# Yufei Shi

412–251–8844 contact@shiyufei.com yufei-shi yshi02 shiyufei.com

## Education

**Carnegie Mellon University** . . . . . **Pittsburgh, PA**
B.S. in Electrical and Computer Engineering, GPA: 3.67 . . . . . Expected May 2024

- **Teaching assistant**: Intro to Computer Systems (4 semesters); Computer Systems and HW-SW Interface (1 semester)
- **Courses**: Intro to Computer Architecture, Parallel Computer Architecture & Programming, OS Design & Implementation

## Experience

**Carnegie Mellon University, ABSTRACT Research Group** . . . . . **Pittsburgh, PA**
Research Assistant . . . . . Jan 2023 — Present

- Conducted research on the **memory consistency model** ⧗ and **asynchronous programming model** ❝ of Coarse-Grained Reconfigurable Architecture (CGRA), a hardware dataflow architecture.

⧗❝ Added support for lightweight threads in the dataflow execution simulator by extending the ISA with a new processing element, implementing thread dispatch synchronization, and resolving legacy bugs within the simulator.

⧗ Analyzed CGRA's memory consistency model by developing testing programs with diverse memory access patterns, tracing their execution on the simulator and identifying violations of sequential consistency in the trace.

⧗ Designed a HW-SW solution for **enforcing sequential consistent execution** of pipelined dataflow programs.

❝ Contributed to the design of a programming language for expressing parallelism in CGRA applications by converting algorithms with irregular memory access patterns into pipelined, work-queue-based implementations.

❝ Contributed to the implementation of a state-machine for synchronizing asynchronously-dispatched task inputs.

## Projects

**Unix-like x86 OS Kernel with Thread Library** | C, Simics . . . . . **Sep-Dec 2023**

- Designed and implemented a Unix-like x86 OS Kernel that supports preemptive multitasking, multiple memory address spaces, and a set of important system calls as well as device drivers for timer, keyboard, and console.
- Implemented kernel thread context switching, scheduler with multiple scheduling queues and prioritized round-robin scheduling algorithm, and kernel-level synchronization primitives to realize **preemptive multitasking**.
- Implemented standard 32-bit x86 two-level paged **virtual memory** that supports zero-filled on-demand paging.
- Implemented **task/thread interface** that works with my implementation of a user-level POSIX-like thread library.

**Parallel Mesh Collision Detector** | C++, CUDA, GDB, OpenMP, Open3D . . . . . **Apr-May 2023**

- Developed a parallel algorithm to **accurately determine the minimum distances between convex meshes**, which can be used to detect potential collisions between objects in real-time for robotic motion planning tasks.
- Implemented Gilbert-Johnson-Keerthi algorithm and optimized it by parallelizing its support function in CUDA.
- Developed a simulation and visualization framework for demonstrating the algorithm running in complex scenes.
- Achieved a **20x speedup** over the baseline by combining **CUDA** and **OpenMP** with additional optimizations.

**In-Order 2-Way Superscalar RISC-V Processor** | C++, SystemVerilog, Synopsys VCS & DC . . . . . **Jan-Apr 2023**

- Designed and implemented an RV32I processor featuring a 2-way superscalar in-order 5-stage pipeline.
- Implemented **branch prediction** and **data forwarding** capabilities to alleviate data hazards in the 5-stage pipeline, and ensured their integration with the final superscalar pipeline to maximize IPC of the processor.
- Conducted **timing and power optimizations** over design iterations by iteratively analyzing synthesis report and making adjustments to the design; achieved a 15% IPS increase while reducing power consumption by 80%.

## Skills

**Programming Languages**: C, C++, Python, Rust, Shell, x86 Assembly, Common Lisp (a little bit)
**Hardware Design Tools**: SystemVerilog, Synopsys VCS, Synopsys Design Compiler, Intel Quartus, Fusion 360
**Developer Tools**: GDB, Git, Make, Valgrind, awk, vim, Regex, Conda, Various Linux Distros, Pin Tool, gem5
**Technologies**: MATLAB, SOLIDWORKS, NumPy, Matplotlib, OpenMP, MPI, CUDA, OpenGL, HTML, LaTeX