Mesh lab #4 – checking and improving coverage

How to work as a group

This lab involves coding, simulating and writing a report. You can work as a group and only turn in one set of files.

The goal of working as a team is to mirror the way you will most likely work in a real job. However, my expectation is that everyone in the group learns the code and runs the simulations. The goal is *not* to have only one person learn the material 😊.

Goals of the lab

In this lab, we'll

- Not touch our mesh hardware at all.
- Improve our testbench by checking *functional coverage* to measure how well it tests the mesh.
- Increase the coverage if needed by choosing the appropriate RCG parameters or by improving the RCG.

Big-picture instructions

- Reuse all of your files from the previous lab. Don't touch any of the hardware files unless you want to, since this lab only changes the testbench code and not the DUT. Our improved version of *tb_mesh_3.sv* will now be called *tb_mesh_4.sv*.
- Add functional-coverage checking to your code as detailed below.
- Come up with some knob setting(s) to get full coverage of our desired coverage points.
- Turn in your final *tb_mesh_4.sv*, as well as a .pdf with answers to the questions below.

What is functional coverage?

We talked in class about different types of test coverage. Functional coverage is one of the most important types of coverage. The nice thing about it is that it kind of means whatever you want it to mean 😊. Perhaps a bit more precisely, achieving good functional coverage means that you have tested the signals that you believe are most important to the functionality of the design.

The wonderful thing about functional coverage is that we get to decide for ourselves what is important! Of course, there are other types of coverage to check also.

As we discussed in class, we would like to ensure that we have covered the following points:

- Each of of the five select lines *vert_sel_pass, vert_sel_me, hori_sel_pass, hori_sel_me* and *hori_sel_turn* must get asserted at some point. In our 4x4 mesh, we don't need all 16 (e.g.,) *vert_sel_pass* lines to go high – but at least one of them must. This may be fairly easy to do.
- Each of the three FIFOs (the driver, horizontal-receiver and vertical-receiver) must fill up at some point. Again, in a 4x4 mesh we don't need all 16 driver FIFOs to fill, but at least one of the 16 must. This may be a bit harder!

Why is getting good functional coverage so important? Because it's one more factor leading to you knowing when your design "probably works."

Using covergroups to check functional coverage

The tool we'll use to check coverage is called a *covergroup*. You can find more detail in Spear chapter 9, or in chapter 19 of the SystemVerilog LRM.

You could create a covergroup in your testbench with code like:

```
covergroup mesh_cov (ref logic vert_sel_pass, ...) @(negedge clk);
      vsp: coverpoint vert_sel_pass;
      other coverpoints and options...;
endgroup
```

This code merely defines the covergroup; you must still instantiate it once for each mesh stop (using a doubly-nested loop). You should instantiate a 2D array of covergroups *cov*[*i*][*j*].

Once the simulation is done, you can check coverage on (e.g.,) *vert_sel_pass* with code such as

```
perc = cov[0][0].vsp.get_inst_coverage(n_bins_cov, n_bins);
$display ("VSP coverage = %0d / %0d bins", n_bins_cov, n_bins);
```

This will tell you the coverage on *vert_sel_pass* in mesh stop #00. Another flavor is

```
perc = cov[0][0].vsp.get_coverage(n_bins_cov, n_bins);
```

As discussed in class, both may be useful; you can pick what works best for you.

Improving functional coverage

Hopefully it will be easy for you to get coverage of all five select lines. It may be harder to fill all of the FIFOs. If so, then your next job is to improve the coverage. Perhaps it will be as simple as choosing different RCG parameters, or you may have to improve your RCG.

It's almost always best to think before you code. If you have a coverage hole, then think about why that area is not being covered by your current tests. Think about the mesh architecture, look at debug logs and waveforms as needed, and come up with an improved strategy. You may well need multiple RCG runs, each with a different set of parameters, to get full coverage.

Simulator limitations:

Several of the EDA-playground simulators have limitations that are relevant to this lab:

- Cadence Xcelium doesn't support arrays of covergroups. It is possible to work around this, but it's perhaps easier to simply not use Cadence for this lab.
- Both Synposis VCS and Aldec Riviera Pro have issues with using the merged-coverage option for covergroups. Again, you can choose simply to not use these simulators.
- In fact, that leaves only one simulator on EDA playground fully supporting covergroups – Mentor Questa. For that reason, Mentor Questa is hereby designated as the Official Simulator of Mesh Lab #4 😊. (But see question #1 below also).

Questions to answer:

1. The section ***Using covergroups to check functional coverage*** above shows two different styles of asking for coverage. For each of those two examples, how many bins are returned for the given call, and why? (Assume that *merge_instances* is off).
2. One way to find whether a single mesh signal (e.g., *vert_sel_pass*) was ever asserted in any of the mesh stops is to use *merge_instances* mode, as you probably did in your code. Is there any way of achieving this goal without needing *merge_instances* mode? If so, you would have been able to use any of the edaplayground simulators.
3. One strategy for filling the HRx and/or VRx FIFOs is to pick a particular destination mesh stop; say MS22. Then have the other 15 mesh stops all launch packets whose destination is MS22.

With 15 sources targeting one single destination, it seems like MS22 should be swamped very quickly. First, will filling the VRx FIFO with this strategy be easy, difficult or impossible, and why? Second, same question for the HRx FIFO. (Note: assume that the environment accepts packets every cycle).

4. Explain how you were able to get coverage of the FIFO-full signals. What problems did you run into and how did you solve them?

What to turn in:

- Turn in your final *tb_mesh_4.sv*. Make sure the RCG settings you used are set in the code; if you needed multiple runs, then note that and leave one run commented out (I'll run both when I check your code).
- A .pdf with your answers to these questions.
- As usual, just have one person per team turn in work; no need for every individual to turn in something separate.