Research Statement – Christopher Torng

Computer architecture is the bridge connecting software to silicon. The coming decades will likely be the most exciting times to invest in computer architecture. The ending of Moore's Law and Dennard Scaling has significantly slowed device-level advancements in compute, but new computing trends are creating opportunities for innovation by bridging fields. Software developers have increasingly moved towards domain-specific languages and frameworks (e.g., for machine learning and graph analytics), creating opportunities for architects to design specialized, highly efficient, domain-specific architectures. Circuits advances in new technologies (e.g., emerging memories, sensors, synchronizers) for emerging markets (e.g., the Internet of Things, augmented reality) have introduced new capabilities that architects can explore and leverage at the system level. These opportunities place a tremendous burden on architects to explore non-traditional systems beyond conventional processor design. In the past, the high costs and challenges of building hardware would have prevented architects from properly exploring these opportunities. Today, a new wave of open-source hardware is providing community-developed infrastructure that is steadily reducing key barriers to hardware innovation. In a broader context, the growing worldwide concerns regarding security, energy, and health indicate that architects will continue to be essential for bridging the gap between software domains and hardware. In the coming era, architects will interact more closely with other disciplines, break traditional computing abstractions, and also create entirely new ones. Tremendous opportunities are available for architects in the coming decades. As John Hennessy and Dave Patterson stated in their Turing Award Lecture, we are entering a new golden age of computer architecture.

My research efforts at Cornell University focus on a common theme of **interdisciplinary co-design across software, architecture, and VLSI/circuits**. My primary research goal is to build specialized architectures that tie together software with the underlying devices. My research approach emphasizes a vertically integrated methodology that spans across languages, runtimes, compilers, instruction sets, microarchitecture, VLSI, circuits, and also includes developing FPGA/ASIC prototypes that concretely demonstrate ideas. As a graduate student at Cornell University I have published six conference papers [1, 5, 7–9, 13] (ISCA, MICRO, DAC, HOTCHIPS), two journal papers [3, 4] (IEEE MICRO, IEEE TCAS I), and four workshop papers/posters [2, 6, 12, 14]. Of these papers, I was the first author or co-first author on two papers at top-tier computer architecture venues [5, 13] (ISCA, MICRO) and a key contributor to an IEEE MICRO Top Picks journal selection from Hot Chips [4]. In addition to my publications, I have been involved in six tapeouts that support my research, including an analog test chip and five digital ASIC test chips. Of these chips, I was the project lead for two chips (BRGTC1 in IBM 130 nm [14] and BRGTC2 in TSMC 28 nm [12]) and university student lead for the DARPA-funded, multi-university project on developing the Celerity SoC in TSMC 16 nm [1, 4]. Finally, I have open-sourced my own modular VLSI build system which we have used at Cornell University to tape out two chips [11].

In this research statement, I will discuss my three key research thrusts: (1) software, architecture, and VLSI co-design for task-based parallel runtimes; (2) architecture and circuit co-design for integrated voltage regulation; and (3) methodologies for rapid ASIC design. I will then end by discussing my plans for future research.

