Aditya Reddy

adityareddy400@gmail.com | github.com/Aditya-1020 | aditya-1020.github.io

Education

Manipal University Jaipur **Aug 2024 – May 2028**

Bachelor of Technology in Electronics and Communication Engineering *Jaipur, India*

- **Relevant Coursework:** Digital Logic Design, Circuit Analysis, Signals & Systems

Technical Skills

- **Hardware Description Languages:** Verilog, SystemVerilog
- **EDA & Simulation Tools:** Vivado, Verilator, GTKWave, LTspice, Vivado CLI
- **Programming:** C, C++, Python
- **Tools/Frameworks:** Git, Make, Linux, GCC, RISC-V Toolchain, Docker, LaTeX

Projects

AdiRiscV — RISC-V RV32IM Processor Core | SystemVerilog, Vivado, GTKWave

- Designed synthesizable single-cycle and 5-stage pipelined RISC-V cores implementing the full RV32IM instruction set
- Built complete hazard management with EX→EX and MEM→EX forwarding, load-use stall detection, and pipeline bubble insertion
- Implemented dynamic branch prediction with 64-entry BTB, G-share PHT using 2-bit saturating counters, and return address stack
- Integrated hardware multiply and a 32-cycle non-restoring signed/unsigned divider for M-extension arithmetic
- Developed assertion-driven verification across 8 test suites achieving 100% pass rate and full functional coverage
- Added performance counters tracking cycles, IPC, branches, and stalls for microarchitectural evaluation

Low Latency Trading Gateway | C++, FIX Protocol, Lock-Free Programming **GitHub Link**

- Engineered high-performance trading gateway using lock-free queues and memory pools, reducing synchronization bottlenecks by 85%
- Implemented zero-copy FIX protocol parsing with state machines, achieving sub-2μs parsing with 1.4μs average latency
- Optimized tick-to-trade latency to 10–15μs average with 2.8μs minimum, sustaining 400+ messages at 98% execution rate
- Applied CPU pinning and cache-aligned memory allocation to eliminate scheduling delays and improve data locality

HTTP/1.1 Web Server | C, POSIX Sockets, Multithreading **GitHub Link**

- Architected custom HTTP/1.1 server in C with raw POSIX sockets and thread pool, eliminating external library dependencies
- Implemented complete HTTP request parsing, static file serving, and MIME type detection with directory traversal protection
- Achieved 11.5K+ requests/sec throughput with 8.7ms average latency under 100 concurrent clients using ApacheBench
- Designed connection pooling and keep-alive mechanisms to reduce TCP handshake overhead by 40%

Activities & Interests

- Volunteered at Bison Asha School for Special Needs, supporting educational activities and organizing inclusive events (2022 – 2024)