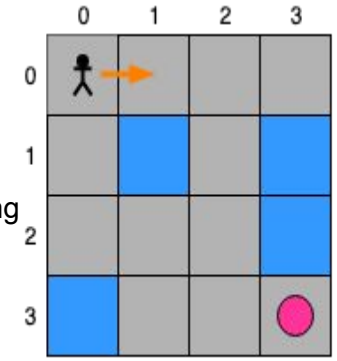
FrozenLake Optimization



Legends:

A thick and safe layer of ice which you can walk over

A hole in the ice. Fall here and you're dead

The frisbee. Be the hero and get it back

You

What are you trying to do?

- Build a Verilog RTL accelerator that runs Q-learning on the FrozenLake MDP in hardware, to find the optimal path through the grid while avoiding holes.

How have others implemented and/or accelerated this algorithm?

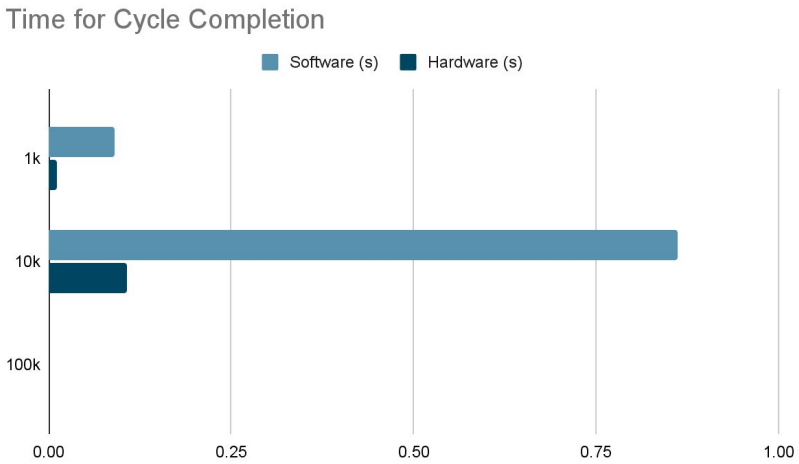
- Standard Python/C++ loops on CPU: sequentially read/write a Q-table, compute updates, select actions with software RNG.
- Slow for high iteration counts.

What are you doing differently/better/etc.?

- Fully pipelined datapath: parallel state-fetch, ALU update, table-write stages
- LFSR-based action selection in hardware
  - Exploits spatial & temporal parallelism for an expected ~8× speedup over single-thread CPU (in theory, we'll see if I actually achieve that)

What have you accomplished so far?

- RTL design & unit testbench for all pipeline stages
- 1k-cycle simulation vs. CPU baseline: correct Q-table updates



What will you do next and what remains to be done until you can declare success?

- Run full-scale tests (100k cycles?)
- Synthesis & power/area estimates
- End-to-end performance & energy comparison vs. software