OpenPiton at 5: A Nexus For Open And Agile Hardware Design

Jonathan Balkind, Ting-Jung Chang, Paul J. Jackson, Georgios Tziantzioulis, Ang Li, Fei Gao, Alexey Lavrov, Grigory Chirkov, Jinzheng Tu, Mohammad Shahrad, David Wentzlaff
Princeton University

***Abstract*—For five years, OpenPiton has provided hardware designs, build and verification scripts, and other infrastructure to enable efficient, detailed research into manycores and systems-on-chip. It enables open-source hardware development through its open design and support of a plethora of open simulators and CAD tools. OpenPiton was first designed to perform cutting-edge computer architecture research at Princeton University and opening it up to the public has led to thousands of downloads and numerous academic publications spanning many subfields within computing. In this article, we share some of the lessons learned during the development of OpenPiton, provide examples of how OpenPiton has been used to efficiently test novel research ideas, and discuss how OpenPiton has evolved due to its open development and feedback from the open-source community.**

THE OPENPITON PROJECT began over six years ago, with 2020 representing the fifth anniversary of its first open-source release (timeline in **Figure 1**). Thirteen releases and thousands of downloads later, we can draw on our experience to see what worked best, what needs improvement, and what would maximize OpenPiton's utility to the open-source hardware community. This article is both an experience report and a progress update on the status of OpenPiton.

OpenPiton is a framework for designing, simulating, emulating, and building manycore processors and systems-on-chip (SoCs), like those shown in Figure 2. OpenPiton provides a highly configurable and extensible, Linux-capable, manycore processor design. Users receive scripts and tools necessary to write their initial hardware components, test on FPGA, build real chips, and run applications on their designs. As a platform for research, OpenPiton has seen wide adoption in many fields of study within computing thanks to its ready-to-use full system stack, push-button scripting, and support of both open-source and commercial tools. Through community outreach, active support, and regular updates, our team has adapted and improved OpenPiton to support the growing needs and interests of the research community.

The Beginning

OpenPiton started in 2013 as a platform for building a single chip, Piton. Piton was a vehicle for several ideas from our research group which needed a full-system software-hardware approach to evaluate their benefits [2]. Having built a general design exemplifying a realistic, OS-capable, scalable manycore, we saw the opportunity to make it available to others with similar research goals. We believe that our internal need to build a general research platform supporting a breadth of research ideas was a key factor leading to

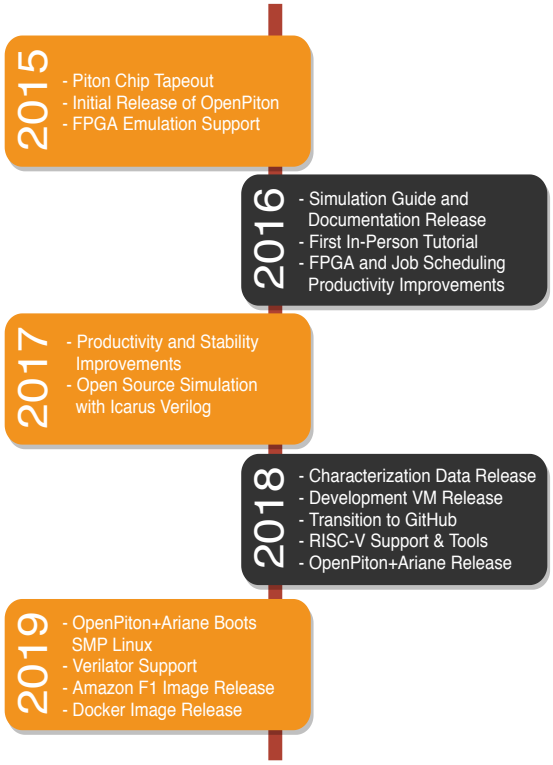

Figure 1. Highlights from OpenPiton's First 5 Years

OpenPiton's broad adoption and we maintain this important focus today.