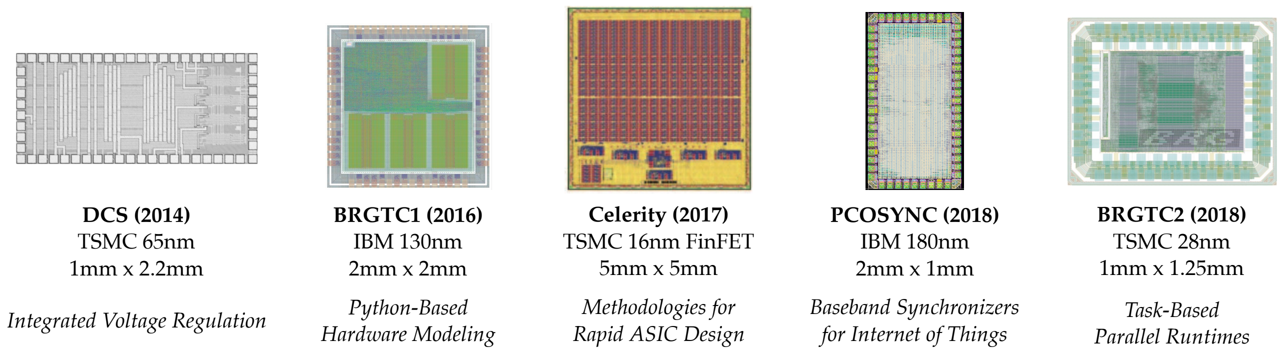

Figure 1. Timeline – I have been involved with six chips in my PhD (five shown) that support my vertically integrated research.

Thrust I – Software, Architecture, and VLSI Co-Design for Task-Based Parallel Runtimes

Software programmers are increasingly moving away from *thread-centric* programming models and instead gravitating towards *task-centric* programming models. Working with threads involves tedious low-level thread management in order to manually balance a workload across multiple processing elements. In a task-centric programming model, the programmer instead dices the workload into units of work called *tasks* which can run in parallel, before handing these off to a runtime that schedules the tasks across worker threads. The task-centric programming model promises higher productivity by allowing programmers to work at a higher level of abstraction while offloading hardware-centric issues to the underlying runtime. While many runtimes are possible, a *work-stealing runtime* is a well-known approach in which worker threads enqueue and dequeue tasks onto the tail ends of per-thread task queues. When a worker finds its queue empty, it attempts to *steal* a task from the head of another worker thread's task queue. Work stealing has been shown to have good performance, space requirements, and communication overhead in both theory and practice (e.g., Intel Cilk++, Intel TBB, OpenMP).

In parallel, recent technology trends suggest that we are entering a new era in which it is becoming feasible to integrate high-quality switching voltage regulators on-chip. Fully integrated voltage regulation (IVR) reduces system cost due to monolithic integration of expensive board-level components but also offers the potential for fine-grain dynamic voltage/frequency scaling (DVFS) in level (i.e., many different voltage levels), space (i.e., per-core regulation), and time (i.e., fast transition times between levels). Industry has marked a clear trend toward fine-grain DVFS with per-core scaling enabled in Intel's Haswell/Broadwell microprocessors with in-package inductors and on-chip regulators and more recently in AMD's Ryzen Mobile microprocessors with per-core linear regulators.

Asymmetry-Aware Work-Stealing Runtimes – The underlying hardware of a work-stealing runtime may exhibit both *static asymmetry* (e.g., different core microarchitecture) and *dynamic asymmetry* (e.g., applying fine-grain DVFS). I made the key observation that runtimes can be made aware of underyling asymmetry to create more efficient schedules and to dynamically tune processing elements. I proposed an asymmetry-aware work-stealing runtime based on three key software/hardware techniques: work-pacing, work-sprinting, and work-mugging. Work-pacing and work-sprinting are novel techniques that greatly improve both performance and energy efficiency of the runtime by *sprinting* (i.e., increasing voltage and frequency) threads that are executing tasks and *resting* (i.e., decreasing voltage and frequency) threads that are waiting for work in the steal loop. To identify the steal loop to the hardware, I instrumented an in-house work-stealing runtime similar to Intel TBB with lightweight hint instructions to delimit task regions and then pushed this activity information down the software-hardware interface to toggle activity bits in each core. To systematically determine the voltages/frequencies for a heterogeneous system, I designed a first-order analytical power/performance model for systems of big and little cores under DVFS. With this analytical model, I made the key discovery that optimal settings will balance *marginal utility* by "selling expensive big-core performance" and "buying cheap little-core performance". Work-pacing and work-sprinting apply this approach similarly in the high- and low-parallel regions. Work-mugging is a previously proposed theoretical technique that enables a waiting big core to preemptively migrate work from a busy little core. I proposed a simple implementation of work-mugging based on lightweight user-level interrupts. Compared to a state-of-the-art asymmetry-oblivious work-stealing runtime, an asymmetry-aware work-stealing runtime can improve performance by up to 1.44$\times$ and energy efficiency up to 1.58$\times$. I presented this work at the 43rd ACM/IEEE Int'l Symp. on Computer Architecture (ISCA). The paper has already encouraged other researchers to explore the impacts of static and dynamic asymmetry on server tail latency (Haque et al., MICRO'17) despite the recency of this publication. To my knowledge, I am the first to propose exploring work-stealing, static asymmetry, and dynamic asymmetry together [13].

