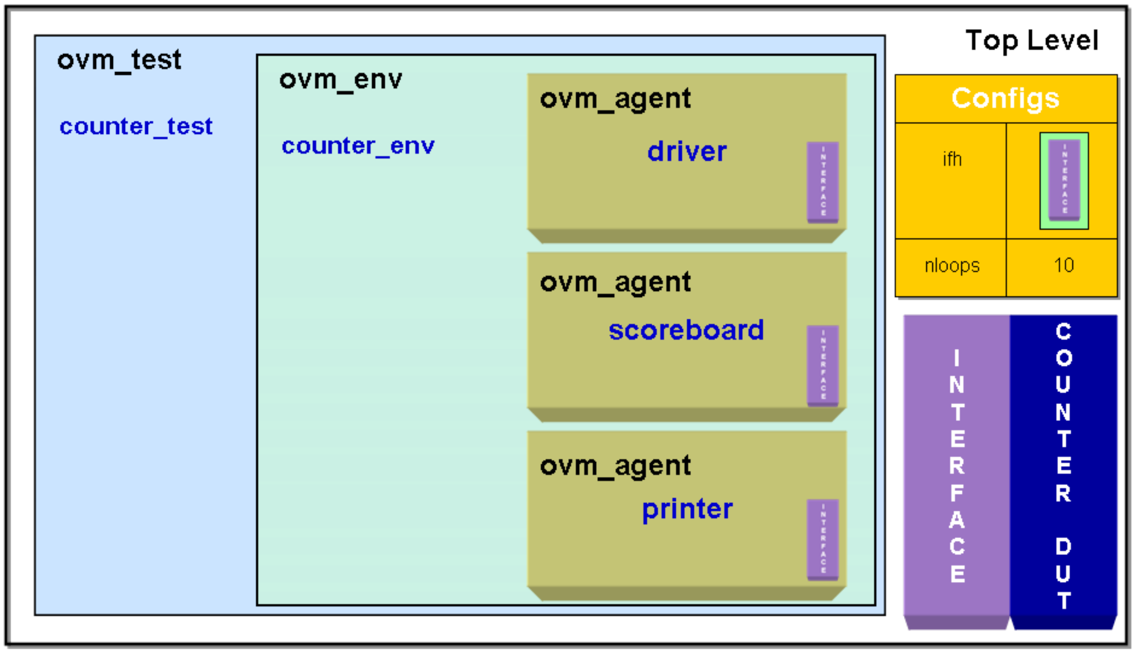
# Lab 6: Creating an OVM Environment

In this lab you will test the counter circuit using an OVM environment to run multiple threads. Your final test bench will be similar to this:



The top level of the design and the execution scripts have been provided. You need to modify only the following files:

- `counter_test.svh` – This file contains the definition of the counter test environment object.
- `counter_env.svh` – This file contains the definition of the counter_env environment object.
- `printer.svh` – This file contains the printer object. This object will print the data from the interface to the screen on each clock cycle.

## Executing the Test

You can run the test by executing the `run.do` script this way:

```
% vsim -c -do run.do
```

Or if you open Modelsim in the lab directory:

```
modelsim> do run.do
```

These commands will compile the design at this time, but nothing will happen. You will need to finish the test, environment, and printer to see results.

## Finishing the Test Object in counter_test.svh.

The counter_test object needs you to do two things:

1. Insert code that will create a counter_env object using the factory.

## Finishing the counter_env object in counter_env.svh

The counter_env object in counter_env.svh needs you to do three things:

1. Declare variables to hold the driver, scoreboard, and printer objects.
2. Create instances of the three objects.

## Complete the counter_agent class

The driver, scoreboard, and printer all use a pointer to the counter_if interface to access the counter's signals. Therefore, all three of them need a virtual interface variable to hold the counter's interface and a build() method that pulls the interface out of the config DB and puts it into the variable.

