



Digital Design Verification

**Lab Manual # 44 – Integrating Multiple UVCs**

**Release: 1.0 Date: 8-Aug-2024**

**NUST Chip Design Centre (NCDC), Islamabad, Pakistan**

***Copyrights ©,*** *NUST Chip Design Centre (NCDC). All Rights Reserved. This document is prepared by NCDC and is for intended recipients only. It is not allowed to copy, modify, distribute or share, in part or full, without the consent of NCDC officials.*

**Revision History**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Revision Number** | **Revision Date** | **Revision By** | **Nature of Revision** | **Approved By** |
| 1.0 | 8/08/2024 | Amber Khan | Complete Manual | - |

# Contents

[Contents 2](#_bookmark0)

[Objective 3](#_bookmark1)

[Tools 3](#_bookmark2)

[Instructions for Lab Tasks 3](#_bookmark3)

[Task 1: Integrating Multiple UVCs 3](#_bookmark4)

# Objective

The objectives of this lab are

* To connect and configure the HBUS UVC, Clock and Reset UVC and three output Channel UVCs.

# Tools

* SystemVerilog
* Cadence Xcelium

# Instructions for Lab Tasks

The submission must follow the hierarchy below, with the folder named after the student (no spaces), and the file names exactly as listed below.

## ./student\_name\_lab12/

## ├── task1\_integ/

## │ ├── tb/

## │ ├── sv/

## ├── router\_rtl

## ├── yapp

## ├── hbus

## ├── channel

## ├── clk\_and\_reset

# Task 1: Integrating Multiple UVCs

For this lab, you will connect the HBUS, Clock and Reset and Channel UVCs to the router DUT. All three UVCs are provided. None of the UVCs use configuration objects.

These are the directories we will be using for this and subsequent labs:

hbus/sv HBUS UVC files

channel/sv Channel UVC files

clock\_and\_reset/sv Clock and Reset UVC files

yapp/sv YAPP input UVC (your files from lab06\_vif)

router\_rtl Router DUT

Your current working directory for this lab will be this one:

task1\_integ Your working directory for this lab

### Setting Up the Directory Structure

1. First – your YAPP UVC is now complete enough to stand by itself. Copy your YAPP files from

lab11/task2/sv into lab12/yapp/sv.

1. We will still be working on the testbench, testclass, and top files. Copy the files from

lab11/task2/tb into lab12/task1\_integ/tb. Work in the lab12/task1\_integ/tb directory.

### Testbench: Channel UVC

1. Update your testbench, router\_tb.sv, to add the Channel UVCs.
   1. Add three handles of the Channel UVC (channel\_env) and create the instances in the build\_phase() method using factory calls.
   2. Use a configuration set method to set the channel\_id property of each Channel instance. The Channel instance for address 0 should have a channel\_id of 0, the Channel instance for address 1 should have a channel\_id of 1and the Channel instance for address 2 should have a channel\_id of 2. For example,

uvm\_config\_int::set(this, "chan0", "channel\_id", 0);

### Testbench: HBUS UVC

1. Update your testbench, router\_tb.sv, to add the HBUS UVCs.
2. Add a handle of the Channel UVC (hbus\_env) and create the instance in the build\_phase() method using factory call.
3. Use configuration set methods to set the num\_masters property of the HBUS UVC to 1, and the num\_slaves property to 0. The HBUS UVC has both master and slave agents. For the router testing, we only need the master agent.

### Testbench: Clock and Reset UVC

1. Update your testbench, router\_tb.sv, to add a handle of the Clock and Reset UVC (clock\_and\_reset\_env) and create the instance in the build\_phase() method using a factory call. This UVC requires no configuration.

### Hardware Top Module hw\_top

1. Update hw\_top\_dut.sv as follows:
   1. Add an interface instantiation for the Clock and Reset. The interface file can be found in the clock\_and\_reset/sv directory. Map the clock, reset, run\_clock and clock\_period interface ports to the local signals of the same name.
   2. Connect the clkgen module instance to the Clock and Reset interface instance by replacing the run\_clock and clock\_period literal port mappings with the local signals of the same name.
   3. Add interface instantiations for the HBUS and all three Channels. The interface files can be found in the sv directory of each UVC directory. Map the ports of the interfaces to the local clock and reset signals of the same name.
   4. As the reset will now be generated by the Clock and Reset UVC, delete the initial block which generates the reset waveform.
   5. Update the port mapping of the router instantiation to connect the Channel and HBUS interface signals.

### UVM Top Module tb\_top

1. Update tb\_top.sv as follows:
   1. Add imports for the Channel, Clock and Reset and HBUS UVC package files. The packages can be found in the sv directory of each UVC.
   2. Set the HBUS, Clock and Reset and Channel UVC virtual interfaces to the correct interface. (Hint: The UVC header files contain typedefs for each interface.)

Use wildcards in the pathname to update all UVC components with a single statement.

Use an absolute hierarchical pathname for the value to select the correct interface instance from the hw\_top module.

### Running Base Test

1. For every UVC (YAPP, Clock and reset, HBUS and Channel) add the following to your run.f.

* An incdir reference to the UVC sv directory (depending upon the location of UVC in reference to your current working directory)
* UVC package filename.
* UVC interface filename.

1. Run a simulation with base\_test only. Check the topology report carefully to make sure all of your UVCs are instantiated and configured correctly. Copy the topology report into a new file for future reference.

### Test Library

1. Add a new test class, simple\_test, in router\_test\_lib.sv as follows (copy from existing tests). Sequencer pathnames can be read from the topology report.
   1. Set the YAPP UVC to create short YAPP packets with a set\_type\_override.
   2. Set the default sequence of the YAPP UVC to yapp\_012\_seq.
   3. Set the default sequence of each Channel UVC to channel\_rx\_resp\_seq. Hint: you can set all three Channel UVCs with a single statement.
   4. Set the default sequence of the Clock and Reset UVC to clk10\_rst5\_seq.
   5. Do not define a default sequence for the HBUS UVC.
   6. Clean up the test library and remove the older tests. Delete or comment out all the other test classes besides base\_test and simple\_test.

### Running Simple Test

1. Run a simulation in GUI mode using simple\_test and verify as follows:
   1. Add YAPP UVC monitor transactions to the waveform viewer.
   2. Add all three Channel UVC monitor transactions to the waveform viewer.
   3. Use the transactions to confirm packets are passed correctly through the router and collected at the right channel.

### Further Integration Testing

1. Write a new YAPP sequence in the yapp/sv/yapp\_tx\_seqs.sv file to generate packets for all four channels (including the illegal address 3). The packets should have incrementing payload sizes from 1 to 22 and parity distribution of 20% bad parity (88 packets in total).

Hint: You could create packets using nested loops for address and payload.

1. Create a new test, test\_uvc\_integration, in the router\_test\_lib.sv file to perform the following:
   1. Set the run\_phase default sequence of the YAPP UVC to the sequence created above.
   2. Set the run\_phase default sequence of the HBUS UVC to set up the router with register field maxpktsize = 20 and enable the router (register field router\_en = 1).

Hint: There is a sequence defined for this in the HBUS master sequences hbus\_master\_seqs.sv.

Hint: The hierarchical path name for the HBUS configuration setting can be read from the topology report.

1. Run a simulation and check the results to see that the three channels are properly addressed, that there is an error signal when parity is wrong, and that packets are dropped if bigger than maxpktsize or have illegal addresses.