# Wrapper Module Test Plan

Hongkuan Yu

Jan 2025

## Contents

# 1 DUT

This module is simple, so I include the SystemVerilog code of `dut.sv` below.

```
module dut(clk,
           rst_n,
           rxd,
           rx_dv,
           txd,
           tx_en);
    input clk;
    input rst_n;
    input[7:0] rxd;
    input rx_dv;
    output [7:0] txd;
    output tx_en;

    reg[7:0] txd;
    reg tx_en;

    always @(posedge clk) begin
       if(!rst_n) begin
          txd <= 8'b0;
          tx_en <= 1'b0;
       end
       else begin
          txd <= rxd;
          tx_en <= rx_dv;
       end
    end
endmodule
```

This `dut.sv` serves as **a simple data wrapper**. The I/O ports of `dut.sv` and their functions are shown in Table 1 below.

| | | | |
|---|---|---|---|
| input (wire) | clk | 1 bit | Global clock |
| input (wire) | rst_n | 1 bit | Active-low reset |
| input (wire) | rxd | 8 bits | Data received |
| input (wire) | rx_dv | 1 bit | Valid for data received |
| output (reg) | txd | 8 bits | Data transmitted |
| output (reg) | tx_en | 1 bit | Valid for data transmitted |

Table 1: I/O ports of `dut.sv` and their functions

The module I/O graphic overview is shown in Figure 1 below.

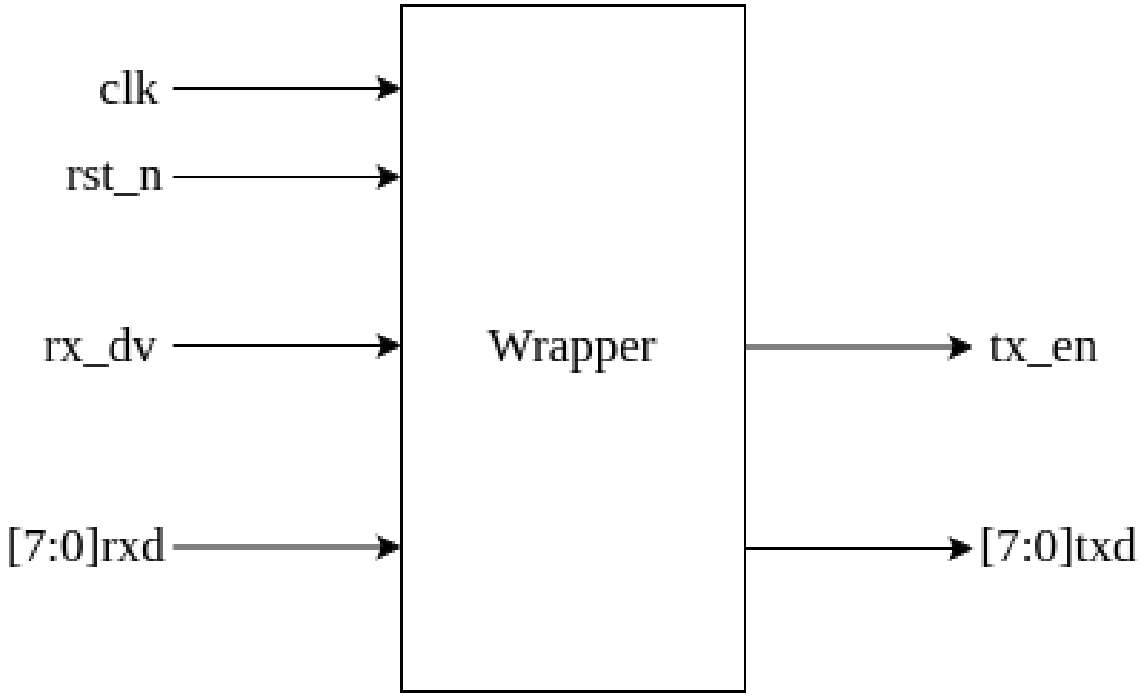


Figure 1: Wrapper Module I/O Overview

## 2 Simulation Testbench (Dynamic)

Write a simulation testbench `dut_tb.sv` to verify the `dut.sv` dynamically by Synopsys VCS. Create `filelist.f` and `Makefile` containing simulation commands.

The results of the simulation test bench meet expectations.

# 3 Formal Verification (Static)

Create `sva` folder for formal verification. Since I currently do not have Synopsys VC Formal or Cadence JasperGold installed, I skipped this section.

# 4 UVM

The UVM components include (from top to bottom): a base test, environment, model, scoreboard, agent, sequencer, sequence, transaction, driver, and monitor. An interface is included for portable design. `top_tb.sv` connects them all.

The structure of the UVM platform is shown in Figure 2 below.