Using OpenPiton

OpenPiton provides a platform for performing two primary classes of activities: building designs into artifacts (such as simulation models, FPGA bitstreams, and silicon chips), and evaluating artifacts (running code on designs, evaluating the produced artifacts' quality). Figure 3 gives an overview of some uses in each class.

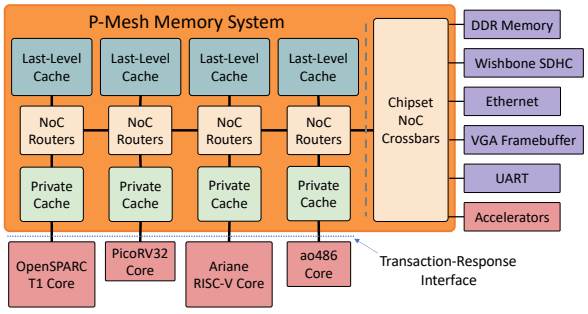

Figure 2. OpenPiton Architecture, adapted [1]

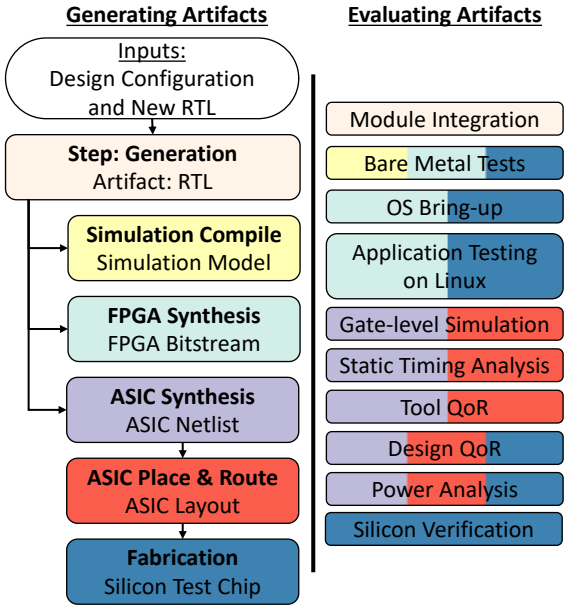

Figure 3. Artifact generation and evaluation using OpenPiton. Coloring of evaluation (right) corresponds with artifacts that may be used (left).

OpenPiton Design Philosophy

Lowering the Barrier to Entry and Enabling Agile Development

Intellectual property (IP) licensing costs are a significant factor in non-recurring engineering (NRE) costs. The growth of active open-source hardware projects paves the way for IP cost reductions by providing high-quality, reusable components. OpenPiton is one of the leading efforts in constructing such an open-source ecosystem targeting the research community.

Push-button

One of the core aspects of the OpenPiton design philosophy which developed over time has been providing a push-button experience.

Lessons: As we have built more designs and users repeatedly asked for assistance, we have come to focus on making common tasks (both building and evaluating artifacts) easy to script around, requiring only a single user-specified command. For example, an OpenPiton simulation model can be compiled with one of six supported simulators using a single command. Likewise, a single invocation of our `protosyn` FPGA synthesis flow can produce an FPGA bitstream of a user-parameterised design for one of many

common Xilinx FPGA development boards or Amazon's EC2 F1 cloud FPGA platform. To evaluate the design, groups of hundreds or thousands of assembly and C tests (or a prebuilt OS image) can then be run on the simulation or FPGA model with a single command.

When used for ASIC creation, our flow enables running all of the steps of synthesis, placement, and routing to generate a final ready-for-manufacture netlist with a single command. Each step can also be run independently and have its results checked, and the steps can be submitted to a job scheduler (SLURM) to run in parallel. This significantly reduces the barrier to entry for new users that can quickly familiarize themselves with the infrastructure, and the different steps of the build and verification process.

Supported Tools and Ease of Porting

To support users with different environments and constraints, we have worked to provide extensive choices of simulators and CAD tools. For hardware simulation, we started only with Synopsys VCS.

