Your Agile Open Source HW Stinks (Because It Is Not a System)

Invited Talk

Michael Bedford Taylor
University of Washington
prof.taylor@gmail.com

Abstract

Exciting times in hardware design are upon us. Spearheaded by the RISC-V ISA, and the recent DARPA POSH/IDEA program which focuses on both open source IP and open source CAD, a large number of open source HW projects are underway. Academics are increasingly releasing their code online. Many agile open source HW projects envision a hypothetical user that may not actually exist. To acquire real users, we must be pragmatic about what kinds of systems our HW will go into and focus on the roadblocks unique to those systems.

Keywords Open Source HW

ACM Reference Format:
Michael Bedford Taylor. 2020. Your Agile Open Source HW Stinks (Because It Is Not a System). In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD '20), November 2–5, 2020, Virtual Event, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3400302.3415791

1 Introduction

Through the efforts of leaders like Krste Asanovic and Andreas Olofsson, and lesser known but equally valorous ones across the planet, a foundation for open source HW is materializing beneath our feet.

Spearheaded by the RISC-V ISA, and the recent DARPA POSH and IDEA programs which focus on both open source IP and open source CAD, a large number of open source HW projects are underway. Academics are increasingly releasing their code online. The potential impact seems limitless; we can imagine that the availability of these tools and designs will allow designers to rapidly create systems with minimal NRE, heralding a new golden age of semiconductor innovation [15].

In the next sections we will highlight some recent developments in the open source HW ecosystem, then talk about some opportunities and eco-system problems, and then conclude with some observations.

1.1 RISC-V.

First, we have the RISC-V ISA [28], which provides a license-free lingua franca for general purpose computation, but also for customization features that provide a foundation for specialization.

RISC-V Cores. Moreover, we have Linux-capable implementations of RISC-V processors like BlackParrot [22], ETH Zurich Ariane [29], and Berkeley Rocket [11], as well as GP-GPU-style compute throughput fabrics like HammerBlade Manycore [8] (descended from Celerity [9, 14, 23]), microcontrollers like Western Digital's SweRV [3], and scalable multicore server processors like the RISC-V incarnation of Princeton OpenPiton [12].

RISC-V unlocking research and education. With these newly available cores, users now have the ability to experiment, use, and share processors without seeking licenses from ARM or Intel. Licensing problems were not solely an issue of money. These licenses in many cases were impossible to obtain, or at least onerous (mandating special rooms that served as data diodes), and even if they could be obtained, US export control restrictions on processor IP prevent a broad class of graduate students from using them. For sure, all of these cores will unlock many research usages that were otherwise impossible. In the near future, both CAD and architecture can work on real models and move away from synthetic benchmarks or simulations,

ICCAD '20, November 2–5, 2020, Virtual Event, USA
© 2020 Association for Computing Machinery.
ACM ISBN 978-1-4503-8026-3/20/11...$15.00
https://doi.org/10.1145/3400302.3415791

respectively. Already, both commercial CAD companies and universities are making extensive use of these infrastructures, largely on commercial CAD tools. Classes are increasingly making use of RISC-V processor designs, and indeed Patterson and Hennessy have new versions of their computer architecture textbooks that employ RISC-V.

Challenges of Commercial Use. The picture for use of these cores in commercial chips is currently murky. Certainly, ARM has better support for their designs than any of the open source efforts. Indeed, SiFive, which is founded with many of the original RISC-V developers, charges money for the use of their closed source derivatives of the open source Rocket core. Moreover, the total budget of typical advanced chip designs greatly exceeds the price of an ARM core, so the expected value calculation of using an untrusted, open source, RISC-V core may indeed be negative (e.g., 5% chance of failure $\times$ 30 million + \$1M saved = -\$0.5M.) (Perhaps the logical step would be to tape out with both a RISC-V and ARM core, and then use the ARM as a backup, and phase it out in future generations...)

Anecdotally, ARM offers large discounts to startups on their cores, eliminating incentives for startups to adopt RISC-V. From a startup perspective, taking a risk to save money would not make sense since they would prefer to focus their resources on their value add – e.g., AI core or whatever.