ASIC Prototyping for Work-Stealing Runtimes – In the spring of 2018, I led a team of six graduate students for two months to design and fabricate a 1$\times$1.25 mm 6.7M-transistor RISC-V system in TSMC 28 nm called BRGTC2 (see Figure 1) which runs a work-stealing runtime. One of the (several) research goals of the chip is to provide detailed performance, area, and energy numbers in an advanced technology node to support future research projects based on hardware acceleration for task-based parallel runtimes. The chip architecture includes four RISC-V RV32IMAF cores which share a 32kB instruction cache, 32kB data cache, and single-precision floating point unit along with microarchitectural mechanisms to mitigate the performance impact of resource sharing. I presented a workshop paper about BRGTC2 at the RISC-V Day Workshop in Fukuoka, Japan [12].

Loop-Task Accelerators for Task-Based Parallel Programs – Software programmability is an important consideration when evaluating the viability of a hardware accelerator. I was involved in the design of a task-centric accelerator template with my colleague Ji Kim, where we motivated and developed an elegant approach for pushing the abstraction of tasks down to the hardware for direct interpretation by the accelerator. I designed a rigorous energy model for the cores and accelerator based on post-PnR gate-level power estimation. This work was published at the 50th ACM/IEEE Int'l Symp. on Microarchitecture (MICRO) [8].

Thrust II – Architecture and Circuit Co-Design for Integrated Voltage Regulation

Fully integrated voltage regulation is an attractive prospect due to its promise of reduced system cost and the potential for fine-grain DVFS. However, on-chip regulators have notable key challenges including lower on-chip conversion efficiencies and on-die area overheads. Careful architecture/circuit co-design can mitigate challenges for integrated voltage regulation while uncovering benefits from both circuits and systems perspectives.

Enabling Realistic Fine-Grain Voltage Scaling with Reconfigurable Power Distribution Networks (RPDNs) – Enabling fine-grain DVFS for a fabric of processing elements typically assumes dedicated per-processing-element regulators integrated on-chip, allowing each processing element to rest and sprint. However, this approach significantly over-provisions regulator area, since all processing elements can never be in the fastest operating mode at the same time without exceeding the thermal budget. Based on this observation, my colleagues and I proposed a new approach called reconfigurable power distribution networks (RPDNs) which applies a classic architecture technique for *resource sharing* in the context of integrated voltage regulation in order to mitigate on-die area overheads. The key idea of an RPDN is to dice the energy storage (e.g., capacitors) into many small "unit cells" shared among the processing elements in the design. Each unit cell contains the flying capacitance and regulator switches required for a switched-capacitor voltage regulator. The unit cells can be flexibly reconfigured through a switch fabric and combined with per-processing-element control circuitry to effectively create multiple regulators "on-demand". This reconfiguration reduces area overhead by avoiding the over-provisioning inherent in dedicated per-processing-element regulators. In addition, simultaneous adjustment of flying capacitance and regulator switching frequency provides an order-of-magnitude improvement in response time, enabling faster voltage scaling for fine-grain DVFS. Professor Apsel and her students developed detailed circuit-level models of RPDNs, while I took the lead in evaluating the architecture-level benefit of fine-grain DVFS using RPDNs. Similar to the approach used in my asymmetry-aware work-stealing runtimes, I first informed hardware of thread activity by embedding lightweight activity hints in the thread library. I designed a DVFS controller using this information and conducted a limit study, gathering important answers regarding how to exploit fine-grain DVFS with the least system complexity: how many voltage levels to support (at least three), how many voltage domains are necessary (finer-grain is better), and how quick voltage transitions must be to adapt to fine-grain activity imbalance in multithreaded applications (order of 100 ns). Finally, I composed SPICE-level models of RPDNs designed by Professor Apsel's students with my architectural models and demonstrated how fine-grain DVFS could improve both performance and energy efficiency at the same time compared to dedicated per-core regulators, while also reducing the area overhead by approximately 40%. I was an equal-contribution first-author and presented the work at the 47th ACM/IEEE Int'l Symp. on Microarchitecture (MICRO) [5]. The paper has been cited across top-tier venues in both the circuits community (e.g., cited in JSSC, TCAS I, VLSI, DAC) and the architecture community (e.g., cited in MICRO, HPCA, ISCA).

