CHIPXPRT

Lab 13

Creating Scoreboard using TLM

Module ID : CX-301

Design Verification

Instructor : Dr. Abid Rafique

Version 1.1

Release: 1.0

Document History

The changes and versions of the document are outlined below:

Version	State / Changes	Date	Author
1.0	Initial Draft	Jan, 2024	Qamar Moavia
1.1	Modified with new exercises	March, 2025	Qamar Moavia

Objectives

For this lab, you will build and connect a scoreboard for the router and create TLM analysis port connections to hook up the scoreboard to the UVCs. The router Module UVC is a complex design, so this lab has been deliberately broken down into two separate steps to build the UVC progressively. The first step is to implement the scoreboard component itself and connect it up to the YAPP and Channel UVCs. For this part of the lab we assume all packets are sent to legal addresses with legal payload length, i.e., the router does not drop any packets.

Tools

- SystemVerilog
- Synopsys VCS

Instructions for Lab Tasks

The required files for this lab can be found on the

```
shared_folder/CX-301-DesignVerification/Labs/Lab13
```

The submission must follow the hierarchy below, with the folder named and the file names exactly as listed below.

```
./Lab13
                ├── task1_sb/
                │   ├── tb/
                    |---sv/
                ├── router_rtl
                ├── yapp
                ├── hbus
                ├── channel
                ├── clk_and_reset
```

Creating Scoreboard using TLM Connections

1. Copy the directory from Lab12 into Lab13. Work in the Lab13/tb directory (or any other directory of your choice such as a separate work directory - make sure you change paths in filelist.f accordingly).
2. A TLM analysis port has already been implemented in the Channel UVC monitor for collected YAPP packets. This port is named `item_collected_port`. Check the Channel monitor in the file `channel/sv/channel_rx_monitor.sv` to make sure you understand how this port is used.
3. Modify `yapp/sv/yapp_tx_monitor.sv` to create an analysis port instance.
    a. Declare an analysis port object, parameterized to the correct type.
    b. Construct the analysis port in the monitor constructor.
    c. Call the port `write()` at the appropriate point.

4. Make a `sv` directory in `Lab13/task1_sb` directory, create the scoreboard, `router_scoreboard.sv`.
    a. Extend from `uvm_scoreboard` and add a component utility macro and a constructor.
    b. As the YAPP and Channel UVCs use different packet types, you will need a custom comparison function to compare `yapp_packet` and `channel_packet` packets. You can use either simple Verilog comparison operators or the `uvm_comparer` class (see slides and reference material for details). A simple comparison operator is provided to you in the file `packet_compare.sv`, copy this into your scoreboard.
    c. Define four analysis imp objects (for the YAPP and three Channels) using `` `uvm_analysis_imp_decl `` macros and `uvm_analysis_imp_*` objects
    d. Create analysis imp instances in the scoreboard constructor.
    e. Use the YAPP `write()` implementation to clone the packet and then push the packet to a queue. **Hint: Use a queue for each address.**

   f. Use Channel `write()` implementations to pop packets from the appropriate queue and compare them to the channel packets, using your custom comparer function.
   g. Add counters for the number of packets received, wrong packets (compare failed) and matched packets (compare passed).
   h. Add a `report_phase()` method to print the number of packets received, wrong packets, matched packets and number of packets left in the queues at the end of simulation.
5. In the tb directory, update the `tb_top` module to include the router scoreboard.
6. In the tb directory, modify the `router_tb.sv` as follows:
   a. Declare and build the scoreboard.
   b. Make the TLM connections between YAPP, Channel, and scoreboard.
   c. Use a test which generates a good number of legal YAPP packets, i.e. short packets with legal addresses, so that no packets are dropped by the router. We could use the Lab 12 multichannel sequence test or if you did not complete that lab, you could modify the Lab 9 exhaustive sequence test to add the Channel and Clock and Reset sequences.
   d. Check that the simulation results are correct, and debug as required.
7. Once you are happy that the scoreboard is working, check that it correctly reports mismatched packets. You can achieve this by commenting out the short YAPP packet type override in your test class which will allow the YAPP UVC to send oversized packets which will be dropped by the router and create mismatches in the scoreboard. Make sure that your HBUS sequencer is executing hbus_small_packet_seq to set the MAXPKTSIZE register to 20.

   Run a simulation with the type override removed and check the log file for the following:

   - Your YAPP UVC is generating packets of type yapp_packet.
   - The RTL router code generates ROUTER DROPS PACKET messages.
   - Your custom comparer function generates uvm_error messages when the comparison fails.
   - Packets are left in the scoreboard queues at the end of simulation.

- Your scoreboard reports the number of received, mismatched and matched packets, as well as the number of packets left in the queues. Check the number of packets add up!

8. (Optional) Write your own custom comparer function which uses uvm_comparer methods instead of Verilog comparisons. Test your new implementation in simulation.