Week 1

Codefest: Warm-up

ECE 410/510

Spring 2025

Before you start

1. If you don't have a Github yet, create one where you'll be hosting all your code.
2. Make the repository public so that your peers can access your code.
3. Decide what platform you want to use do document everything you produce in this class. The platform of your choice must be public and accessible to your peers.
    a. Blog (e.g., Wix, Wordpress, Squarespace, GoogleSites)
    b. Technical reports on Github
    c. Github wiki
    d. …
4. **Submit your Github and platform choice(s) on Canvas by Sun, Apr 6, 11:59pm.**

During the codefest:

- Slack will be our main collaborative platform (at least for now).
- Post questions, solutions, insights in the #codefests channel.
- Present what you've got, whether it works or not.

Challenge #4

Learning goals:

- Experiment with LLM-assisted chip design
- Do "vibe coding" and experience the problems associated with it.
- Install and test the entire workflow/toolchain.

Tasks:

1. Replicate the Johns Hopkins paper:
    a. **Designing Silicon Brains using LLM: Leveraging ChatGPT for Automated Description of a Spiking Neuron Array**
    b. https://arxiv.org/abs/2402.10920
2. Use your favorite LLM.
3. Experiment with the queries and see what happens.
4. Keep track of all the queries you made that got you to the finished result.
5. If you want to go all the way down to the ASIC, you can use OpenLane (https://www.zerotoasiccourse.com/terminology/openlane) to convert HDL into GDS (used for ASICs).
6. Compare the results of your version with their paper.
7. Can you think of any improvements to their solution?
8. Can you do a design with a RLU instead if a LIF neuron? https://en.wikipedia.org/wiki/Rectifier_(neural_networks). Or a Hodgkin–Huxley neuron model?
9. Document your results and findings carefully. What did you learn?

Challenge #5

Learning goals:

- Learn how to analyze and profile AI/ML, and other workloads, e.g., written in Python
- Learn how to identify bottlenecks and parallelism
- Learn how to think about candidate execution architectures.
- Do "vibe coding" and experience the problems associated with it.

Tasks:

1. Pick 3 different Python programs/workloads. E.g.,
    a. Differential equation solver
    b. Convolutional neural network
    c. Traveling Salesman Problem (TSP)
    d. Quicksort
    e. Matrix multiplication
    f. A cryptography algorithm, e.g., AES
    g. …
2. Either write your own code (probably not enough time), download some code, or ask your LLM to generate examples.
3. Compile the code into Python bytecode. Ask your LLM how to do that. Or look it up. Hint: `py_compile`.
4. Disassemble the bytecode and look at the instructions. Hint: `dis`
1. Can you guess what virtual machine Python uses just by looking at the bytecode?
2. How many arithmetic instructions are there? Hint: http://vega.lpl.arizona.edu/python/lib/bytecodes.html
3. Write a script that counts the number of each instruction. Hint: ask your LLM.
4. Compare the instruction distribution for your 3 workloads.
5. Use a profiler to measure the execution time and resource usage of your codes. Hint: `cProfile`. Hint: `snakeviz` allows for interactive visualization.
6. Ask your LLM to write you some code to analyze the algorithmic structure and data dependencies of your code to identify potential parallelism.
7. Now, knowing all these details, what instruction architectures would you build for each of these workloads?
10. Document all your findings and insights carefully. What did you learn?