**Xingyu (Tom) Wang**

Vancouver, BC || 1 (604) 388-5164 || **fortily@student.ubc.ca|| personal website: https://luckunately.github.io/**

# EDUCATION

**University of British Columbia** Vancouver, Canada

*Bachelor of Applied Science in Computer Engineering, CGPA: 4/4.33 08/2021 – 09/2025(expected)*

* + Related Courses: **Algorithm and Data structure, Machine Learning, Software** **Construction, firmware programming**, Computing System, Computer Architecture, Digital and Microsystem design, Error Control Coding,

# SKILLS SUMMARY

|  |  |  |  |
| --- | --- | --- | --- |
| **Programming Languages** | **Engineering skills** | **Programming skills** | |
| * Java, Python, C, C++ * System Verilog, Assembly * Bash, Makefile * Latex, Markdown | * Algorithms and Data structures * Microprocessor and system buses * Computer Architecture * Cache and Page Prefetching | * Software Hardware Interface * Embedded Programming * Deep Learning Algorithms * Git, GDB, Linux environment |

# WORK EXPERIENCE

**Full-time Student Research Assistant** *May, 2024 – Present*

*UBC Systopia Lab Vancouver, BC*

* + **Aim**: Investigate the applicability of the Learned Relaxed Belady (LRB) machine-learning model for cache and

page pre-fetching.

* + **Methods**: Collect SPEC 2017 and GAP traces with PIN and fltrace, and apply machine learning methods to prefetch cacheline/page
  + **Progress**: Tune LSTM model Add Attention Layer. Analyze trace. Experiment with heuristic methods. Hardware-Software Codesign for Prefetching
  + **Supervision under**: Shaurya Patel, Prof. Alexandra Fedorova.

# PROJECTS

**Microsystem Design with Microprocessor** *Jan 2024 - April 2024*

* + Build memory, data bus, various I/O around a M68K CPU on FPGA. Interact with CPU using **embedded C programming**.
  + Implemented components include DRAM controller, Cache Controller, SPI, Canbus, I2C, ADC/DAC, and **Simple RTOS** usage with **multi-threading** and **priority interrupts**.
  + Integrate the above components with VGA and Voice modules, and **map addresses** accordingly both in RTL design and C programming to produce a Tetris game with the M68K CPU.

# IoT: Client and Server interaction *Nov 2023 – Dec 2023*

* + Summon multiple processes/threads to mimic client-server behaviours. Send packets between multiple clients and servers through the internet and process requests concurrently while maintaining coherence.
  + **Concurrency, multi-threading**, software development, debugging, collaboration and teamwork.
  + Work done in Java.

# Simple shell program *March 2024*

* + Interactive shell executable implemented in C on Linux server. Implemented common shell command execution, kill (with or without core dump), sleep and resume processes, error handling and process management.
  + **Linux API**, **signal handler**, concurrent management, gcc, makefile and gdb.

# Supervised Learning on Audio Files *Nov 2023 – Dec 2023*

* + Collect Audio files, and process with Fourier Transform to get frequency data from Audio waveform. Apply PCA to reduce the dimension. Label data
  + Supervised learning with **Support Vector Machine** and **Neural Network**, comparing the performance, memory usage, and efficiency of training and predicting.

# AWARDS

* + Dean’s Honors List 2021 - 2024
  + NSERC Awards May 2024