Qian Zhang — Research Statement

My research area is Software Engineering for Data and Compute Intensive Systems. My research spans software engineering (SE), hardware design, and data-intensive scalable computing (DISC) systems.

Emerging hardware, like FPGAs and quantum circuits, is shaping the future of heterogeneous computing; however, the use of such extraordinary computing power is restricted to a few software developers with hardware expertise. **The vision of my research** is to **democratize heterogeneous computing** with re-designed developer productivity tools. I have contributed to lowering the barriers of porting traditional software to heterogeneous platforms by reinventing automated test input generation [1, 2, 3, 4], automated refactoring [5], and transpilation [6] tools.

My Ph.D. research *focused on hardware design*—heterogeneous computing with energy-efficient ASIC and FPGA accelerators. My approach traded accuracy for energy efficiency. My work on approximate computing focused on: (1) accuracy-configurable hardware design [7, 8, 9, 10, 11, 12, 13, 14]; (2) runtime precision control [15, 16]; and (3) resilience-aware task scheduling on multicore platforms [17, 18]. My Ph.D. work produced 12 research papers in top-tier conferences and journals, including **DAC, ICCAD, DATE**, and **TCAD**. My pioneering work in designing computation and memory co-optimized neural network accelerators was ranked top 1% of the most cited papers published in DATE'15, to date. Although cutting-edge hardware technologies provide substantial benefits, they are often out of reach for general software developers. Software tools are essential to the practical adoption of emerging hardware.

My postdoc research *at UCLA focused on software engineering*—reinventing testing, refactoring, and program repair tools for heterogeneous computing and big data analytics. My postdoc work produced 6 research papers in top-tier conferences, including **ESEC/FSE, ICSE, ASE**, and **ASPLOS**. All of my recent work has produced open-source tools [19, 20, 21, 22] to encourage extended adoption.

1. Efficient Fuzzing with Reduced Testing Latency

Fuzz testing finds error-inducing inputs in software by repeatedly executing the program with new inputs that are mutated from seed inputs. However, the long latency of big data and heterogeneous applications prohibits the applicability of fuzzing. For example, Apache Spark needs about 10 seconds to warm up a cluster for each run. Likewise, heterogeneous applications take from minutes to hours for kernel simulations. I have been on the quest to build *practical and efficient* fuzzers that can overcome this challenge.

#1. BigFuzz (ASE 2020). My work BigFuzz [3] makes fuzzing tractable for Spark data analytics programs. I observe that a significant chunk of big data analytics code comes from the DISC framework; however, most bugs appear in the implementation of application code and user-defined functions (UDFs). This observation forms the fundamental insights behind BigFuzz —we can reduce the testing latency by abstracting the DISC framework using executable specifications and focusing on application logic. Additionally, BigFuzz mutates seed inputs guided by common error types and monitors the joint dataflow and UDF coverage. With framework abstraction, BigFuzz achieves 1477× speedup than random fuzzing.

#2. HeteroFuzz (ESEC/FSE 2021). Heterogeneous applications offload their compute-intensive kernels to accelerators like FPGAs, which can lead to different outcomes across platforms. My work HeteroFuzz adapts fuzzing to detect divergence errors between CPU and CPU+FPGA by monitoring accelerator-relevant value spectra (*e.g.,* bitwidth and stack depth). Two additional optimizations include: (1) adjusting the activation probability of input mutations based on the potential to reveal divergence errors and (2) reducing unnecessary invocation of hardware simulators by memorizing the seen input ranges. Compared to a downgraded version of Heterofuzz without spectra monitoring, HeteroFuzz finds 7.78× divergence-inducing inputs which detect 3.7× more divergence symptoms.

Broader Impacts. I helped my advisor write an NSF proposal on this very topic of automated fuzz testing for data-intensive and compute-intensive systems, which was funded this year. This NSF medium grant is directly based on my postdoc research of BigFuzz and HeteroFuzz.

2. Effortless Programming with Automated Refactoring and Transpilation

High-level synthesis (HLS) tools enable engineering heterogeneous applications with high-level programming languages such as C/C++ and automatically generate hardware descriptions of kernel functions.

While HLS tools take kernel code in C variants, a developer must perform a substantial amount of manual refactoring to make it *synthesizable* and *efficient*. These efforts are often tedious and error-prone. To address this challenge, I have worked on automated refactoring of HLS-C programs [5] and automated transpilation from C to HLS-C [6].