Short Term: Trusting Your Open Source RISC-V Core by Hiding It. A potentially bright spot for adoption of open source RISC-V cores lies where the RISC-V core is hidden; i.e. the end user does not program the RISC-V core, but rather uses a *system* that employs the core. For example, NVidia announced that they use a RISC-V core for a control processor on their GPUs [1]. When the RISC-V core is a hidden component of the product, then there is a much greater tolerance for bugs. Workarounds can be deployed in the compiler, library or firmware; and security attack surface is minimized by restricting access to the code that is running. For similar reasons, open source RISC-V cores are especially suited for research prototypes, where workarounds are easy.

Short Term: Trusting That Open Source RISC-V Core by Using It Only The Way It Was Tested. A similar corollary is, you can expect a core to behave the way it behaved in simulation. So if a user has a very static use case; i.e. booting the GPU, then it can rely on that to work (mostly) the way it did in simulation, given sufficiently accurate modeling of the environment.

Long Term: Massive, massive open source regression suites for RISC-V cores. If these open source cores are constantly evolving in an "agile manner" – how will we ever be able to move beyond the "hidden core" phase? By building a massive open source regression suite for RISC-V, the kind that has never been seen before. Effectively, this is the shared task that all of the open source RISC-V core implementers will have to contribute to in order to be mutually successful, even as they compete.

1.2 Open Source IP.

The DARPA POSH program has funded a variety of open source IP blocks, including the BlackParrot RISC-V processor, and two open source FPGA efforts, one from Princeton [20] and one from Utah [25]. The SystemVerilog-based BlackParrot processor is intended to be the first community-maintained RISC-V core, i.e., the "Linux of RISC-V". The FPGA fabrics, in addition to providing able research substrates for future FPGA and CAD research, potentially provide the opportunity to add flexibility to open source tapeouts, perhaps so they can more easily find their niches after fabrication.

There is also a POSH-funded 10-100G Ethernet Mac core from LeWiz Communications [5], which could potentially be useful but in many cases would be best paired with an open-source TCP/IP offload engine, for which ASIC versions do not yet appear to exist[1].

The PyMTL3 [16] environment seeks to enhance productivity in HW design, offering seamless Python-driven RTL development and simulation, and advanced verification features.

The BaseJump STL Library for SystemVerilog [26] provides a set of uniform interfaces for all of the commonly designed hardware primitives, elevating the level of abstraction for hardware design. It is used as a foundation for both BlackParrot and the HammerBlade Manycore systems. Such standardization across the open source HW community can reduce bug densities and increase testing leverage, enabling unparalleled levels of productivity.

[1]High performance FPGA versions, many of them HLS-based, do exist, however, e.g. [24].

Tech	250nm	180nm	130nm	90nm	65nm	40nm	28nm	16nm
Mask cost ($)	65K	105K	290K	560K	700K	1.25M	2.25M	5.70M
Cost per wafer ($)	720	790	2,950	3,200	3,300	4,850	7,600	11,100
Wafer diameter (mm)	200	200	300	300	300	300	300	300

Table 1. Cost of using old nodes (from 2017). Data from [19].

1.3 Open Source CAD Tools.

Another bright spot in recent open source developments are the tools. Much as the GNU GCC compiler paved the way for GNU/Linux by providing a uniform C interface to diverse hardware, having open source RTL-to-GDS tools could pave the way for a healthy open source ecosystem by providing widespread compatibility (something missing across current-day commercial CAD tools!) across many process nodes.

Simulation. The Verilator SystemVerilog simulator has been proven very stable and widely compatible with industry-standard VCS, with the exception of testbench support, which generally must be rewritten with considerable labor costs. Verilator enables cost-effective scaleout of many simulations across the cloud (perfect for executing the "long term RISC-V regression suite"), but does not replace the signoff functionality of commercial tools (i.e. simulation with parasitics.)

