03/03/19 HOMEWORK # 03

Saqib Khan
Sak2454

1) What is phasing in UVM? How do objections help in UVM phasing?

Ans) UVM phases are a synchronizing mechanism for the environment. Phases can be grouped into 3 categories

1) Build phases :- these phases are executed at the start of UVM test bench simulation, where the test bench components are constructed, configured and test bench components are connected.

2) Run-time phases :- will get executed from start of simulation till the end of simulation.

3) Clean-up phases :- results of test cases are collected & reported.

UVM provides an objection mechanism to allow hierarchical status communication among components which is helpful in deciding the end of test.

2) What are virtual interfaces in System Verilog? Where are virtual interfaces useful in a System Verilog UVM test bench?

Ans) Virtual interfaces provide a mechanism for separating abstract models and test programs from the actual signals that make up the design.

Virtual interfaces are useful for developing the test components independent of the device under test (DUT) port while working with multi-port protocol.

3) When writing System Verilog code for UVM class, which phases can be defined as functions and which as tasks?

Ans) All UVM phases except the run_time phase are execute in zero-time. Run-time phase involves all time consuming activities, such as driving test sequences, and monitoring output etc. Therefore, run-time phase is classified as a "task", while all other phases are defined as functions.

4) What does it mean for a phase to be top-down or bottom-up? Which UVM phases are top-down / bottom-up?

Ans) A UVM phase can be executed in one of two ways;

1) top-down: where component tree is traversed in top to bottom manner
2) bottom-up :- where the child object is executed first and then traverses up to the parent object

Build phase is top-down because the parent object instantiates the child object during its build phase

Run-time phase is bottom-up because all components have to be run in parallel

5) What are the roles of each of the following components?

Sequencer:- is the primary stimulus generator, which creates the transactions, randomizes them and sends them to the driver

Driver:- is responsible for decoding the transaction received from the sequencer and also drives the DUT interface signals.

Monitor:- It's responsibility is to observe communication on the DUT interface.
It can include a protocol checker that can immediately find any pin level violations of the communication protocol.

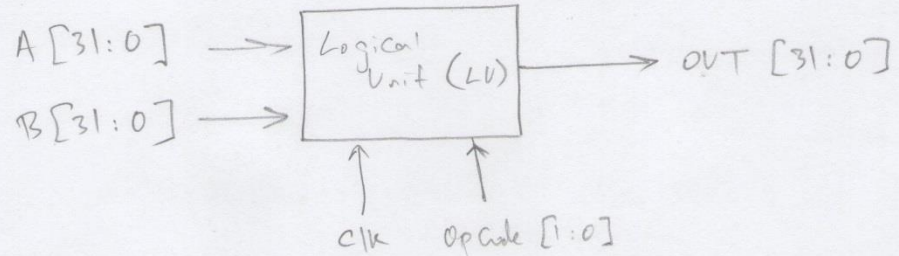
The monitor can be replaced by the driver. However, it limits the scalability of the driver and also prevents the driver from being reused for the testbench.

6) What is the advantages and differences of 'uvm_component_utils() and 'uvm_object_utils()?

Ans) The utils macros define the infrastructure needed to enable the object/component for correct factory operation.

The reason there are two macros is because the factory design pattern fixes the # of arguments that a constructor can have. Classes derived from uvm_object have constructures with one argument, a string name.
While uvm_component have two arguments, a name and a uvm_component parent.

7) Logical Unit Specification

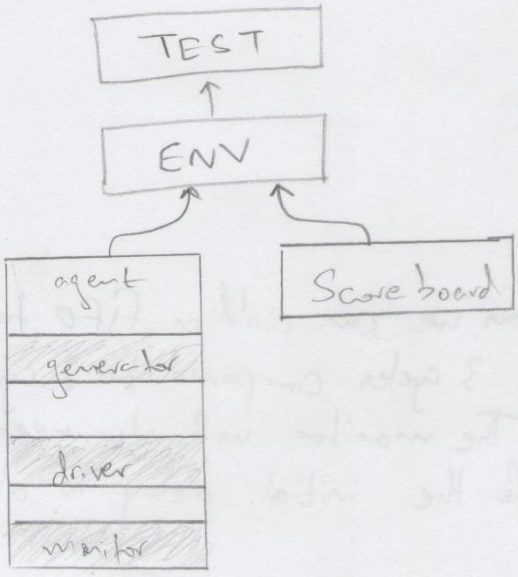


3 Steps involved in Verification Plan

A) Creating Verification Plan

1) test few scenarios of A NAND B
2) test A NOR B
3) test A XOR B
4) test invalid opcode

B) Testbench Architecture



C) Writing Verification Environment / Test bench

In a pipelined version, we can add a fifo to handle the increased latency of 3 cycles compared to 1 cycle in the non-pipelined version. The monitor will also need to be adjusted to account for the initial delay to allow the pipe to fill-in.