Rather than copy and paste the code three time, we are going to create a class that has the virtual interface data member, and that has a build() method to pull that interface out of the database. We'll call this class counter_agent.

Once we've defined the counter_agent class, we'll extend it to create the driver, scoreboard ,and printer.

The file counter_agent.svh contains much of the code needed to create the counter agent. This includes the declarations of the variables needed to create the build method.

Your job is to create the build() method so that it gets the virtual interface out of the config DB and puts it into a variable called i. All the three children classes will assume that i has been set to the interface.

## Completing the scoreboard Class in scoreboard.svh

The scoreboard object checks to make sure the counter is acting properly. It has a behavioral model of the counter and it compares the DUT to the behavior of the model.

The file scoreboard.svh contains most of the code for the scoreboard class. You only need to declare the class by extending counter_agent.

## Completing the driver Class in driver.svh

The driver class creates random stimulus and drives it into the counter_if interface. The interface, in turn, drives the DUT.

The file driver.svh contains most of the information in the driver. You need to do the following:

- Declare the driver class by extending counter_agent.
- Modify the build() method to use the build() method in the counter_agent.

### Completing the printer Class in printer.svh

The printer class samples the following signals on the negative edge of the clock in interface i and prints them to the screen:

- i.data_in
- i.q
- i.ld
- i.inc

The printer class extends the counter_agent class and leverages its build() script.

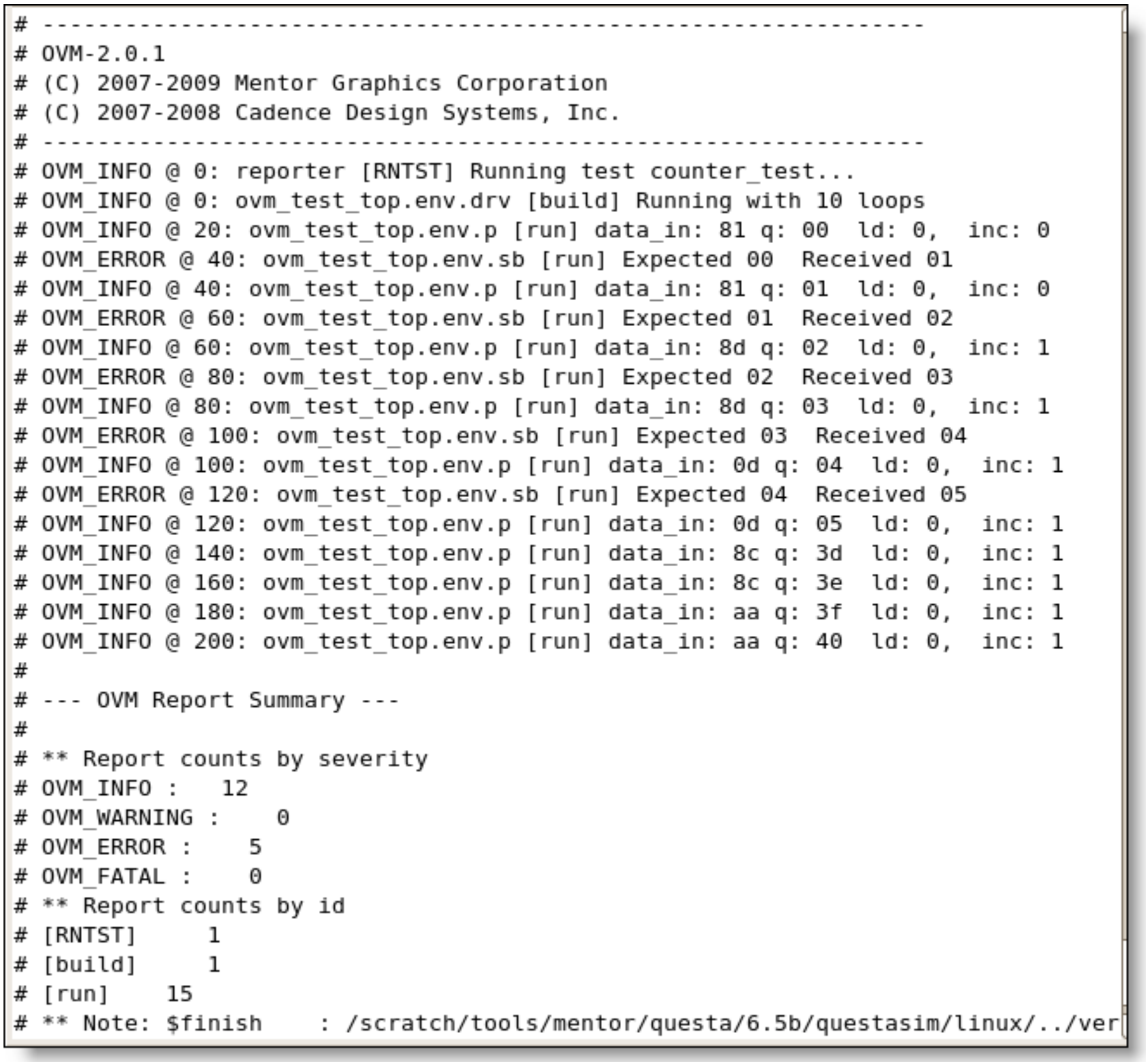
You need to do the following three things to complete the printer.

