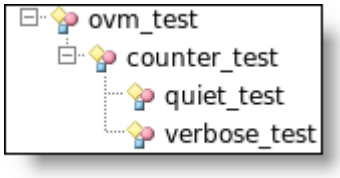
Lab 10: Controlling Reporting

In this lab you will use the OVM reporting features to reduce the output from a test without modifying the components in the environment. Instead, you will modify the `end_of_elaboration()` method in the test.

The counter test bench has three tests:



`counter_test`

The counter test has no `end_of_elaboration()` method. When you run `counter_test` you get 108 OVM messages:

```
# ** Report counts by severity
# OVM_INFO :  108
# OVM_WARNING :    0
# OVM_ERROR :    6
# OVM_FATAL :    0
# ** Report counts by id
# [RNTST]     1
# [run]   113
```

This is the base for the `verbose_test` and the `quiet_test`. Both of these tests extend the `counter_test` and simply add an `end_of_elaboration()` method.

`verbose_test`

The `verbose_test` extends the `counter_test` and adds an `end_of_elaboration()` method:

```
1  class verbose_test extends counter_test;
2
3    `ovm_component_utils(verbose_test)
4
5    function new(string name = "", ovm_component parent);
6      super.new(name, parent);
7    endfunction : new
8
9
10    virtual function void end_of_elaboration();
11       super.end_of_elaboration();
12       env.set_report_verbosity_level_hier(OVM_DEBUG);
13    endfunction : end_of_elaboration
14
15
16 endclass : verbose_test
```

The `end_of_elaboration()` method sets the verbosity to `OVM_DEBUG`. This causes all the agents in the test bench to print debug messages:

```
# OVM_INFO src/driver.svh(27) @ 820: ovm_test_top.env.drv [run] Driver got data: f9 op: load
# OVM_INFO src/monitor.svh(29) @ 831: ovm_test_top.env.mon [run] Monitor got req data: f9 op: load
# OVM_INFO src/monitor.svh(33) @ 831: ovm_test_top.env.mon [run] Monitor got data: f9
# OVM_INFO src/predictor.svh(31) @ 831: ovm_test_top.env.pred [run] Predictor predicted data: f9
# OVM_INFO @ 831: ovm_test_top.env.p_req [run] data: f9 op: load
# OVM_INFO @ 831: ovm_test_top.env.p_rsp [run] data: f9
# OVM_INFO src/comparator.svh(21) @ 831: ovm_test_top.env.comp [run] Comparator Actual: data: f9
# OVM_INFO src/comparator.svh(26) @ 831: ovm_test_top.env.comp [run] Comparator Predicted: data: f9
# OVM_INFO @ 831: ovm_test_top.env.comp [run] passed: data: f9
```

This is useful because you can follow a transaction through the test bench. It creates 269 info messages:

```
# ** Report counts by severity
# OVM_INFO :  269
# OVM_WARNING :    0
# OVM_ERROR :    6
# OVM_FATAL :    0
# ** Report counts by id
# [RNTST]     1
# [run]   274
```

quiet_test

The `quiet_test` also extends the `counter_test`:

```
 1 class quiet_test extends counter_test;
 2
 3    `ovm_component_utils(quiet_test);
 4    int error_file;
 5    function new(string name="", ovm_component parent);
 6       super.new(name, parent);
 7    endfunction : new
 8
 9    virtual function void end_of_elaboration();
10       super.end_of_elaboration();
11       // Please add calls to the OVM reporting tools to reduce the number
12       // of calls with OVM_INFO severity to 1, and to limit the number of
13       // errors to 3.
14       // The error messages should go into a file called error.txt
15    endfunction : end_of_elaboration
16 endclass // tester
```

Your job is to add OVM reporting calls to `end_of_elaboration()` so that the test quits after three errors and there is only one `ovm_info` message. The error messages should not be printed to the screen. Instead, they should be written to a file called "`error.txt`".

```
# --- OVM Report Summary ---
#
# Quit count reached!
# Quit count :          3 of          3
# ** Report counts by severity
# OVM_INFO :    1
# OVM_WARNING :    0
# OVM_ERROR :    3
# OVM_FATAL :    0
# ** Report counts by id
# [RNTST]     1
# [run]     3
```

Running the Test

You can run the test in the student directory with this command:

```
% vsim –c –do –run.do
```

This will run the three tests in the following order:

```
verbose_test
counter_test
quiet_test
```

Each test should create fewer messages than the test before.