Dynamic Capacitance Sharing (DCS) Analog Test Chip – Dynamic capacitance sharing is a novel circuits technique for dynamically sharing small units of capacitance across multiple on-chip switched-capacitor regulators for reduced on-chip area and order-of-magnitude faster voltage transition times. Professor Alyssa Apsel and her students led a tapeout with Professor Christopher Batten and myself contributing to the fabrication of four monolithically integrated switched-capacitor DC-DC converters with the DCS technique in 65-nm CMOS (see Figure 1). I designed the digital configuration blocks in full-custom with Cadence Virtuoso while adhering to a traditional track-based standard-cell-based approach. I was also deeply involved in the post-silicon test and characterization of the chip at different voltages, switching frequencies, topologies, and load currents. The characterization of the test chip was published in the IEEE Transactions on Circuits and Systems I (IEEE TCAS I), a top-tier circuits journal [3].

Thrust III – Methodologies for Rapid ASIC Design

Rising SoC design costs have created a formidable barrier to hardware design when using traditional design tools and methodologies. In particular, small chip-building teams in business (i.e., chip-based startups), in academia (i.e., research groups), and even in the U.S. Department of Defense struggle to build meaningfully complex designs with a limited workforce in reasonable timeframes. I have been involved in a range of efforts to reduce the costs and challenges of ASIC design for small teams based on productive toolflows and open-source hardware.

The Celerity Open-Source 511-Core RISC-V Tiered Accelerator Fabric – Celerity is a 5×5 mm 385M-transistor SoC implemented in an advanced 16 nm technology across four universities (UC San Diego, U of Washington, U of Michigan, and Cornell). The chip is a DARPA-funded research vehicle for a range of productive hardware design and verification tools that are creating excitement in the hardware community including: synthesizable analog IP (synthesizable PLL, digital LDO), high-level synthesis (complex HLS-generated binarized neural network accelerator), Python-based hardware modeling (composition logic, BNN wrapper logic), techniques for area-efficient and high-bandwidth manycore networks (496-core RISC-V tiled manycore, remote-store programming model), and reuse from open-source libraries and generators (BaseJump IP, Chisel-generated RISC-V Rocket cores). As the Cornell University student lead, I was a key player in integrating all of these methodologies and tools into the final chip. I collaborated frequently with Prof. Dreslinski (Michigan) as well as with Prof. Taylor (Washington). In addition to leading the Rocket+BNN accelerator logical/physical design, I also contributed to project management and team-building. I visited the University of Michigan as an academic exchange student for two months, where I worked with Prof. Dreslinski and made key contributions to physical design and verification. I trailblazed our first power strategy as well as our first DRC/LVS-clean block layout. I developed our gate-level simulation infrastructure and contributed to the final top-level integration, LVS, DRC, power signoff, and timing closure. The Celerity SoC was published in Hot Chips: A Symposium on High Performance Chips [1], covered in the media in the EE Times [10], and followed by an IEEE MICRO Top Picks from Hot Chips selection [4], where I was a key contributor.