Lessons: As users adopted OpenPiton, we learned that they had different resources available and had to broaden our simulator support. Accordingly, we added Cadence NCSim, Mentor Questa/ModelSim, and open-source simulators Icarus Verilog and Verilator. With its cross-platform codebase, OpenPiton has become an attractive test case for EDA developers, which led to Riviera-PRO's developers contributing support. For ASIC, we extended our commercial tool flow to support open-source EDA tools, such as Yosys, OpenSTA, and OpenTimer. Supporting these open-source tools lowers the barrier of entry for at-scale, real-system manycore research.

OpenPiton's original synthesis and backend physical design scripts were based on IBM's 32nm silicon-on-insulator (SOI) process, as used for the Piton ASIC. As we moved to support our and others' use of different commercial or open-source processes, we realised that we needed a flow which would support such a change.

Lessons: The philosophy that emerged is to enable agile porting across different technology nodes. To achieve this, we separated the scripts into two different groups, design-specific and process-specific, so that users can quickly identify and adapt to their desired process technology. With this change, we support many new process design kits, including GLOBALFOUNDRIES 12nm Fin field-effect transistor (FinFET) process, Synopsys 32/28nm educational process, open-source FreePDK45, and FreePDK15. Our users have adopted a variety of other processes.

We similarly support easy porting to new Xilinx FPGAs. OpenPiton designs have run on more than ten different Xilinx boards. To target a new FPGA, the user only needs to create a handful of new files and part-specific IP configurations.

Open Source Support and Collaboration

In developing OpenPiton, we have leveraged a variety of open IP to improve the platform and reduce design effort (and thus NRE). We have also had the privilege to work with other developers and researchers to create new designs and prototypes which integrate open tools and IP with OpenPiton.

OpenPiton+Ariane

OpenPiton+Ariane is the world's first open-source, SMP Linux-booting RISC-V manycore. The platform is the fruit of our collaboration with ETH Zürich's PULP team where we integrated the Ariane RISC-V core into OpenPiton [1].

PyOCN Network Replacement

PyOCN [3] is an effort to provide "a unified framework that vertically integrates multiple research methodologies to enable productively exploring the on-chip network design space". The PyOCN developers replaced OpenPiton's network-on-chip with PyOCN-generated crossbars and meshes, with the goal of a completely configurable NoC topology in OpenPiton. This will enable research into optimized network topologies for full-stack systems.

EDA

Though OpenPiton first targeted the Computer Architecture community, it has provided much utility to the EDA community. Its modularized, scalable, configurable, and heterogeneous nature make it an excellent source of design benchmarks for CAD tool development. Additionally, OpenPiton's synthesis and backend scripts straightforwardly generate properly formatted input files.

We have worked closely with many open-source CAD tool developers. On one hand, Open-

Piton provides a rich set of designs for developers stress-testing their tools. For example, a number of OpenPiton's modules were used to demonstrate LSOracle, an automated mixed logic synthesis framework [4]. On the other hand, we have integrated open tools, such as Yosys, OpenTimer [5], Icarus Verilog, and OpenROAD [6] into our originally commercial tool based ASIC flow. The purely open-source flow based on open cell libraries further extends the usability of the framework.

Verification
Verification researchers have also adopted OpenPiton as a benchmark. For example, the L1.5 cache has been used as a driving example for Instruction Level Abstraction (ILA) [7] based verification, a formal abstraction for processors and accelerators.

FuseSoC
To move towards a more structured dependency description format, we are replacing industry standard file lists (flists) with the FuseSoC CAPI description format. Moving away from flists enables us to move to a more modular and configurable framework. The CAPI description format clearly describes the dependencies between different modules and naturally creates a design hierarchy, which assists the overall development effort when importing external IP and exporting our IP as usable IP for others. To tackle the new challenges we closely collaborated with the FuseSoC developers, suggesting features, reporting bugs, and contributing support for tools such as VCS.

Peripherals
Peripherals are critical to users interested in system-level and especially full-OS functionality. The closed/proprietary peripherals we started with restricted us in a variety of ways. Moving to open alternatives for SD/SDHC/SDXC, DDR, and VGA controllers (among others) brought higher quality, maintainability, and modifiability.