Synthesis. The Yosys synthesis tool has found wide use in academia and the open source community, although it does not support SystemVerilog and its output PPA may fall significantly short. Its key benefit is its open sourceness, which enables it to be modified for interesting new flows that cannot be easily modeled with out of the box commerical tools. Yosys also has support for formal verification including SMT solvers and SAT solvers. SystemVerilog synthesis remains a key weakness that hopefully will be addressed in the near term, as many parsers are under development and being evaluated by the Symbiflow sv-tests framework [7].

Automatic Place and Route. The DARPA IDEA program has funded OpenROAD [10], a powerful no-human-in-the-loop place and route flow that targets GlobalFoundries 12nm technology. Several external users have mapped it to other technologies, including the UW OpenROAD FreePDK45 flow [13] that targets the eponymous freely available synthetic 45nm PDK, and the OpenLANE flow [6], which targets the real-life Google-sponsored Skywater 130nm flow. Since Automatic Placement and Routing is easily the most expensive CAD flow component, and verification of the output is relatively easy to do using existing methodologies, OpenROAD is really the centerpiece of open source CAD, and strong PPA results by the end of the project would "show the light at the end of the tunnel" and buttress the open source design world such that the other pieces would likely follow very quickly afterwards. PPA-strong OpenROAD will unlock research in ML-driven CAD and allow massive exploration of design spaces via cost-effective cloud-scaleout.

Parasitic Extraction. Missing from these flows is open source parasitic extraction, simulation with parasitics, and formal equivalency checking. It's a little crazy to envision a tapeout without at least some of these.

2 Opportunities and Eco-System Challenges.

With the current slow motion, live-action enactment of the end of CMOS scaling and Moore's Law, the research community has increasingly looked towards designing ASIC-based accelerators that exploit specialization in order to surpass power- and energy- limited general-purpose devices like CPUs and GPUs. ASIC accelerators are able to attain order-of-magnitude improvements in energy-efficiency (W per op/s) and cost-performance ($ per op/s) over general-purpose substrates, which helps in optimizing total cost of ownership (TCO) in the datacenter [18, 19, 21, 27] and also in attaining new features for mobile devices as shown in this brilliant analysis [2] of exponentially growing numbers of accelerator blocks on Apple iPhone chips.

Acceptance of a future of accelerators brings some promise that perhaps open source can help drive the design of these accelerators by reducing startup costs. In particular, accelerators could leverage older nodes, reducing mask and IP costs to a point where open