Batten Research Group Test Chip 1 (BRGTC1) – In 2016, I led a team of eight graduate students to build our research group's first computer architecture test chip. I was involved from architecture all the way down to silicon. BRGTC1 is a $2{\times}2$ mm 1.3M-transistor chip in IBM 130 nm designed and implemented using our new open-source Python-based hardware modeling framework called PyMTL. PyMTL brings compelling productivity benefits to hardware design and verification. The key idea of the framework is to embed a hardware design language within the high-level Python host language and then to leverage Python's powerful features (e.g., parameterization, reflection) as well as access to rich libraries (e.g., full-featured software testing frameworks) to rapidly design, verify, and compose hardware. I built BRGTC1 to silicon-validate PyMTL and to experiment with PyMTL interaction with HLS. The chip includes a simple pipelined processor, custom LVDS clock receiver, 16 kB of SRAM, and application-specific HLS-generated accelerators. I also worked with a masters student to emulate BRGTC1 on an FPGA (i.e., a Xilinx Zedboard) before final tapeout. I presented a poster and short writeup about BRGTC1 at Hot Chips [14].

Batten Research Group Test Chip 2 (BRGTC2) – As previously discussed, BRGTC2 was designed to support research based on hardware acceleration for work-stealing runtimes. BRGTC2 is also designed and implemented almost entirely in PyMTL, contributing to a focus on new LLVM-inspired analysis and transform pass support within PyMTL to facilitate hardware design and test. While working on the chip, I designed and open-sourced my *Modular VLSI Build System* [11] to increase the flexibility and reuse of ASIC flows. The key idea is to avoid rigidly structured ASIC flows that cannot be repurposed and to instead break the ASIC flow into modular steps that can be re-assembled into different flows. Finally, BRGTC2 is also a research vehicle for exploring a synthesizable PLL (ported from Celerity) and its potential as an open-source PLL generator for the community. I wrote a workshop paper based on BRGTC2 describing trends in a new era of silicon prototyping in computer architecture research [12].

High-Productivity SoC Design based on High-Level Synthesis – I interned at NVIDIA in 2017 with Brucek Khailany's ASIC/VLSI team, where I explored performance-accurate SystemC simulation with HLS feedback. While the work was left incomplete, the project was an example of an emerging class of research built upon the new abstraction of HLS-based hardware design. A small piece of my work was included in a broader paper on productive SoC design with HLS libraries published in the 55th ACM/IEEE Design Automation Conference (DAC) [7].

Future Research

I plan to continue my broad research goal of building specialized architectures that tie together software with the underlying devices. Despite my primary focus as a computer architect, my vertically integrated research approach and my previous experience gives me flexibility to work across the computing stack.

I am particularly interested in applying my research approach to the Internet of Things (IoT). Healthcare, power grids, smart homes, and many others can benefit from the IoT. For example, IoT healthcare can enable independent living for the elderly, while IoT smart traffic can quickly respond to emergencies. While the IoT often assumes processing in the cloud, I am particularly interested in *fog computing*, where edge processing occurs in a "smart gateway". Fog computing can provide better security (i.e., keep private info local), real-time response (i.e., low network latency), less power (i.e., close by), preprocessing (i.e., reduce data sent to cloud), and abstraction (e.g., abstract away underlying sensors). Computer architects have great potential to build architectures for smart gateways by interfacing with the cloud (software) and perception layer (devices). As a concrete example, network protocol needs differ between the cloud and the perception layer. The de facto cloud network protocol is IPv6, but this is expensive and over-provisioned for energy-sensitive sensors operating on batteries or harvested energy. Since distance often determines power, we could dynamically switch between cheaper protocols (e.g., Zigbee, 6LoWPAN) depending on the nearest smart gateway, which could be further complicated by obstacles moved in between. Also of note are reconfigurable sensors, which may normally observe in low resolution but then sharpen after detecting an oddity. This creates a wide range of power demand with burstiness, which I can potentially address with fine-grain power management. I am interested in IoT as an architect comfortable with both software (i.e., the cloud) and VLSI/circuits (i.e., the perception layer). I was also involved in an ASIC tapeout for an IoT-oriented scalable baseband synchronizer in collaboration with Prof. Apsel's group on the unpublished PCOSYNC project (see Figure 1).