Industry
We work with a number of industrial partners on their use and enhancement of OpenPiton. The platform has proved particularly valuable for EDA tool development and validation. Our partners have tested manycore design scalability to as many as 2500 OpenSPARC T1 cores and have run emulations of 128 or more Ariane cores. With some EDA users routinely running OpenPiton designs through their tools, our academic and industrial users alike can use OpenPiton with confidence that the code is well validated and works with their tools of choice. With the addition of new heterogeneous elements into OpenPiton, our EDA partners can target the design without concern for generality, as it is becoming very reflective of a modern, industrial-grade SoC.

Forward Thinking and Compatibility

In developing OpenPiton, we have come to learn that it is crucial to build a system which balances efficiency with configurability and extensibility. Today we aim to be forward thinking and design for forward compatibility.

P-Mesh Packet Format
Our original NoC packet format was dense and provided little room for growth. As the design became more complex, this brittle approach limited not just our flexibility but also our creativity.

Lessons: By changing to provide copious extra encoding space in the packet format, it became easy to extend the design without modifications becoming intrusive. We have seen new message types added to the coherence protocol, larger packets used for bulk data transfers, more dimensions added to the network, and existing fields repurposed for new uses, all without interference with the system's regular operation. The only cost for this research flexibility was a one-to-two flit overhead on messages which are often ten or more flits long.

Configuration Registers
Researchers need software-controlled mechanisms to measure and configure their new research components. As with the NoC packet format, we learned to provide flexible configuration registers. We use these registers to enable, disable, and configure features and to read performance counters. The registers are integrated with the memory system (accessible through reserved physical addresses) where they can be easily extended and are placed behind cores' memory management units for protection.

Convenience and Productivity Improvements

Between our uses of OpenPiton for research and those of our users, we prioritize quality of life improvements to save research time.

Lessons: We noticed that if we had to complete a menial task in pursuit of a research goal, our users would often ask about those same tasks. To this end, we have made a variety of productivity-enhancing changes.

For example, we originally saw Debian boot take around 45 minutes with the OpenSPARC T1 core. Changes like improving frequency on FPGA and moving to a full 4-wire SDHC boot medium brought this down to less than five minutes.

Educational Use

OpenPiton has been used as a project platform in both undergraduate and graduate level computer architecture courses at Princeton University. It provides a great opportunity for the students to work on a large-scale, industry-standard, silicon-proven manycore framework. In 4-6 week course projects, students have implemented different architectural designs and verified research ideas through RTL simulation and FPGA emulation. Impressive student projects have been developed, including hardware transactional memory systems, comparing and evaluating different cache replacement policies, implementing different network topologies, and the development of our (now default) highly-parameterized chipset crossbar generator.

Managing and Maintaining an Open Infrastructure

Community Building and Platforms

Though much of OpenPiton's infrastructure was created to aid our own research, one of our primary goals in releasing OpenPiton is enabling others to easily and efficiently perform computing research. We used multiple methods to grow and maintain an active research community around the OpenPiton infrastructure.

Building the Community

Our primary tactic in building a community has been lowering the barrier to entry of using OpenPiton. Though early OpenPiton releases contained thorough documentation including user guides and architectural specifications, this documentation alone was not very effective at helping users get to work. We noted that we repeatedly referred to the same portions of our documentation when helping new users.

Lessons: Targeting the community's most common pain points, we improved our documentation and created a getting-started guide to make it easier for researchers to join our community. We also created virtual machine and Docker container environments preinstalled with open tools to run OpenPiton simulations.

We have also run thirteen tutorials on OpenPiton at major conferences and universities across the world. This invaluable opportunity enabled us to directly help researchers join our community while also letting us learn more about how users are interested in using OpenPiton. However, users who could not attend also showed interest in accessing our learning materials.

Lessons: We made available slides and a video recording of the tutorials online to ensure everyone has access to this useful information, not just those who were able to travel to one of our tutorials.