#1. HeteroRefactor (ICSE 2020). My work HeteroRefactor [5] expands the automated refactoring in SE to ease the process of creating synthesizable and efficient FPGA circuits using HLS. The key insight behind this work is to optimize FPGA kernels by leveraging dynamic input characteristics. HeteroRefactor collects dynamic invariants over sample inputs and maps the invariants to three kinds of HLS optimizations: (1) recursion transformation for synthesizability, (2) integer bit optimization, and (3) floating-point tuning with a probabilistic guarantee. HeteroRefactor achieves up to 83% BRAM reduction, 61% Flip-Flop reduction, 39% Lookup Table reduction, and 52% DSP reduction with *no* human effort required.

#2. HeteroGen (ASPLOS 2022). My work HeteroGen [6] transpiles a C/C++ program to its equivalent HLS-C version. The insight is to leverage search-based program repair for legacy code rewriting. HeteroGen also generates tests to ensure behavior preservation and expedites the exploration with dependence-based search. Its auto-generated inputs provide 2.7× coverage than pre-existing tests. Dependence-based search contributes to 35× speedup than random search-based repair exploration.

Broader Impacts. This line of research is a part of the Intel/NSF CAPA project collaborating with people from the SE/PL community and the computer design automation community. My work is the first attempt to adapt dynamic invariant analysis and automated software repair to the hardware design domain.

3. Differential Testing for Quantum Software Stacks

QDiff (ASE 2021). A quantum computer is another silicon alternative that serves as a hardware accelerator. I mentored and collaborated with a Ph.D. student Jiyuan Wang in my group to design QDiff for testing quantum language constructs, compilers, and backend simulators. It has three innovations: (1) quantum-specific program variant generation; (2) outcome comparison based on K-S test; and (3) circuit filtering based on static characteristics (*e.g.,* circuit depth). Quantum hardware is in its imperfection; understanding different outcomes across a QSS can help construct a robust quantum ecosystem in the future.

Future Work

In the **long term**, my research aims to create a subfield of *software engineering for distributed, heterogeneous hardware enabled cloud platforms* that reinvents existing software tools and a system runtime for the new area of heterogeneity-enabled clouds. In the **short term**, I will investigate:

Interactive Human-in-the-Loop Design Space Exploration. Source code performance tuning is a key component of HLS programming, known as design space exploration. I propose to investigate *a guided exploration to expedite the tuning process.* The hypothesis is that an interactive visualization that lets a human designer decide the next set of configuration choices may outperform a fully automatic exploration. The proposed research will solicit users' guidance for early rejection of design choices, demonstrating an exciting interdisciplinary direction between SE, HCI, and hardware.

Offloading Runtime for Heterogeneous Applications. Offloading arbitrary inputs to hardware accelerators can lead to runtime errors and performance degradation. I propose to investigate *runtime selective offloading with test generation and guard synthesis.* The hypothesis is that correctness, performance, and resources are input data dependent. To this end, we will generate inputs to expose different features (*e.g.,* sparsity and graph connectivity), extract predicates on those input features as guard conditions, and selectively offload real-world inputs to hardware for ensuring correctness and improving performance.

Application-Level Interactive Performance Debugging. Performance debugging is a daunting task in heterogeneous systems—lacking tools to reflect unexpected stalls at hardware level to application code. I propose to investigate *a high-level interactive performance debugging tool* with on-demand code instrumentation and performance monitoring. The proposed tool lets users formulate a performance query in a domain-specific language and automatically inserts corresponding code for simulation and monitoring.

The success of my research will significantly reduce the cost of developing, debugging, and optimizing heterogeneous applications for typical software developers.

References

[1] **Qian Zhang**, Jiyuan Wang, and Miryung Kim. "HeteroFuzz: Fuzz Testing to Detect Platform Dependent Divergence for Heterogeneous Applications". In: *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE)*. Athens, Greece, 2021.

[2] Jiyuan Wang, **Qian Zhang**, Guoqing Harry Xu, and Miryung Kim. "QDiff: Differential Testing of Quantum Programming Frameworks". In: *Proceedings of The 36th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. 2021.