- Write the class declaration and ovm_component_utils macro.
- Create a build function declared as: virtual function void build()
- Create the run() task to print values on the negative edge of the clock.

Your printer class should create output that looks like the line below.

## Running the Test

When you run the test the result should look like this:



```
# ----------------------------------------------------------------
# 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 @ 0: ovm_test_top.env.drv [build] Running with 10 loops
# OVM_INFO @ 20: ovm_test_top.env.p [run] data_in: 81 q: 00  ld: 0,  inc: 0
# OVM_ERROR @ 40: ovm_test_top.env.sb [run] Expected 00  Received 01
# OVM_INFO @ 40: ovm_test_top.env.p [run] data_in: 81 q: 01  ld: 0,  inc: 0
# OVM_ERROR @ 60: ovm_test_top.env.sb [run] Expected 01  Received 02
# OVM_INFO @ 60: ovm_test_top.env.p [run] data_in: 8d q: 02  ld: 0,  inc: 1
# OVM_ERROR @ 80: ovm_test_top.env.sb [run] Expected 02  Received 03
# OVM_INFO @ 80: ovm_test_top.env.p [run] data_in: 8d q: 03  ld: 0,  inc: 1
# OVM_ERROR @ 100: ovm_test_top.env.sb [run] Expected 03  Received 04
# OVM_INFO @ 100: ovm_test_top.env.p [run] data_in: 0d q: 04  ld: 0,  inc: 1
# OVM_ERROR @ 120: ovm_test_top.env.sb [run] Expected 04  Received 05
# OVM_INFO @ 120: ovm_test_top.env.p [run] data_in: 0d q: 05  ld: 0,  inc: 1
# OVM_INFO @ 140: ovm_test_top.env.p [run] data_in: 8c q: 3d  ld: 0,  inc: 1
# OVM_INFO @ 160: ovm_test_top.env.p [run] data_in: 8c q: 3e  ld: 0,  inc: 1
# OVM_INFO @ 180: ovm_test_top.env.p [run] data_in: aa q: 3f  ld: 0,  inc: 1
# OVM_INFO @ 200: ovm_test_top.env.p [run] data_in: aa q: 40  ld: 0,  inc: 1
#
# --- OVM Report Summary ---
#
# ** Report counts by severity
# OVM_INFO :   12
# OVM_WARNING :    0
# OVM_ERROR :    5
# OVM_FATAL :    0
# ** Report counts by id
# [RNTST]     1
# [build]     1
# [run]    15
# ** Note: $finish    : /scratch/tools/mentor/questa/6.5b/questasim/linux/../ver
```