Maintaining the Community

Early on in OpenPiton's lifetime, providing direct, personalized support to our users ensured a new user would remain an active user.

Lessons: The OpenPiton Google group came to provide a forum for personalized support for each community member while also **creating an archive of common problems and their solutions**. Having public communication showed that the community itself was being actively supported, not just the codebase.

We used user feedback to improve OpenPiton then broadcast those changes through our mailing list.

Lessons: Providing regular updates highlighting requested features emphasized the active nature of our development and provided examples of positive change caused by community participation.

Community Contributions

Throughout the development of OpenPiton, we have been delighted to receive a variety of crucial community contributions. Significant among such contributions were frequency improvements for our FPGA implementation, bug fixes to eliminate data corruption in our original SD controller,

documentation improvements, simulator support, FPGA flow improvements, RTL portability improvements, and code modernization. Several of these enhancements were not on our roadmap at all but provided value to us and a large number of our users. In the case of the SD data corruption, we were aware that there was a bug (but not the details) at the time we were transitioning to the newer SDHC controller and the fix helpfully illuminated a similar bug to us in the new controller.

Closing The Loop

OpenPiton is great for system-on-chip (SoC) research in multiple domains, from applications to architecture to design automation. The framework enables users to expediently design, synthesize, test for bugs, check for timing violations, evaluate for area overhead, and prototype on FPGAs. Even better, OpenPiton offers the software support for SMP Linux-booting manycores, which opens up more exciting research opportunities. With OpenPiton as an explicitly modularized, ready-to-modify, and coherent framework, developing and proving research ideas is less ad-hoc and time-consuming. Therefore, OpenPiton is the ideal starting point for prototyping new ideas and enables closing the development and validation loop of many research projects. The coherent framework enables the easy exploration of an idea at different levels of hierarchy, which reveals latent problems and enhances implementations' completeness. Following are some example pieces of research that have been enabled.

Execution Drafting

Execution Drafting (ExecD) is an energy saving microarchitectural mechanism for multithreaded processors, which exploits duplicate computation [8]. The evaluation of ExecD started with a custom-built, trace-based simulator. Utilizing OpenPiton, ExecD was then implemented in RTL by modifying the OpenSPARC T1 core, and was synthesized to estimate the area overhead. Implementing ExecD in OpenPiton revealed several tricky implementation details that were abstracted away in simulation. The modified core was later taped out in Piton and the energy behavior was characterized in the real silicon. Notably, a hardware bug in Piton's ExecD implementation was found due to a mismatch between energy results from simulation and characterization. Going back to RTL simulation, it was then possible to feed the Piton characterization data into a bug-fixed implementation, which achieved better-than-estimated energy efficiency. Running ExecD on real hardware requires the kernel to decide when to enable the ExecD hardware. Notably, with OpenPiton's full-stack platform, kernel patches for real ExecD silicon was developed as part of a single semester undergraduate OS research project.

MITTS and PiCL

Other projects which closed the research loop with OpenPiton include MITTS [9] and PiCL [10]. Both started out with high-level simulations which needed further validation in the form of RTL implementation. Using OpenPiton as a platform, they were each integrated into the P-Mesh cache coherence system and then evaluated in either the Piton ASIC (for MITTS) or FPGA (for PiCL), where the final results could be compared back to the original first-order simulations. Both projects required low-level software support which was best evaluated in a full-system stack like that provided by OpenPiton.

Piton Chip Characterization

The Piton ASIC prototype was taped-out on IBM's 32nm SOI process and has been carefully studied for a detailed power and energy characterization [2]. The Piton chip is a 25-core instantiation of OpenPiton's scalable manycore architecture. At a basic level, the characterization results provide useful data about manycore processors taped out in advanced technology processes. More interestingly, researchers can use the combination of our open-source RTL, tool flow, and characterization results to develop and improve their own models. The data we provided back to the community is beginning to improve area, performance, power, and energy models and simulations of modern parallel processors.