Closer to my current research direction, I am also excited about combining specialized architectures with hardware design patterns that inherently decrease the costs and challenges of design and verification. I am interested in extending previous work on *globally asynchronous, locally synchronous* (GALS) methodologies, where the key idea is to dice a large design into smaller synchronous islands which can be tiled together in "Lego-block" fashion using asynchronous interfaces. The three main benefits of a GALS methodology are: (1) reduced global clock power, (2) reduced design complexity by tiling smaller sub-designs, and (3) inherently supporting fine-grain DVFS for each island. GALS-based designs are good candidates for dynamic performance scaling of often-bottlenecked resources in sharing architectures (e.g., shared convolution and other engines) due to the support for fine-grain DVFS. Unfortunately, sub-dividing a GALS tile into resource-level GALS "sub-tiles" provides only minimal further benefit to clock power and design complexity while also increasing the number of asynchronous crossings, which are notoriously challenging to verify (and verification is a key challenge for GALS designs). One research idea that may address these challenges is to quantize the space of possibilities to synchronous *ratiochronous* relationships (e.g., frequency ratios of 1-to-3, 2-to-3), which enables industry-standard static timing analysis techniques to verify timing within a tile. Applying ratiochronous design at the resource level can sidestep verification challenges while still enabling (quantized) fine-grain DVFS within a tile. A ratiochronous design style can be a key enabling factor for designing sharing architectures that are capable of dynamically sprinting through resource bottlenecks using fine-grain DVFS.

I plan to seek funding from the government (e.g., Department of Defense, NSF) as well as from companies (e.g., SRC). I have experience with DARPA CRAFT and two programs in DARPA's Electronics Resurgence Initiative (ERI): DARPA SDH and DARPA POSH. I personally attended internal DARPA CRAFT PI meetings, where I observed the demanding nature of DARPA-funded projects. I contributed to an NSF grant proposal with Prof. Batten and Prof. Studer at Cornell for the NSF XPS program in 2014. Although the proposal was not funded, I gained valuable experience in grant writing. In June 2018, I attended the Google Accelerated Compute Research Summit with my advisor to create industry-academia ties. I interned at NVIDIA, where by watching their own DARPA CRAFT push, I observed how the goals of companies and academics can synergistically align. I plan to seek funding for ASIC prototypes from the DoD and from companies, where such efforts more readily align with their research strategies.

I am excited to build specialized architectures that tie together software with the underlying devices, and I hope to collaborate with researchers across the stack and also in other domains (e.g., security, energy, and health). In the coming decades, I look forward to being involved in the most exciting times to invest in computer architecture.

References

[1] T. Ajayi, K. Al-Hawaj, A. Amarnath, S. Dai, S. Davidson, P. Gao, G. Liu, A. Lotfi, J. Puscar, A. Rao, A. Rovinski, L. Salem, N. Sun, C. Torng, L. Vega, B. Veluri, X. Wang, S. Xie, C. Zhao, R. Zhao, C. Batten, R. G. Dreslinski, I. Galton, R. K. Gupta, P. P. Mercier, M. Srivastava, M. B. Taylor, and Z. Zhang. Celerity: An Open Source RISC-V Tiered Accelerator Fabric. *Symp. on High Performance Chips (Hot Chips)*, Aug 2017.

