ECE 410 Project

Heilmeier Questions:

1. **What are you trying to do? Articulate your objectives using absolutely no jargon.**
   I'm trying to design and implement a custom hardware accelerator chiplet that speeds up the "Q-learning" algorithm utilized in the FrozenLake problem. Essentially, the FrozenLake problem searches for the optimal path through a "frozen lake" while avoiding the holes present in the surface. This is done through iterating through loops to see what works and what doesn't.

2. **How is it done today, and what are the limits of current practice?**
   Generally, the algorithm for this problem is implemented using Python/C/C++ via standard CPU. Each training cycle loops sequentially through the states, queries the Q-table, computes any updates, and writes back to the table.
   As far as limits are concerned, standard CPU cache architecture is not optimal for this use case. Similarly, Python and C/C++ slow the state action table down due to overhead restrictions. Also, even though the environment is small and somewhat straightforward, rapid iteration is difficult as running high cycle counts can take a long time.

3. **What is new in your approach and why do you think it will be successful?**
   My approach aims to use a Verilog RTL accelerator with a pipelined datapath for the Q-table, an ALU for the updates, and an LFSR for random action selection. Compared to the standard software implementation, the operations are parallelized in hardware, which should result in considerable speedup over single-threaded CPU performance.

4. **Who cares? If you are successful, what difference will it make?**
   If successful, this project proves that even small-scale discrete-state RL can be hardware pipelined. Future students can interactively explore hardware/software design entirely in the simulation, rather than needing actual implemented hardware to check things. Potentially, this kind of rapid prototyping could help on-device learning for robotics/IoT stuff, without needing physical boards for hardware.

5. **What are the risks?**
   The biggest risk is probably the potential for limited scope. A tiny FrozenLake accelerator might not generalize to larger problems.

6. **How much will it cost?**
   Assuming all tooling (software, pretty much) is open source, then there are no corresponding license fees, for zero cost. Though, to run the simulation, the more computer power the better, for the sake of an efficient and timely workflow.

7. **How long will it take?**
   CPU profiling - 1wk
   RTL design + unit testbench - 2wks
   Cycle-accurate sim runs - 2wks
   Synthesis + power/area estimation - 2wks
   Integration + comparative analysis - 1wk
   It is extremely likely that it will take longer than this, and even longer than the term allows for for a properly documented, efficient, clean final product.

8. **What are the mid-term and final "exams" to check for success?**
   Mid-term: RTL sim demonstrates correct Q-table, 1k training cycle count run vs. standard single-thread CPU baseline test.
   Final: Synthesis report with utilization and power estimates, sim confirming timing and cycle counts, summary table for speed and energy use between the simulated accelerator vs. the standard CPU.