

Lab 5: Creating an OVM Test

Now that we've reached the end of Lab 5, you have everything you need to create a test in the OVM. You can hook SystemVerilog code up to an interface, drive signals, and check responses in the interface.

In this lab, we will create an OVM test bench for the counter design that we've used throughout the course. This skill is the basis for more powerful test benches that we'll create in later classes.

In this lab we are going to create a test by extending the `ovm_test` class, and we are going to connect that test to the DUT by passing an interface through the configuration data base.

The test for the counter can be run with the `run.do` script, but the script assumes that you have accomplished three tasks:

1. Modify the `top.sv` file to create an interface holding object and pass it to the OVM. You must also specify the number of times you want to run through the testing loop.
2. Modify the `interface_holder.svh` file to define a class called `interface_holder`.
3. Modify the `counter_test.svh` file to add a `build()` method to the existing test. The build method needs to access the configuration DB and pull the `interface_holder` object out of the DB. The `build()` method must also get a value for the number of loops we will execute from the DB.

Each of these source files has comments in it that shows where to put the new code. The rest of the code has already been written including the `run()` method in the test that tests the counter.

The labs can be completed using the design patterns shown in the lectures.

Creating the Interface Holder

The `interface_holder` class will carry the virtual interface for the counter. Modify the file `interface_holder.svh` to add the correct data member and constructor.

Create the Top module

The top module stitches the interface to the device under test, puts the interface into the interface holder (ifh) and puts ifh into the configuration database.

The module is all set up with the instantiation of the interface and the DUT. You just need to create the always block that puts the interface-holding object into the configuration and calls `run_test()`.

Create the counter_test object

The script `run.do` assumes that there is a test called `counter_test`:

```
1 |if [file exists work] {vdel -all}
2 |vlib work
3 |vlog -novopt -f compile_sv.f
4 |vsim -novopt top +OVM_TESTNAME=counter_test
5 |log /* -r
6 |run -all
7 |
```

There is a `counter_test` object defined in the file `counter_test.svh`. This class extends the `ovm_test` class. The class is complete except for the `build()` method:

```
1 class counter_test extends ovm_test;
2
3     `ovm_component_utils(counter_test);
4
5     virtual interface counter_if i;
6     integer nloops = 2;
7     interface_holder ifh;
8     ovm_object tmp;
9
10    function new(string name="", ovm_component parent);
11        super.new(name, parent);
12    endfunction : new
13
14
15    virtual function void build();
16        super.build();
17
18    // Create a build() method that will do the following:
```

The `counter_test.svh` file contains the instructions you need to create the `build()` method. You need to get the interface holder out of the configuration db and set `i` equal to the interface stored in the holder. You also need to set `nloops`.

Running the Test

When you execute the `run.do` script you should see the following:

```
# -----
# OVM-2.0.1
# (C) 2007-2009 Mentor Graphics Corporation
# (C) 2007-2008 Cadence Design Systems, Inc.
# -----
# OVM_INFO @ 0: reporter [RNTST] Running test counter_test...
# OVM_INFO @ 20: ovm_test_top [run] data_in: 81 q: 00 ld: 0, inc: 0
# OVM_INFO @ 40: ovm_test_top [run] data_in: 63 q: 00 ld: 0, inc: 1
# OVM_INFO @ 60: ovm_test_top [run] data_in: 8d q: 01 ld: 0, inc: 1
# OVM_INFO @ 80: ovm_test_top [run] data_in: 12 q: 02 ld: 0, inc: 1
# OVM_INFO @ 100: ovm_test_top [run] data_in: 0d q: 03 ld: 0, inc: 1
# OVM_INFO @ 120: ovm_test_top [run] data_in: 3d q: 04 ld: 1, inc: 0
# OVM_INFO @ 140: ovm_test_top [run] data_in: 8c q: 3d ld: 0, inc: 1
# OVM_INFO @ 160: ovm_test_top [run] data_in: c6 q: 3e ld: 0, inc: 1
# OVM_INFO @ 180: ovm_test_top [run] data_in: aa q: 3f ld: 0, inc: 1
# OVM_INFO @ 200: ovm_test_top [run] data_in: 77 q: 40 ld: 0, inc: 1
#
# --- OVM Report Summary ---
#
# ** Report counts by severity
# OVM_INFO : 11
# OVM_WARNING : 0
# OVM_ERROR : 0
# OVM_FATAL : 0
# ** Report counts by id
# [RNTST] 1
# [run] 10
```