1. The FIFOs we use in our mesh have only a depth of 4. What problems can that cause? (they fill, and then we must bounce). Can you think of any way to make the mesh still work other than infinite-size FIFOs? (Yes, bouncing).

* We need a priority control logic. When a FIFO is full, that FIFO will be prioritized. So we can evacuate the full FIFO first to prevent full FIFO stuck.
* If the FIFO is full, then the packets will be directed to a near mesh stop

1. We chose vertical-first routing, but mentioned that horizontal-first would work equally well. Can you think of any reasons why you may want the choice to be dynamic (i.e., the mesh decides on the fly which way to route), or even to allow alternating back and forth more than once dynamically?

* If the following FIFO is stuck or broken, the package will reroute to the destiny quicker.
* The packets can be routed into two paths, reduce the possibility that the packets get stuck in the traffic.
* The packets can get to the destination faster by choosing a shorter path. (back and forth)

1. Consider verifying single-precision (i.e., 32-bit) floating-point add exhaustively, and assume you have a very fast model that can run 1G tests/second, and furthermore that you have a farm of 1000 such machines. How long would this exhaustive verification take? Is it practical? Is it the best use of CPU time?

(232\*232) / (210\*230)=194.18 days

1G tests/second= 230tests/second, 1000=210

Theoretically, it is practical. But you won’t want to spend 194 days testing add function only.

No

1. Read a discussion of the Pentium FPU bug at https://daviddeley.com/pentbug/. Then first, describe the bug. Second, what are your odds of hitting it with any of our RCG knobs? Finally, how would you create content if you suspected a similar bug could be present on a new FPU?