1. Describe the knobs that you have implemented.

* Packets per cycle can be adjusted by testers from 1 to 5.
* Testers can adjust whether the virtual environment will take data from meshes by inputting 0 (always take data), 1 (take data randomly) or any other number (take data half of the time).

1. A bit of reverse engineering for you – describe just how we now check that the mesh has correctly delivered all packets.

* If all packets arrive at the destination correctly, this information is displayed directly.
* If not, find out how many and which packets are lost, see if there are similarity between those packets. If the answer is yes, hopefully we can find the reasons directly.
* If not, find the displayed information (not found, wrong destination or just not all packets received.) and check the waveforms.
* Focus on the start mesh, turning mesh and the destination, check the waveforms of FIFOs, ‘ring\_in’s and ‘ring\_out’s.
* Usually, the bugs are at those two mesh stops, if not go along the path and check the mesh one by one.

1. Your RCG delivers random or constrained-random packets to the mesh. One common problem with random code is that it may wind up being illegal. Are there any combinations of packets that are illegal in our case? I.e., any that we simply cannot present to the mesh? If so, how does your RCG avoid them?

* RCG may generate packets with the same departure location. In this case, there will be only one package survive.