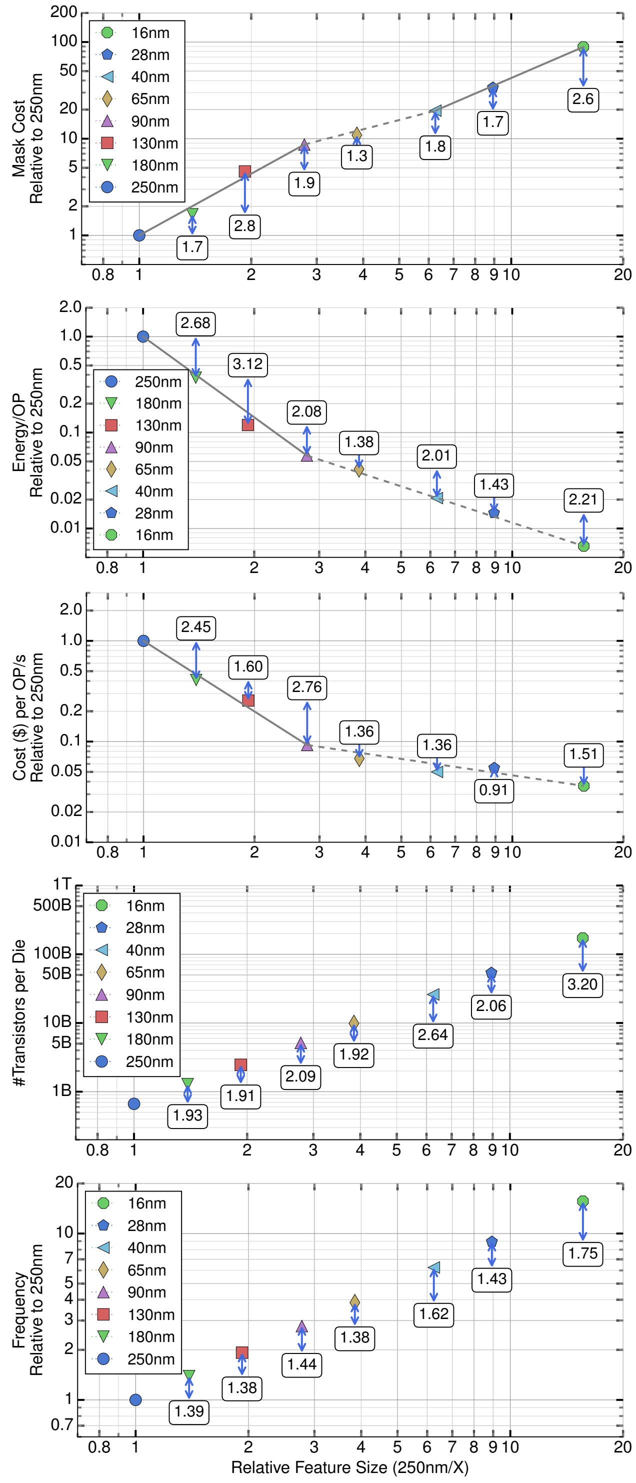

Figure 1. Node Technology trade-offs, normalized to 250nm. #'s show multiplicative benefits as node advance. Data from [19].

source IP and CAD tools can attain sufficiently high savings relative to other costs that they start to make sense. Because of the end of Dennard Scaling, post-90nm nodes have a smaller marginal benefit compared to older nodes, as shown in the change of slope in the PPA metrics graphed in Figure 1, although it is still exponential. So the PPA hit of being in the older node is potentially amortizable with the specialization that the accelerator will bring.

Does Optimizing Mask Costs Make Sense? In Table 1, we show 2017 pricing for full-reticle manufacturing of silicon chips. 2020 pricing could be approximated by shifting over the prices to the left by one slot. Although using an older node to save mask costs seems desirable, it must always be put in context of the cost of engineering. In Silicon Valley, a ten person engineering team working for a year might easily cost the same as today's 16nm fab costs. So if shooting for an older node means that they spend another year optimizing to make up for it, it doesn't make sense economically. However in other parts of the world, labor could be significantly less, and older nodes could make more sense. Or alternatively, if open source tools and IP provide sufficiently high leverage that one person can do a tapeout on their own, then older nodes would also become very compelling, even in Silicon Valley.

The Physical IP Problem. In Table 2, we find the price of IP blocks. Most digital chips will require at least the DRAM Controller and DRAM PHY, so the implied \$275K-\$875K is not insignificant. Open source DRAM controllers can be realized in pure Verilog; especially since the DFI standard defines the interface to the DRAM PHY. The DRAM PHY is much more problematic, since they typically require analog circuits that prevent free exchange because of NDAs related to design rules. Interestingly, LPDDR can be implemented without analog circuit implementations and uses off-the-shelf LVCMOS I/Os. A prototype LPDDR DRAM controller and PHY can be found as part of U. Washington's BaseJump STL[2]. LPDDR at 400 mbps per pin is about 5× off in bandwidth from more modern DDR; but the latency is close, maybe 1.3× off; so for processor cores (like the RISC-V processors described earlier), which are latency-bound, it is reasonable, for accelerators, which are often bandwidth-bound it is less so. Typical server accelerator chips will need PCI-E, which

[2]https://github.com/bespoke-silicon-group/basejump_stl/tree/master/bsg_dmc

Tech Node (nm)	250	180	130	90	65	40	28	16
DRAM Ctlr	NA	NA	125	125	125	125	125	125
DRAM PHY	NA	NA	150	165	175	280	390	750
PCI-E Ctlr	NA	NA	90	90	125	125	125	125
PCI-E PHY	NA	NA	160	180	325	375	510	775
PLL	15	15	15	20	30	50	35	50
LVDS IO	7.5	7.5	0	150	90	36	40	200
Standard Cells, SRAM	0	0	0	0	0	100	100	100

Table 2. IP Licensing Costs increase with advancing Technology Nodes. Commonly used IP licensing costs across tech nodes, in late 2016, thousands of USD. Costs generally rise with node, but there are some irregularities. Data from [19].

has a similarly scary price tag. Like the DRAM PHY, the PCI-E PHYs required advanced analog design and are not likely to appear in open source form.