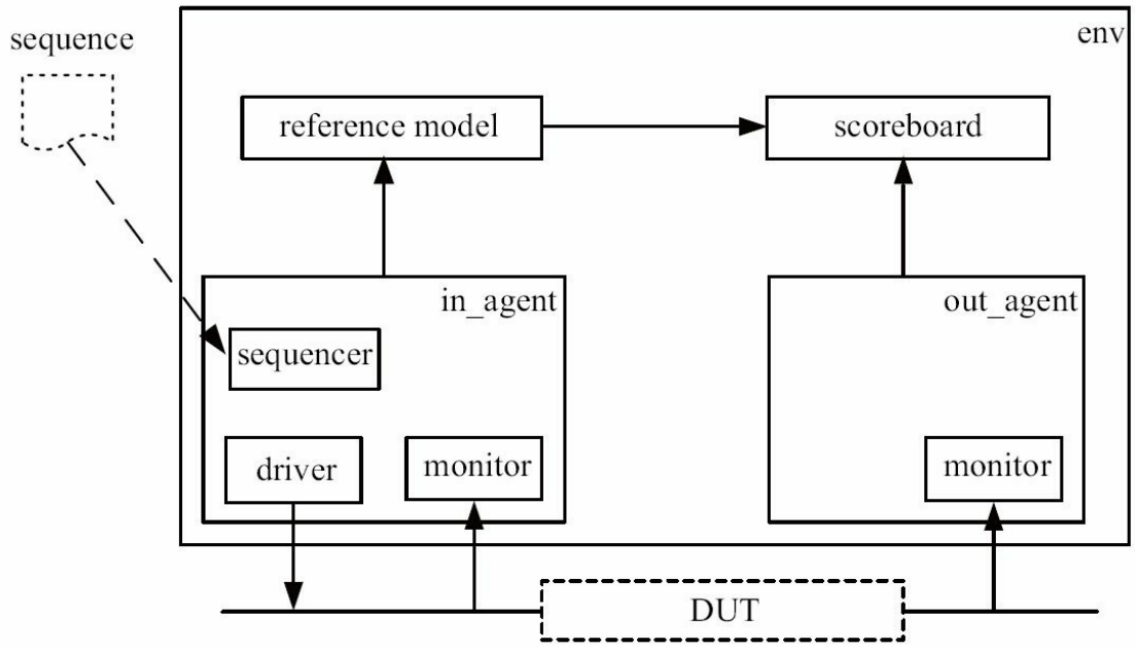


Figure 2: Structure of UVM Platform

The UVM tree is shown in Figure 3 below.

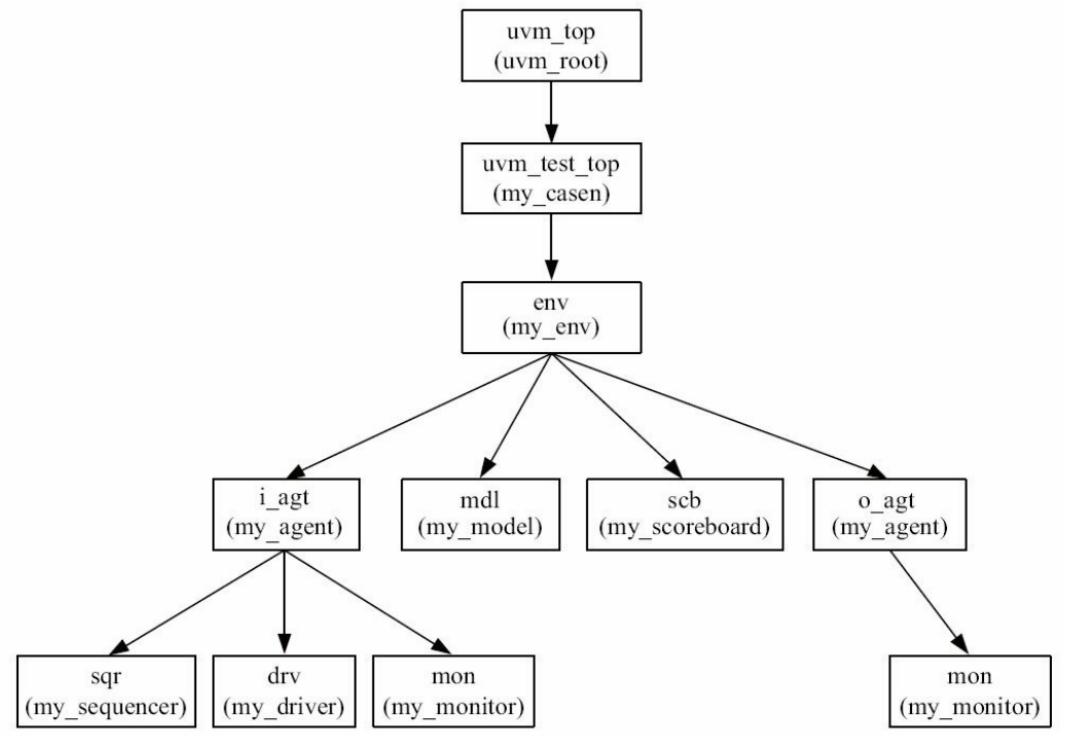


Figure 3: UVM Tree

The transaction mimics the bitstream of an ethernet connection by generating random MAC addresses and other related information. Test case 0 does not restrict the bitstream length. Test case 1 restricts the bitstream length to 60 bytes. Both test cases passed.

# References