Lab 4: UVM Introduction

Sri Sai Harshith Yellanki
Shamit Savant

Part 2 – Adding a Debug Test and Sequence

Objective

In Part 2 of the lab, we extended the UVM testbench with a new deterministic test and sequence to support easier **waveform-based debugging**. Unlike Part 1, which emphasized constrained-random coverage, this part prioritizes visibility and predictability in packet contents and structure. Our goal was to generate clearly traceable packets in the simulation waveform for manual inspection.

What We Did

We made a few targeted additions to our UVM testbench to support debugging:

- In the filter_sequence.svh file, we added a new sequence called filter_debug_sequence. Instead of random packet lengths, we set the length equal to the loop index. For example, the first packet had 0 bytes, the second had 1, the third had 2, and so on.
- We also filled all payload bytes with the fixed value 0x55 to make them easy to spot in the waveform.
- We used a variable message type which alternated between 0x0, 0x5, 0xA, 0x3. (Simulation Screenshot 1)
- We then fixed the message type to 0x0 so that every packet would pass through the DUT. (Simulation Screenshot 2)
- We created a new test in filter_debug_test.svh. This test starts the filter_debug_sequence and connects it to the environment. It raises and drops objections properly to ensure the simulation runs to completion.
- In the filter_tb_pkg.sv file, we added the new filter_debug_test.svh include so our new test would be compiled with the rest of the testbench.
  In the Makefile, we changed the default test from filter_packet_test to filter_debug_test. This ensured that when we ran the simulation using the make command, our debug test would be executed.

What the Debug Test Does

The test is designed to make each packet visually distinct and easy to follow in the waveform:

- The packet length increases by 1 in every iteration, which makes it easy to track in the waveform.

- The payload is filled with the same value, 0x55, so it's easy to recognize.
- The message type is fixed at 0x0, which the DUT recognizes as valid, ensuring each packet is forwarded.

This structure made it very easy to look at the waveform and understand exactly what was happening at each step. We could spot headers, track how the length was affecting the output, and clearly see any unexpected behavior.

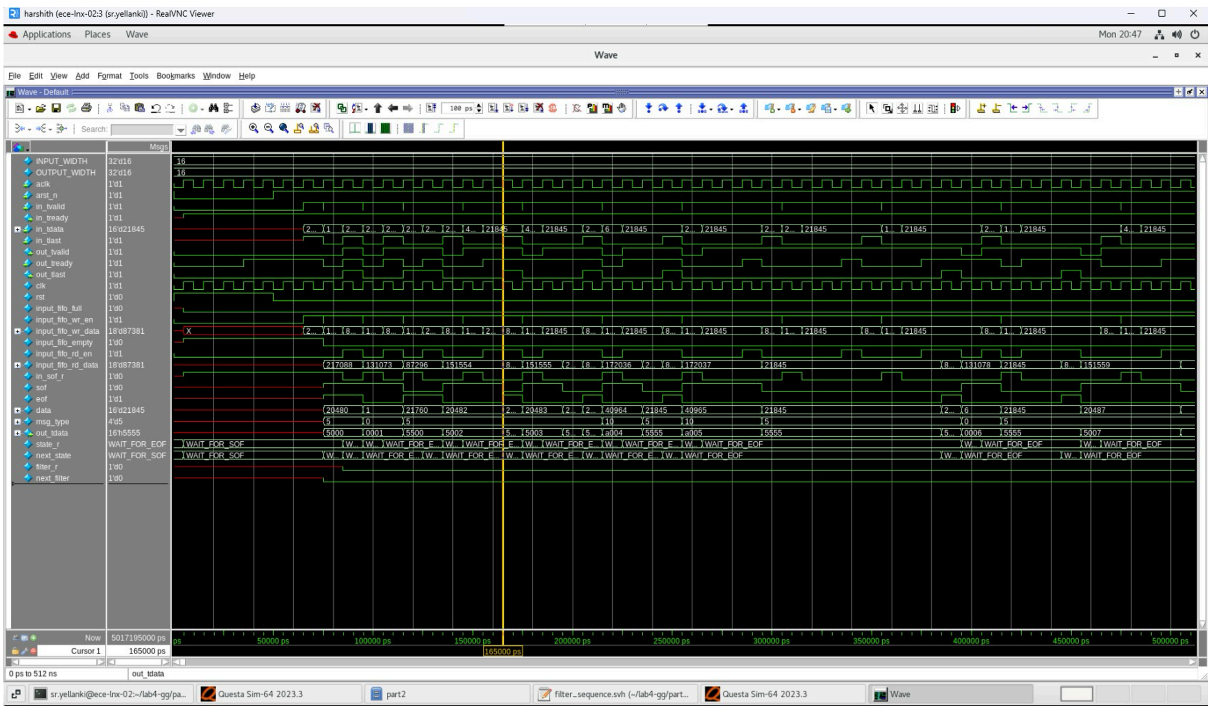
Simulation and Results

We ran the simulation using the GUI mode and looked at the waveform. Everything behaved exactly as expected:

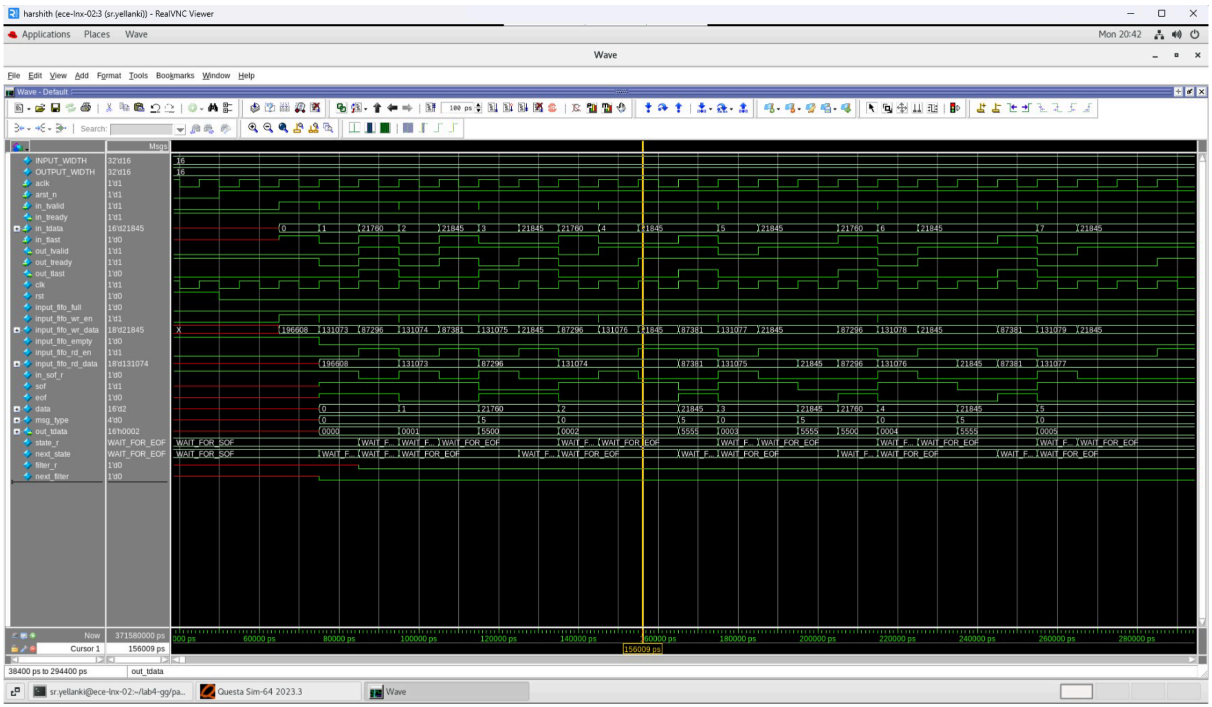
- Each packet had the right length based on the loop index.
- All payloads were filled with 0x55 and were aligned correctly.
- The output matched the input for each valid packet.

This confirmed that the DUT was handling the debug packets correctly, and the testbench was functioning as intended.