[2] T. Ajayi, K. Al-Hawaj, A. Amarnath, S. Dai, S. Davidson, P. Gao, G. Liu, A. Rao, A. Rovinski, N. Sun, C. Torng, L. Vega, B. Veluri, S. Xie, C. Zhao, R. Zhao, C. Batten, R. G. Dreslinski, R. K. Gupta, M. B. Taylor, and Z. Zhang. Experiences Using the RISC-V Ecosystem to Design an Accelerator-Centric SoC in TSMC 16nm. *Workshop on Computer Architecture Research with RISC-V (CARRV)*, Oct 2017.

[3] I. Bukreyev, C. Torng, W. Godycki, C. Batten, and A. Apsel. Four Monolithically Integrated Switched-Capacitor DC-DC Converters with Dynamic Capacitance Sharing in 65-nm CMOS. *IEEE Transactions on Circuits and Systems I (TCAS-I)*, Nov 2017.

[4] S. Davidson, S. Xie, C. Torng, K. Al-Hawaj, 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. G. 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*, Mar 2018.

[5] W. Godycki*, C. Torng*, I. Bukreyev, A. Apsel, and C. Batten. Enabling Realistic Fine-Grain Voltage Scaling with Reconfigurable Power Distribution Networks. *Int'l Symp. on Microarchitecture (MICRO)*, Dec 2014. * = *equally contributing co-first authors*.

[6] S. Jiang, C. Torng, and C. Batten. An Open-Source Python-Based Hardware Generation, Simulation, and Verification Framework. *Workshop on Open-Source EDA Technology (WOSET)*, Nov 2018.

[7] B. Khailany, E. Krimer, R. Venkatesan, J. Clemons, J. Emer, M. Fojtik, A. K. andMichael Pellauer, N. Pinckney, Y. S. Shao, S. Srinath, C. Torng, S. L. Xi, Y. Zhang, and B. Zimmer. A Modular Digital VLSI Flow for High-Productivity SoC Design. *Design Automation Conf. (DAC)*, Jun 2018.

[8] J. Kim, S. Jiang, C. Torng, M. Wang, S. Srinath, B. Ilbeyi, K. Al-Hawaj, and C. Batten. Using Intra-Core Loop-Task Accelerators to Improve the Productivity and Performance of Task-Based Parallel Programs. *Int'l Symp. on Microarchitecture (MICRO)*, Oct 2017.

[9] J. Kim, C. Torng, S. Srinath, D. Lockhart, and C. Batten. Microarchitectural Mechanisms to Exploit Value Structure in SIMT Architectures. *Int'l Symp. on Computer Architecture (ISCA)*, Jun 2013.

[10] R. Merritt. ASIC done in nine months for $1.3M. Online Webpage, 2018 (accessed Oct 20, 2018). `https://www.eetimes.com/document.asp?doc_id=1332192&page_number=10`.

[11] C. Torng. The Modular VLSI Build System. Online Webpage, 2018 (accessed Oct 20, 2018). `https://github.com/cornell-brg/alloy-asic`.

[12] C. Torng, S. Jiang, K. Al-Hawaj, I. Bukreyev, B. Ilbeyi, T. Ta, L. Cheng, J. Puscar, I. Galton, and C. Batten. A New Era of Silicon Prototyping in Computer Architecture Research. *RISC-V Day Workshop*, Oct 2018.

[13] C. Torng, M. Wang, and C. Batten. Asymmetry-Aware Work-Stealing Runtimes. *Int'l Symp. on Computer Architecture (ISCA)*, Jun 2016.

[14] C. Torng, M. Wang, B. Sudheendra, N. Murali, S. Jayasuriya, S. Srinath, T. Pritchard, R. Ying, and C. Batten. Experiences Using A Novel Python-Based Hardware Modeling Framework For Computer Architecture Test Chips. *Poster at the Symp. on High Performance Chips (Hot Chips)*, Aug 2016.