Chiplets to the rescue? The DARPA CHIPS program has envisioned the possibility of libraries of chiplets that host expensive IPs, and that can allow rapid composition. Intel has standardized their AIB standard for the purpose of communicating between chiplets [4]. For example, we could imagine chiplets for optical communication, PCI-E or DRAM PHYs. There are two sticking points; first the PHY itself for chiplets is rapidly becoming more advanced itself and soon will surpass PCI-E, making it once again necessary to have some source of this PHY block, and second, the fabrication technology for the interposers is often restricted to the foundry's most lucrative customers. Hopefully these two issues will be resolved and the open source PHY problem will be solved, perhaps by foundries providing this as a standard IP.

3 Thinking about the System.

Hopefully by this point in the paper, we have convinced you that Open Source HW is an exciting and vibrant area. But it is also clear that there is a huge amount of uncertainty. Here are some key questions:

Who will use your open source HW? Given this constrained world, who will use your open source HW design? Are you abstractly assuming a kind of user that doesn't actually exist? Currently, the best clarity I have is on this issue is: *People whose systems have the flexibility to work around the issues that it will have. Students in classes, researchers, and chip companies that can "hide your hardware" or plan on using it "exactly as in simulation."*

Beyond the RTL, what is the system around your Open Source HW? Thinking about who will use it, what kind of system will they need built around it? Students will be building homework assignments and need clear docs and infrastructure. Researchers will be running benchmarks and need good scripts and preported benchmarks. Chip companies will be running software on top of your RTL and will want a software SDK. For an accelerator, people will want to tools for performance analysis and to have libraries that do the hard work for them. They will want programming languages and compilers to program an ML accelerator.

We must build out these components and make them super easy to use. Recent examples include the FireSim system [17] by Berkeley which is a system built on top of their Berkeley Rocket SoC for academic architectural datacenter simulations. The HammerBlade RISC-V ML/Graphs system reaches toward the end user by implementing the CUDA-lite language, and by supporting PyTorch. The BlackParrot [22] system has a ready-built SDK, and accelerator integration guide that allows for fast accelerator/SoC composition.

How will you discover what systems might be built with your agile HW? For this, you must talk to lots of different people and be .. dare I say? .. *agile* in recognizing realistic use cases. But, even then, in the end, after all of that searching, you may find it is easier to become the user you are looking for!

This material is based on research sponsored by Air Force Research Laboratory (AFRL) and Defense Advanced Research Projects Agency (DARPA) under agreement number FA8650-18-2-7856 and FA8650-18-2-7852. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of Air Force Research Laboratory (AFRL) and Defense Advanced Research Projects Agency (DARPA) or the U.S. Government.

References

[1] RISC-V in NVidia. *Sixth RISC-V workshop*, 2017. URL https://riscv.org/wp-content/uploads/2017/05/Tue1345pm-NVIDIA-Sijstermans.pdf.

[2] *Apple iPhone Accelerator Die Photo Analysis*, 2019. URL http://vlsiarch.eecs.harvard.edu/research/accelerators/die-photo-analysis/.

[3] *Western Digital Open Source SWERV core*, 2019. URL https://github.com/chipsalliance/Cores-SweRV.

[4] *AIB PHY Repo*, 2020. URL https://github.com/chipsalliance/aib-phy-hardware.

[5] LeWiz Ethernet MAC Core. *website*, 2020. URL https://github.com/taylor-bsg/LMAC_CORE2.

