Aaditya Suri

236-990-5666 | aadityasuri01@gmail.com | Linkedin | Github | Website

Education

University of British Columbia — Vancouver, BC
Bachelor of Applied Science – Computer Engineering — **Expected May 2025**

Technical Skills

Languages: C, C++, Java, Python, SQL, CUDA, SystemVerilog, Tcl
Frameworks: PyTorch, PyTest, JUnit, NumPy, Flask, Guice, Qt, Postgres, OpenMP
Tools: Git, Bash, Docker, Maven, CMake, Quartus, Modelsim, Innovus
Concepts: Deep Learning, Computer Architecture, OOP, Operating Systems, REST, Databases, RTL Design

Work Experience

Software Engineer Intern — **May 2024 – August 2024**
Citi — *Mississauga, ON*

- Contributed towards the risk team's core library to build a scalable **distributed cache** using **Apache Ignite** in-memory database, leveraging **Java generics** to ensure versatile data compatibility.
- Achieved a 20% speedup in cache request processing by migrating the legacy data access service to using **Ignite scan queries**, significantly enhancing cache access times across 10+ projects supported by the core library.
- Developed a **Splunk HEC** log handler for **Python**, utilizing a **multithreaded** model to effectively manage network latency and minimize performance impacts on applications. Deployed as an internal PyPi package using **Jenkins**.
- Enhanced code quality by adding 50+ **unit tests** using a **TDD** approach with **Guice** and **JUnit**, achieving 90% class coverage and ensuring robust test coverage.

Software Engineer Intern — **April 2023 – December 2023**
Netgear — *Richmond, BC*

- Developed 30+ **Python** library methods for an automated hotspot device validation framework using the **SCPI** protocol to interface with CMX500 call boxes, enhancing RF testing capabilities with new **LTE** and **NR 3GPP** functionalities.
- Reduced calibration time by up to 60% and eliminated path-loss data spikes by completely redesigning the instrument calibration software for multiple call boxes, power meters and signal generators.
- Wrote an extendable library for **PyQT** using **Matplotlib** and **multiprocessing** to enable asynchronous graph rendering.
- Optimized the BLER test procedure, reducing completion time by 75% and preventing timeouts by developing an adaptive algorithm based on AIMD congestion handling protocol.
- Streamlined deployment and updates across lab computers by creating a **PyInstaller** script to automatically parse project requirements and install executables, simplifying maintenance and setup.

Physical Design Engineer Intern — **January 2023 – April 2023**
Microchip Technology Inc. — *Burnaby, BC*

- Configured and debugged multiple **Tcl**, **Bash** and **Perl** scripts in a Unix environment as part of the complete **PnR** flow for TSMC and UMC tape-outs.
- Performed Logical Equivalency Checking (**Conformal LEC**) and Layout vs. Schematic (**LVS**) checks for 28nm and 16nm designs, while also contributing to Engineering Change Order (**ECO**) processes to ensure design integrity and functionality.
- Contributed to an internal **Python** GUI for analysis of Cadence tool logs by fixing multiple **Regex** and GUI bugs.

Projects

Smolar | *C, OpenMP, Python*

- Contributed to an open source project to build a numpy like vector library from scratch in **C**.
- Implemented several methods like N-d matrix multiplication, exploiting cache locality and **OpenMP** threads for speedup.
- Added Python bindings for the **C** functions using the **ctypes** library to ease unit testing.

Matmult accelerator | *Python, Tcl, SystemVerilog*

- Working on a tensor processing unit design inspired by Google's TPU chip for accelerating **machine learning** inference.
- Implemented a systolic array architecture in **SystemVerilog** for General Matrix Multiply (GeMM) speedup.
- Created an extensive self-checking **testbench** using **SystemVerilog** and **Tcl** scripts to verify the design.

Vector Search Engine | *Python, C++, Pandas, NumPy, SQL*

- Designed and developed a search engine based on the **TF-IDF** algorithm and cosine similarity.
- Made use of **NumPy** and **Pandas** libraries for extremely fast mathematical calculations.
- Profiled the code to identify performance bottlenecks and reduce indexing time by 90% with **C++** using **PyBind11**.

Conditional PixelCNN++ | *Python, PyTorch, NumPy, Deep Learning*

- Implemented a custom **PyTorch** pipeline to fuse labels with image pixel data in an existing **PixelCNN++** model to achieve conditional image generation.
- Trained the **generative model** for classification and achieved more than 80% accuracy on the classification test set.
- Experimented with multiple model architectures to achieve satisfactory image generation. Logged all training runs using a Weights and Biases dashboard.

Reliable Transport Protocol | *C, TCP. UDP, Networking*

- Designed a performant **TCP** like transport layer protocol over **UDP** to guarantee packet acknowledgements.
- Implemented custom data structures in **C** to maximize throughput.
- Achieved 90% bandwidth utilization compared to the standard TCP implementation using several flow control methods.

OS161 | *C, Assembly, Operating Systems*

- Implemented the OS file system and syscalls like open, close, read, write, fork, waitpid, exec, etc. using **C** and **Assembly**.
- Added support to the OS to handle concurrent threads using **locks** and **semaphores**.
- Implemented the entire Virtual memory system for the operating system.

MaskMatcher | *Python, PyTorch, Raspberry Pi*

- Developed a **machine learning** model using a **Convolutional Neural Network (CNN)** for classification of faces wearing masks using the **PyTorch** library with over 80% accuracy. Optimized to run smoothly on **Raspberry Pi**.
- Used **OpenCV** and **Haar Cascades** to determine confidence interval of face mask detection.
- Implemented multithreading on a **Raspberry Pi** to capture video from a webcam, run the **machine learning** model and live stream the video to a website simultaneously.

Wikipedia Mediator Server | *Java, Multithreading, Web Servers, JSON*

- Setup a **multithreaded** Java web server to handle concurrent JSON requests from multiple clients.
- Integrated the **JWiki API** into program to serve data from Wikipedia.
- Utilized a **BFS algorithm** to find minimum number of Wikipedia links required to get to another page, beating existing implementations in terms of speed and accuracy.

ARC4 Decryption Circuit | *SystemVerilog, ModelSim, Intel Quartus*

- Developed a Decryption circuit for **ARC4 cipher** and implemented it on the **DE1-SoC FPGA** using **Intel Quartus**.
- Simulated and debugged hardware modules in **ModelSim** by writing extensive Testbenches using **Verilog**.
- Achieved up to **40% reduction** in decryption completion time by adding multiple decryption modules to crack the cipher in parallel.

Volunteer Experience

Software Engineer
UBC Thunderbots

- Contributed towards updating **Python** and **C++17** bindings using **PyBind11**.
- Expanded internal C++ geometry library methods.
- Worked on multithreading existing HRVO algorithm implementation for increasing the path sampling rate for robot navigation and obstacle avoidance.

Firmware Engineer
UBC Open Robotics

- Used **C** to create and test the **firmware** and electrical design of a gripper for the RoboCup@Home project
- Calibrated various **sensors** to work with other electrical and mechanical components such as **motors** and **actuators**
- Programmed **microcontrollers** for making the gripper work in sync with the arms of the robot and the software