Exercise 07: ***PyVSC*** Coverage in ***cocotb*** test

Objective: *Introduction to coverage collection using the PyVSC library.*

Task: *Implementation of a coverage collector in the cocotb test exercise using the PyVSC library.*

Locate the *cocotb* exercise in `<ROOT>/exercises/E03_sat_cocotb_test`. Create a file to implement the *coverage collection* with the name `sat_filter_coverage.py`. Inside the file, create a `covergroup` called `covergroup_ssdt`. The `covergroup` must contain a `coverpoint` for the `data`. The `coverpoint` must cover the following bins:

- `data` when is 0;
- `data` when is the maximum value;
- `data` in the range between 0 and the maximum value.

Read more in the *PyVSC* Coverage documentation.

Now, the coverage collector must be introduced in a test. Locate the random test which randomizes the input data (`sat_random_test_pyvsc_rnd`), implemented using the constraints and create the coverage collector implemented. The coverage collector must sample the `out_data` signal when `out_valid` goes high.

The test should generate a coverage report before ending. From `utilities.py`, located in `<ROOT>/exercises/E03_sat_cocotb_test`, import the `create_coverage_report`. The method requires as input the name of the test case, e.g., `create_coverage_report("sat_random_test_pyvsc_rnd`

Look at the coverage results for each test inside `<ROOT>/exercises/E03_sat_cocotb_test/sim_build` directory. Analyze the files `<test-name>_cov.txt` generated. The `PyUCIS-viewer` tool can also be used to visualize the coverage results, by running, e.g.:

```
(.venv) [<username>@<servername> tb]$ pyucis-viewer
sim_build/sat_random_test_pyvsc_rnd_cov.xml
```

Analyze the coverage report. How good are the results for the coverage using the test?