Evolution of OpenPiton and Ongoing Improvements

Open Development

For the first few years of OpenPiton, we only provided our releases as tarballs downloadable at `openpiton.org`. This gave us the opportunity

before release to vet and remove code that may contain sensitive elements. However, tarballs have downsides and are unproductive for users moving to newer releases. Users asked us to take a more convenient and transparent approach to development.

Lessons: In 2018, we brought OpenPiton to GitHub (https://github.com/PrincetonUniversity/openpiton) and moved to a more open development model. We wrote a new Git history for the project reflecting the external releases and now use Git to straightforwardly release features while maintaining their history. Entire new features like OpenPiton+Ariane were built in the open, making their entire development history public. With this new approach, external contributors can easily submit Pull Requests (PRs) for bugfixes and new features. Furthermore, the Git history acts as a guide for integrating new features into OpenPiton. Going forward, we plan to develop as much in public as we can. This open development has led us to adopt new strategies for modularizing and firewalling sensitive IP so that we can release without having to manually vet the code.

PyHP

PyHP has served as a powerful tool in developing OpenPiton. Embedding arbitrary Python for Verilog generation enabled us to straightforwardly and productively parameterize the P-Mesh cache system. Our highly parameterizable P-Mesh chipset crossbars utilize PyHP to parse the XML design configuration and generate RTL at design time. This power is crucial in building realistic, large scale designs.

Lesson: As we have added heterogeneity, we have reached a trade-off point. There are challenges in generating multiple heterogeneous instances of components while enabling synthesis and simulation tools to intelligently deduplicate identical sub-components. From here, one might extend their generators to look more like higher-level hardware description languages. To avoid this complexity, we make more use of Verilog's native generate for heterogeneous features like heterogeneous tile instantiations.

Heterogeneity

RISC-V and Heterogeneous ISA

From OpenPiton's inception, we have focused on providing a high quality, scalable, cache coherent memory system. We have long said that we are not attached to the OpenSPARC T1 core's SPARC V9 ISA. However, we had not demonstrated P-Mesh's generality until recently.

With RISC-V's emerging importance to the research community, we wanted to live up to our claim. In 2017 we began to develop what we now call the Transaction-Response Interface (TRI) and the "Bring Your Own Core" (BYOC) platform [1]. TRI is a flexible and low overhead interface to connect new cores to OpenPiton and it is uniquely designed to enable the coexistence of a variety of cores, regardless of ISA. BYOC provides a level of heterogeneity that enables new research into cache-coherent, heterogeneous-ISA systems not just at the architecture level, but throughout the software stack.

To date, ten cores of four different ISAs (plus the QEMU full-system emulator) have been connected to BYOC. All cores share a common platform in terms of the peripherals and coherent memory operations available. Additionally, they all benefit from OpenPiton's thorough validation. When bringing up Linux on OpenPiton+Ariane, we went from single-core to dual-core boot in just one day, with no RTL modifications necessary. This was a startling level of productivity, particularly given the microarchitectural and ISA differences between the OpenSPARC T1 and Ariane cores.

Accelerators

To better enable research into heterogeneous SoCs, we have integrated several accelerators. The first, MIAOW [11], is an open-source GPGPU which implements AMD's Southern Islands ISA. For AI/ML inference, students in Princeton's Advanced Computer Architecture class connected Nvidia's deep learning accelerator (NVDLA). MIAOW and NVDLA were connected to the P-Mesh chipset crossbars and can be programmed from Linux running on the host core, making accelerator research easy.

Coherence

Modifying the cache coherence system to support different coherence models is an open research

question which requires a careful and principled approach. Designing the coherence protocol (in our case, P-Mesh) to be flexible and extensible is critical in enabling users' adaptations.

Our principled coherence protocol enabled us to formally model the original P-Mesh protocol in the Murphi modeling language and verified that it will operate deadlock-free. This gives us confidence in the foundations on which we are implementing our modifications and extensions.

From our experience so far, extensions like coherent/non-coherent direct memory access (DMA) and software-controlled cache coherence can be implemented in P-Mesh with the addition of just one or two new message types and the creative reuse of existing operations. The modified protocol can use the existing three NoCs while remaining deadlock-free.

