**Recommendation Letter**

This week, I explored how computation can go beyond traditional processors. I skimmed parts of the IEEE IRDS “Beyond CMOS” roadmap, focusing on new device technologies, since the full 200+ page document was too long to finish. I also read a paper on physical systems that solve differential equations inherently, which helped me think differently about how problems can be solved using nature’s behaviour. For the codefest, I used ChatGPT to replicate a spiking neuron design from a Johns Hopkins paper and experimented with other neuron types like RLU and Hodgkin-Huxley. I also profiled different Python workloads to find performance bottlenecks and potential for parallelism. This helped me understand how software and hardware interact, and how AI tools can assist in design.

In Week 2, I began laying the groundwork for my main project on designing a custom chiplet to accelerate parts of an AI algorithm. I started profiling my chosen code to find bottlenecks and understand which parts might benefit from hardware acceleration. For the weekly challenge, I implemented a simple perceptron that learns NAND and XOR functions using a sigmoid activation and the perceptron learning rule. I also started working on a multi-layer perceptron and explored how backpropagation works. This helped me connect basic neural network concepts to real hardware design and gave me direction for developing my accelerator.

This week, I focused on the FrozenLake Q-learning example to identify bottlenecks and test hardware acceleration. I chose the Q-value update as the key part to move to hardware and began modeling it using PyMTL. I also benchmarked a GPU-accelerated version of the Python code to compare performance. This gave me a clearer idea of where hardware can make a real difference.

This week, I wrote my first SystemVerilog module to implement the Q-value update logic from the FrozenLake example and simulated it using EDA Playground. I also explored GPU programming by benchmarking CUDA-based SAXPY and Fibonacci implementations. Comparing them with sequential versions helped me understand parallelism and performance scaling better. These tasks improved both my HDL coding and GPU profiling skills.