E5 2687W (Sandy)†	≈18 mm²@32nm	**7.36**	3,400	25,007	-
Intel Xeon E5 2667 (Ivy)*	≈12 mm²@22nm	**5.60**	3,300	18,474	1.96
ARM Cortex-A15*	2.8 mm²@28nm	**4.72**	2,116	9,977	1.50
RV64 BOOM four-wide*	1.4 mm²@45nm	**4.70**	1,500	7,050	1.50
RV64 BOOM two-wide*	1.1 mm²@45nm	**3.91**	1,500	5,865	1.25
ARM Cortex-A9 (Kayla Tegra 3)*	≈2.5 mm²@40nm	**3.71**	1,400	5,189	1.19
MIPS 74K‡	2.5 mm²@65nm	**2.50**	1,600	4,000	-
RV64 Rocket*	0.5 mm²@45nm	**2.32**	1,500	3,480	0.76
ARM Cortex-A5‡	0.5 mm²@40nm	**2.13**	1,000	2,125	-

Results collected from *the authors (using gcc51 -O3 and perf), †[3], or ‡ [1]. The Intel core areas include the L1 and L2 caches.

Table 2: A sample of academic out-of-order processors.

	IVM[13]	SCOORE[7]	FabScalar[8, 11]	Sharing[14]	BOOM
fully synthesizable	√		√	√	√
FPGA		√	√		√
parameterized			√		√
floating point		√			√
atomic support					√
L1 cache	√	√	√	√	√
L2 cache			√	√	√
virtual memory					√
boots Linux					√
multi-core			√		√
ISA	Alpha (sub-set)	SPARCv8	PISA (sub-set)†	?	RISC-V
lines of code	?	?	65,000†	?	9,000 + 11,500

†Information was gathered from publicly available code at [4].

5. Related Work

There have been many academic efforts to implement out-of-order cores. The Illinois Verilog Model (IVM) is a 4-issue, out-of-order core designed to study transient faults.[13] The Santa Cruz Out-of-Order RISC Engine (SCOORE) was designed to efficiently target both ASIC and FPGA generation. However, SCOORE lacks a synthesizable fetch unit.

FabScalar is a tool for composing synthesizable out-of-order cores. It searches through a library of parameterized components of varying width and depth, guided by performance constraints given by the designer. FabScalar has been demonstrated on an FPGA,[8] however, as FabScalar did not implement caches, all memory operations were treated as cache hits. Later work incorporated the OpenSPARC T2 caches in a tape-out of FabScalar.[11]

The Sharing Architecture is composed of a two-wide out-of-order core (or "slice") that can be combined with other slices to form a single, larger out-of-order core. By implementing a slice in RTL, they were able to accurately demonstrate the area costs associated with reconfigurable, virtual cores.[14]

6. Lessons Learned

Single-board FPGAs have gotten more capable of handling mobile processor designs. *Chisel* provides a back-end mechanism to generate memories optimized for FPGAs, but requires no changes to the processor's source code. While some coding patterns map poorly to FPGAs (e.g., large variable shifters), generally techniques that map well to ASICs also map well to FPGAs.

Re-use is critical. Some of the most difficult parts of building a processor – for example the cache coherency system, the privileged ISA support, and the FPGA and ASIC flows – came to *BOOM* "for free" via the *Rocket-chip* SoC generator. And as the *Rocket-chip* SoC evolves, *BOOM* inherits the new improvements.

Benchmarks are harder to use than they should be. Benchmarks can be difficult to work with and exhibit poor performance portability across different processors, address modes, and ISAs. Many benchmarks (like CoreMark) are written to target 32-bit addresses, which can cause poor code generation for 64-bit processors. We built a histogram generator into the RISC-V ISA simulator to help direct us to potential problem areas. However, additional compiler optimizations are needed to improve 64-bit RISC-V code generation.

We were also surprised to find that SPECINT contains significant floating point code – an integer core may spend over half its time executing software FP routines. As academic SPEC results are typically reported in terms of CPI, we must be careful to not optimize for the wrong cases. We added hardware FP support to *BOOM* to address this issue.

Finally, we found SPEC difficult to work with, especially in non-native environments. We created the *Speckle* wrapper to help facilitate cross-compiling and generating portable directories to run on simulators and FPGAs.[5]

Diagnosing bugs that occur *billions* of cycles into a program is hard. We mostly rely on a *Chisel*-generated C++ simulator for debugging, but at roughly 30 KIPS, 1 billion cycles takes 8 hours. A torture-test generator (and a suite of small test codes) is invaluable.

Acknowledgments

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. 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.

References

[1] "ARM Outmuscles Atom on Benchmark," http://parisbocek.typepad.com/blog/2011/04/arm-outmuscles-atom-on-benchmark-1.html/.

[2] "Chisel: Constructing Hardware in a Scala Embedded Language," https://chisel.eecs.berkeley.edu/.

[3] "Coremark EEMBC Benchmark," https://www.eembc.org/coremark/.

[4] "FabScalar pre-release tools," http://people.engr.ncsu.edu/ericro/research/fabscalar/pre-release.htm.

[5] "Speckle: A wrapper for the SPEC CPU2006 benchmark suite." https://github.com/ccelio/Speckle.

[6] "The RISC-V Instruction Set Architecture," http://riscv.org/.

[7] W. Ashmawi *et al.*, "Implementation of a power efficient high performance fpu for scoore," in *Workshop on Architectural Research Prototyping (WARP), held in conjunction with ISCA-35*, 2008.

[8] B. H. Dwiel *et al.*, "Fpga modeling of diverse superscalar processors," in *Performance Analysis of Systems and Software (ISPASS), 2012 IEEE International Symposium on*. IEEE, 2012, pp. 188–199.

[9] A. Jaleel, "Memory characterization of workloads using instrumentation-driven simulation," *Web Copy: http://www. glue. umd. edu/ajaleel/workload*, 2010.

[10] Y. Lee *et al.*, "A 45nm 1.3 ghz 16.7 double-precision gflops/w risc-v processor with vector accelerators," in *European Solid State Circuits Conference (ESSCIRC), ESSCIRC 2014-40th*. IEEE, 2014, pp. 199–202.

[11] E. Rotenberg *et al.*, "Rationale for a 3d heterogeneous multi-core processor," in *Computer Design (ICCD), 2013 IEEE 31st International Conference on*, Oct 2013, pp. 154–168.

[12] T. Sherwood *et al.*, "Automatically characterizing large scale program behavior," *ACM SIGARCH Computer Architecture News*, vol. 30, no. 5, pp. 45–57, 2002.

[13] N. J. Wang *et al.*, "Characterizing the effects of transient faults on a high-performance processor pipeline," in *Dependable Systems and Networks, 2004 International Conference on*. IEEE, 2004, pp. 61–70.

[14] Y. Zhou and D. Wentzlaff, "The sharing architecture: sub-core configurability for iaas clouds," in *Proceedings of the 19th international conference on Architectural support for programming languages and operating systems*. ACM, 2014, pp. 559–574.