[6] *OpenLANE RTL to GDSII Flow*, 2020. URL https://github.com/efabless/openlane.

[7] Symbiflow sv-tests. *website*, 2020. URL https://symbiflow.github.io/sv-tests/.

[8] Michael Taylor et al. The HammerBlade RISC-V Manycore: A programmable, scalable RISC-V fabric. *FOSDEM*, 2020. URL https://fosdem.org/2020/schedule/event/riscv_hammerblade/.

[9] A. Rovinski et al. A 1.4 GHz 695 Giga Risc-V Inst/s 496-Core Manycore Processor With Mesh On-Chip Network and an All-Digital Synthesized PLL in 16nm CMOS. In *2019 Symposium on VLSI Circuits*, pages C30–C31, 2019.

[10] T. Ajayi, V. A. Chhabria, M. Fogaça, S. Hashemi, A. Hosny, A. B. Kahng, M. Kim, J. Lee, U. Mallappa, M. Neseem, G. Pradipta, S. Reda, M. Saligane, S. S. Sapatnekar, C. Sechen, M. Shalan, W. Swartz, L. Wang, Z. Wang, M. Woo, and B. Xu. Toward an open-source digital flow: First learnings from the openroad project. In *Proceedings of the 56th Annual Design Automation Conference 2019*, DAC '19, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450367257. doi: 10.1145/3316781.3326334. URL https://doi.org/10.1145/3316781.3326334.

[11] K. Asanović, R. Avizienis, J. Bachrach, S. Beamer, D. Biancolin, C. Celio, H. Cook, D. Dabbelt, J. Hauser, A. Izraelevitz, S. Karandikar, B. Keller, D. Kim, J. Koenig, Y. Lee, E. Love, M. Maas, A. Magyar, H. Mao, M. Moreto, A. Ou, D. A. Patterson, B. Richards, C. Schmidt, S. Twigg, H. Vo, and A. Waterman. The Rocket Chip Generator. Technical Report UCB/EECS-2016-17, EECS Department, University of California, Berkeley, Apr 2016. URL http://www2.eecs.berkeley.edu/Pubs/TechRpts/2016/EECS-2016-17.html.

[12] J. Balkind, T.-J. Chang, P. Jackson, G. Tziantzoulis, A. Li, F. Gao, A. Lavrov, G. Chirkov, J. Tu, M. Shahrad, and D. Wentzlaff. OpenPiton at 5: A Nexus for Open and Agile Hardware Design. *IEEE Micro*, 40:22–31, 2020.

[13] S. Davidson and M. B. Taylor. *University of Washington OpenRoad FreePDK 45*, 2019. URL https://github.com/bsg-idea/uw_openroad_free45.

[14] S. Davidson, S. Xie, C. Torng, K. Al-Hawai, A. Rovinski, T. Ajayi, L. Vega, C. Zhao, R. Zhao, S. Dai, A. Amarnath, B. Veluri, P. Gao, A. Rao, G. Liu, R. K. Gupta, Z. Zhang, R. Dreslinski, C. Batten, and M. B. Taylor. The Celerity Open-Source 511-Core RISC-V Tiered Accelerator Fabric: Fast Architectures and Design Methodologies for Fast Chips. *IEEE Micro*, 38(2): 30–41, 2018.

[15] J. L. Hennessy and D. A. Patterson. A New Golden Age for Computer Architecture. *Commun. ACM*, 62(2):48–60, Jan. 2019. ISSN 0001-0782. doi: 10.1145/3282307. URL https://doi.org/10.1145/3282307.

[16] S. Jiang, P. Pan, Y. Ou, and C. Batten. PyMTL3: A Python Framework for Open-Source Hardware Modeling, Generation, Simulation, and Verification. *IEEE Micro*, 40(4):58–66, 2020.