The Next 5 Years

OpenPiton has grown from its inception as an architectural research platform for scalable many-cores to a powerful tool and valuable resource enabling agile research into many layers of the computing stack. Over the next few years, it will evolve to support the study of new and interesting research areas including heterogeneous computing, accelerators, open-source EDA, and other areas of interest within the diverse OpenPiton user community. This evolution will include the tapeout of a number of exciting new prototype chips. Meanwhile, we will use our open and agile design philosophy to ensure OpenPiton remains an easy-to-use, efficient, and powerful tool for computing research.

Acknowledgment

This work was supported in part by the National Science Foundation under Grant CNS-1823222 and Grant CCF-1453112; and in part by the Air Force Research Laboratory (AFRL) and Defense Advanced Research Projects Agency (DARPA) under Agreement FA8650-18-2-7846, Agreement FA8650-18-2-7852, and Agreement FA8650-18-2-7862. The U.S. Government is authorized to reproduce and distributed reprints for Governmental purposes notwithstanding any copyright notation thereon. 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), the NSF, or the U.S. Government. The authors would like to thank all contributors to OpenPiton for making the platform what it is today.

Jonathan Balkind is currently working toward the Ph.D. degree with the Department of Computer Science, Princeton University. His research interests include computer systems, programming languages, and computer architecture with the aim of improving the efficiency of modern multicore systems in mobile and datacenter environments. Balkind received the M.Sci. degree in computing science from the University of Glasgow and the M.A. degree in computer science from Princeton University. Contact him at jbalkind@princeton.edu.

Ting-Jung Chang is currently working toward the Ph.D. degree with the Department of Electrical Engineering, Princeton University. Her research interests include computer architecture, memory systems, and emerging transistor technologies. Chang received the M.A. degree in electrical engineering from Princeton University. Contact her at tingjung@princeton.edu.

Paul J. Jackson is currently working toward the Ph.D. degree with the Department of Electrical Engineering, Princeton University. His research interests include parallel computer architecture, energy efficiency, and serial computation. Jackson received the M.A. degree in electrical engineering from Princeton University. Contact him at pjj@princeton.edu.

Georgios Tziantzioulis is currently a Postdoctoral Research Associate with the Department of Electrical Engineering, Princeton University. His current research focus is on the design of power and energy efficient computer systems for Data-centers and Cloud Services. Tziantzioulis received the Ph.D. degree in computer engineering from Northwestern University and the Diploma degree in computer and communication engineering from University of Thessaly. Contact him at georgios.tziantzioulis@princeton.edu.

Ang Li is currently working toward the Ph.D. degree with the Department of Electrical Engineering, Princeton University. His research interests include heterogeneous computer architecture, reconfigurable computing, and system/software design for general-purpose reconfigurable fabrics. Li received the M.A.

degree in electrical engineering from Princeton University. Contact him at angl@princeton.edu.

Fei Gao is currently working toward the Ph.D. degree with the Department of Electrical Engineering, Princeton University. His research interests include in-memory compute, memory systems, and many-core processor design. Gao received the M.A. degree in electrical engineering from Princeton University. Contact him at feig@princeton.edu.

Alexey Lavrov is currently working toward the Ph.D. degree with the Department of Electrical Engineering, Princeton University. His research interests include multitenant I/O devices, multicore processor design, and hardware support for networking. Lavrov received the M.A. degree in electrical engineering from Princeton University and the M.A. degree in applied physics and math from the Moscow Institute of Physics and Technology. Contact him at alavrov@princeton.edu.

Grigory Chirkov is currently working toward the Ph.D. degree with the Department of Electrical Engineering, Princeton University. His research focuses on manycore processors and coherence protocols. Chirkov received the B.S. degree in applied physics and math from the Moscow Institute of Physics and Technology. Contact him at gchirkov@princeton.edu.

Jinzheng Tu is currently working toward the Ph.D. degree with the Department of Electrical Engineering, Princeton University. Her research focuses on future computation and communication beyond the end of Moore's Law. Tu received the B.S. degree in electrical engineering from Tsinghua University. Contact her at jinzheng@princeton.edu.