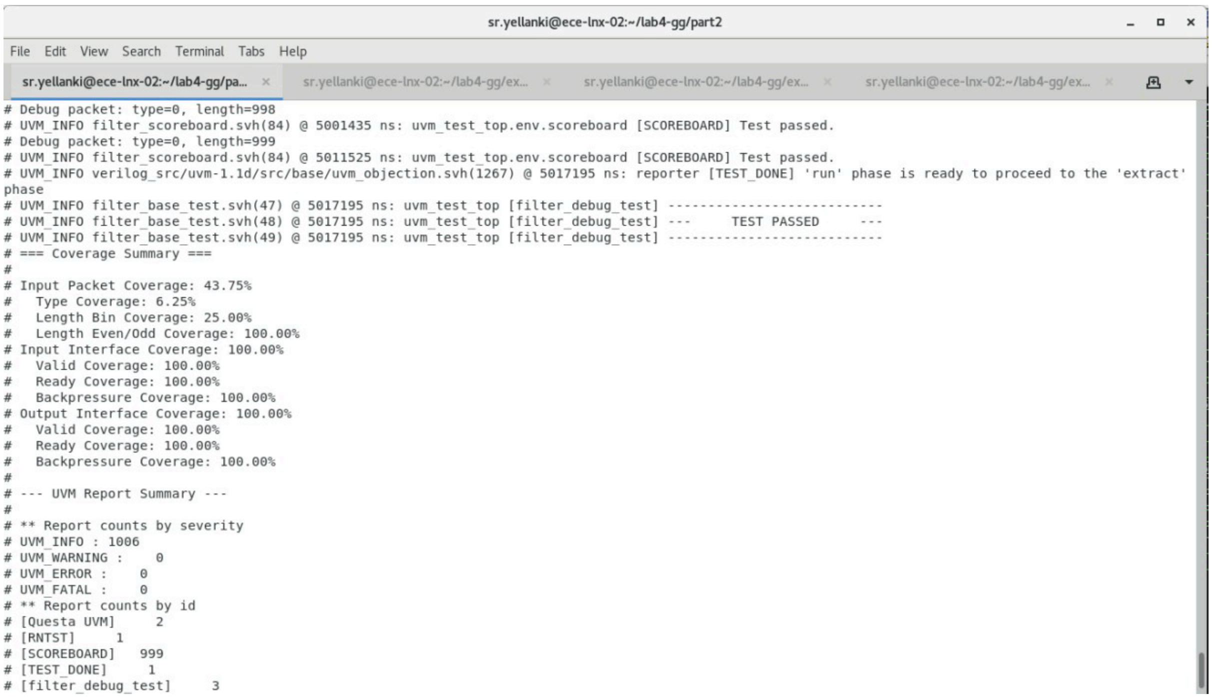
Screenshots



```
INPUT_WIDTH          32'd16
OUTPUT_WIDTH         32'd16
aclk                 1'd1
arst_n               1'd1
in_tvalid            1'd1
in_tready            1'd1
in_tdata             16'd21845
in_tlast             1'd1
out_tvalid           1'd1
out_tready           1'd1
out_tlast            1'd1
clk                  1'd1
rst                  1'd0
input_fifo_full      1'd0
input_fifo_wr_en     1'd1
input_fifo_wr_data   18'd87381
input_fifo_empty     1'd0
input_fifo_rd_en     1'd1
input_fifo_rd_data   18'd87381
in_sof_r             1'd0
sof                  1'd0
eof                  1'd1
data                 16'd21845
msg_type             4'd5
out_tdata            16'h5555
state_r              WAIT_FOR_EOF
next_state           WAIT_FOR_SOF
filter_r             1'd0
next_filter          1'd0
```



```
INPUT_WIDTH          32'd16
OUTPUT_WIDTH         32'd16
aclk                 1'd1
arst_n               1'd1
in_tvalid            1'd1
in_tready            1'd1
in_tdata             16'd21845
in_tlast             1'd0
out_tvalid           1'd1
out_tready           1'd1
out_tlast            1'd0
clk                  1'd1
rst                  1'd0
input_fifo_full      1'd0
input_fifo_wr_en     1'd1
input_fifo_wr_data   18'd21845
input_fifo_empty     1'd0
input_fifo_rd_en     1'd1
input_fifo_rd_data   18'd131074
in_sof_r             1'd0
sof                  1'd1
eof                  1'd0
data                 16'd2
msg_type             4'd0
out_tdata            16'h0002
state_r              WAIT_FOR_EOF
next_state           WAIT_FOR_EOF
filter_r             1'd0
next_filter          1'd0
```


sr.yellanki@ece-lnx-02:~/lab4-gg/part2
File
Edit
View
Search
Terminal
Tabs
Help
sr.yellanki@ece-lnx-02:~/lab4-gg/pa...
sr.yellanki@ece-lnx-02:~/lab4-gg/ex...
sr.yellanki@ece-lnx-02:~/lab4-gg/ex...
sr.yellanki@ece-lnx-02:~/lab4-gg/ex...

```
# Debug packet: type=0, length=998
# UVM_INFO filter_scoreboard.svh(84) @ 5001435 ns: uvm_test_top.env.scoreboard [SCOREBOARD] Test passed.
# Debug packet: type=0, length=999
# UVM_INFO filter_scoreboard.svh(84) @ 5011525 ns: uvm_test_top.env.scoreboard [SCOREBOARD] Test passed.
# UVM_INFO verilog_src/uvm-1.1d/src/base/uvm_objection.svh(1267) @ 5017195 ns: reporter [TEST_DONE] 'run' phase is ready to proceed to the 'extract'
phase
# UVM_INFO filter_base_test.svh(47) @ 5017195 ns: uvm_test_top [filter_debug_test] ---------------------------
# UVM_INFO filter_base_test.svh(48) @ 5017195 ns: uvm_test_top [filter_debug_test] ---      TEST PASSED     ---
# UVM_INFO filter_base_test.svh(49) @ 5017195 ns: uvm_test_top [filter_debug_test] ---------------------------
# === Coverage Summary ===
#
# Input Packet Coverage: 43.75%
#   Type Coverage: 6.25%
#   Length Bin Coverage: 25.00%
#   Length Even/Odd Coverage: 100.00%
# Input Interface Coverage: 100.00%
#   Valid Coverage: 100.00%
#   Ready Coverage: 100.00%
#   Backpressure Coverage: 100.00%
# Output Interface Coverage: 100.00%
#   Valid Coverage: 100.00%
#   Ready Coverage: 100.00%
#   Backpressure Coverage: 100.00%
#
# --- UVM Report Summary ---
#
# ** Report counts by severity
# UVM_INFO : 1006
# UVM_WARNING :    0
# UVM_ERROR :    0
# UVM_FATAL :    0
# ** Report counts by id
# [Questa UVM]     2
# [RNTST]     1
# [SCOREBOARD]   999
# [TEST_DONE]     1
# [filter_debug_test]     3
```