[17] S. Karandikar, H. Mao, D. Kim, D. Biancolin, A. Amid, D. Lee, N. Pemberton, E. Amaro, C. Schmidt, A. Chopra, Q. Huang, K. Kovacs, B. Nikolic, R. Katz, J. Bachrach, and K. Asanović. Firesim: FPGA-Accelerated Cycle-Exact Scale-out System Simulation in the Public Cloud. In *Proceedings of the 45th Annual International Symposium on Computer Architecture*, ISCA '18, page 29–42. IEEE Press, 2018. ISBN 9781538659847. doi: 10.1109/ISCA.2018.00014. URL https://doi.org/10.1109/ISCA.2018.00014.

[18] M. Khazraee, L. Vega, I. Magaki, and M. Taylor. Specializing a Planet's Computation: ASIC Clouds. *IEEE Micro*, May 2017.

[19] M. Khazraee, L. Zhang, L. Vega, and M. B. Taylor. Moonwalk: NRE Optimization in ASIC Clouds. In *Proceedings of the Twenty-Second International Conference on Architectural Support for Programming Languages and Operating Systems*, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450344654. doi: 10.1145/3037697.3037749. URL https://doi.org/10.1145/3037697.3037749.

[20] A. Li and D. Wentzlaff. PRGA: An Open-source Framework for Building and Using Custom FPGAs. *2019 Workshop on Open Source Design Automation (OSDA '19)*, 2019.

[21] I. Magaki, M. Khazraee, L. Vega, and M. Taylor. ASIC Clouds: Specializing the Datacenter. In *International Symposium on Computer Architecture (ISCA)*, 2016.

[22] D. Petrisko, F. Gilani, M. Wyse, D. C. Jung, S. Davidson, P. Gao, C. Zhao, Z. Azad, S. Canakci, B. Veluri, T. Guarino, A. Joshi, M. Oskin, and M. B. Taylor. BlackParrot: An Agile Open-Source RISC-V Multicore for Accelerator SoCs. *IEEE Micro*, 40 (4):93–102, 2020.

[23] A. Rovinski, C. Zhao, K. Al-Hawaj, P. Gao, S. Xie, C. Torng, S. Davidson, A. Amarnath, L. Vega, B. Veluri, A. Rao, T. Ajayi, J. Puscar, S. Dai, R. Zhao, D. Richmond, Z. Zhang, I. Galton, C. Batten, M. B. Taylor, and R. G. Dreslinski. Evaluating Celerity: A 16-nm 695 Giga-RISC-V Instructions/s Manycore Processor With Synthesizable PLL. *IEEE Solid-State Circuits Letters*, 2(12):289–292, 2019.

[24] M. Ruiz, D. Sidler, G. Sutter, G. Alonso, and S. López-Buedo. Limago: an FPGA-based Open-source 100 GbE TCP/IP Stack. In *2019 29th International Conference on Field Programmable Logic and Applications (FPL)*, pages 286–292. IEEE, Sep 2019. doi: 10.1109/FPL.2019.00053.

[25] X. Tang, E. Giacomin, A. Alacchi, B. Chauviere, and P. Gaillardon. Openfpga: An opensource framework enabling rapid prototyping of customizable fpgas. In *2019 29th International Conference on Field Programmable Logic and Applications (FPL)*, pages 367–374, 2019.

[26] M. B. Taylor. BaseJump STL: SystemVerilog needs a Standard Template Library for Hardware Design. In *Design Automation Conference*, June 2018.

[27] M. B. Taylor, L. Vega, M. Khazraee, I. Magaki, S. Davidson, and D. Richmond. ASIC Clouds: Specializing the Datacenter for Planet-Scale Applications. *CACM*, pages 103–109, 2020.

[28] A. Waterman. *Design of the RISC-V Instruction Set Architecture*. PhD thesis, EECS Department, University of California, Berkeley, Jan 2016. URL http://www2.eecs.berkeley.edu/Pubs/TechRpts/2016/EECS-2016-1.html.

[29] F. Zaruba and L. Benini. The Cost of Application-Class Processing: Energy and Performance Analysis of a Linux-Ready 1.7-GHz 64-Bit RISC-V Core in 22-nm FDSOI Technology. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 27 (11):2629–2640, 2019.