Mohammad Shahrad is currently working toward the Ph.D. degree with the Department of Electrical Engineering, Princeton University. His research aims to improve the efficiency of public cloud systems through better resource management and enhanced vertical integration. Before joining Princeton University, he received the bachelor's degree in Electrical Engineering from Sharif University of Technology. Contact him at mshahrad@princeton.edu.

David Wentzlaff is currently an Associate Professor with the Electrical Engineering Department, Princeton University. His research interests include parallel computer architecture, architectures for cloud computing, and biodegradable computing systems. He has received the NSF CAREER award, the DARPA Young Faculty Award, the AFOSR Young Investigator Prize, and the Princeton E. Lawrence Keyes Faculty Advancement Award. Wentzlaff received the master's and Ph.D. degrees in electrical engineering and computer science from the Massachusetts Institute of Technology. Contact him at wentzlaf@princeton.edu.

REFERENCES

1. J. Balkind, K. Lim, F. Gao, M. Schaffner, G. Chirkov, A. Li, A. Lavrov, T. Nguyen, Y. Fu, F. Zaruba, K. Gulati, L. Benini, and D. Wentzlaff, "BYOC: A "Bring Your Own Core" framework for heterogeneous-ISA research." ASPLOS, March 2020.
2. M. McKeown, A. Lavrov, M. Shahrad, P. J. Jackson, Y. Fu, J. Balkind, T. M. Nguyen, K. Lim, Y. Zhou, and D. Wentzlaff, "Power and energy characterization of an open source 25-core manycore processor." in *HPCA*, 2018, pp. 762–775.
3. C. Tan, Y. Ou, S. Jiang, P. Pan, C. Torng, S. Agwa, and C. Batten, "PyOCN: A unified framework for modeling, testing, and evaluating on-chip networks," in *ICCD*. IEEE, 2019.
4. W. L. Neto, M. Austin, S. Temple, L. Amaru, X. Tang, and P.-E. Gaillardon, "LSOracle: a logic synthesis framework driven by artificial intelligence," in *ICCAD*. IEEE, 2019, pp. 1–6.
5. T.-W. Huang and M. D. Wong, "OpenTimer: A high-performance timing analysis tool," in *ICCAD*. IEEE, 2015, pp. 895–902.
6. T. Ajayi, D. Blaauw, T. Chan, C. Cheng, V. Chhabria, D. Choo, M. Coltella, S. Dobre, R. Dreslinski, M. Fogaça *et al.*, "OpenROAD: Toward a self-driving, open-source digital layout implementation tool chain," in *DAC*. ACM, 2019, p. 76:1–76:4.
7. B.-Y. Huang, H. Zhang, P. Subramanyan, Y. Vizel, A. Gupta, and S. Malik, "Instruction-level abstraction (ILA): a uniform specification for system-on-chip (SoC) verification," *TODAES*, vol. 24, no. 1, p. 10, 2018.
8. M. McKeown, J. Balkind, and D. Wentzlaff, "Execution Drafting: Energy efficiency through computation deduplication," in *MICRO*. IEEE, 2014, pp. 432–444.
9. Y. Zhou and D. Wentzlaff, "MITTS: Memory inter-arrival time traffic shaping," in *ISCA*. IEEE, 2016, pp. 532–544.
10. T. Nguyen and D. Wentzlaff, "PiCL: A software-transparent, persistent cache log for nonvolatile main memory," in *MICRO*. IEEE, 2018, pp. 507–519.
11. R. Balasubramanian, V. Gangadhar, Z. Guo, C.-H. Ho, C. Joseph, J. Menon, M. P. Drumond, R. Paul, S. Prasad, P. Valathol, and K. Sankaralingam, "Enabling GPGPU low-level hardware explorations with MIAOW:

An open-source RTL implementation of a GPGPU," *ACM Transactions on Architecture and Code Optimization*, vol. 12, no. 2, Jun 2015.