[3] **Qian Zhang**, Jiyuan Wang, Muhammad Ali Gulzar, Rohan Padhye, and Miryung Kim. "BigFuzz: Efficient Fuzz Testing for Data Analytics Using Framework Abstraction". In: *Proceedings of The 35th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. 2020.

[4] **Qian Zhang**, Jiyuan Wang, Muhammad Ali Gulzar, Rohan Padhye, and Miryung Kim. "Efficient Fuzz Testing for Apache Spark Using Framework Abstraction". In: *2021 IEEE/ACM 43rd International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*. 2021.

[5] Jason Lau*, Aishwarya Sivaraman*, **Qian Zhang***, Muhammad Ali Gulzar, Jason Cong, and Miryung Kim. "HeteroRefactor: refactoring for heterogeneous computing with FPGA". In: *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering (ICSE)*. (***Equal first authors)**. 2020.

[6] **Qian Zhang**, Jiyuan Wang, Guoqing Harry Xu, and Miryung Kim. "HeteroGen: Transpiling C to Heterogeneous HLS Code with Automated Test Generation and Program Repair". In: *Proceedings of the 27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*. 2022.

[7] **Qian Zhang** and Qiang Xu. "Approxit: A quality management framework of approximate computing for iterative methods". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)* (2020).

[8] Zhaohui Zhang, Haichao Zhu, and **Qian Zhang**. "ARSketch: Sketch-Based User Interface for Augmented Reality Glasses". In: *Proceedings of the 28th ACM International Conference on Multimedia (ACM MM)*. 2020.

[9] **Qian Zhang**, Feng Yuan, Rong Ye, and Qiang Xu. "Approxit: An approximate computing framework for iterative methods". In: *Proceedings of the 51st Annual Design Automation Conference (DAC)*. 2014.

[10] **Qian Zhang**, Ting Wang, Ye Tian, Feng Yuan, and Qiang Xu. "ApproxANN: An approximate computing framework for artificial neural network". In: *2015 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. 2015.

[11] **Qian Zhang**, Ye Tian, Ting Wang, Feng Yuan, and Qiang Xu. "Approxeigen: An approximate computing technique for large-scale eigen-decomposition". In: *2015 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. 2015.

[12] Ye Tian, **Qian Zhang**, Ting Wang, and Qiang Xu. "Lookup table allocation for approximate computing with memory under quality constraints". In: *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. 2018.

[13] Ye Tian, Ting Wang, **Qian Zhang**, and Qiang Xu. "ApproxLUT: A novel approximate lookup table-based accelerator". In: *2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. 2017.

[14] Ye Tian, **Qian Zhang**, Ting Wang, Feng Yuan, and Qiang Xu. "ApproxMA: Approximate memory access for dynamic precision scaling". In: *Proceedings of the 25th edition on Great Lakes Symposium on VLSI (GVLSI)*. 2015.

[15] Ting Wang, **Qian Zhang**, and Qiang Xu. "ApproxQA: A unified quality assurance framework for approximate computing". In: *Design, Automation & Test in Europe Conference & Exhibition (DATE)*. 2017.

[16] Ting Wang, **Qian Zhang**, Nam Sung Kim, and Qiang Xu. "On effective and efficient quality management for approximate computing". In: *Proceedings of the 2016 International Symposium on Low Power Electronics and Design (ISLPED)*. 2016.

[17] **Qian Zhang**, Ting Wang, and Qiang Xu. "On resilient task allocation and scheduling with uncertain quality checkers". In: *2017 22nd Asia and South Pacific Design Automation Conference (ASP-DAC)*. 2017.

[18] Juan Yi, **Qian Zhang**, Ye Tian, Ting Wang, Weichen Liu, Edwin H-M Sha, and Qiang Xu. "ApproxMap: On task allocation and scheduling for resilient applications". In: *2016 21st Asia and South Pacific Design Automation Conference (ASP-DAC)*. 2016.

[19] BigFuzz. `https://github.com/UCLA-SEAL/BigFuzz`. 2021.

[20] HeteroFuzz. `https://github.com/UCLA-SEAL/HeteroFuzz`. 2021.

[21] HeteroRefactor. `https://github.com/UCLA-SEAL/HeteroRefactor`. 2021.

[22] QDiff. `https://github.com/UCLA-SEAL/QDiff`. 2021.