UVM testbench summary

apb_component.sv

UVM component instantiating a single UVM agent.

apb_component_agent.sv

UVM agent instantiating and connecting the component's internals.

apb_component_driver.sv

UVM driver defining procedures to send packets to the DUT and to reset it.

apb_component_monitor.sv

UVM monitor with scoreboard functionalities, it defines monitoring procedures and keeps an internal model of the DUT's register set.

apb_component_package.sv

Package including all the files related to the UVM testing component and `typedef`-ing `uvm_config_db#(virtual apb_interface)` as `apb_interface_config`.

apb_component_sequence_library.sv

Library defining the available test sequences: a do-nothing base class defines the objection mechanisms, while multiple extending classes define the actual sequences.

apb_component_sequencer.sv

UVM sequencer inside the testing component.

apb_interface.sv

Interface to the APB slave including tasks detailing how to reset the DUT, send a packet to it and capture its response.

apb_packet.sv

Class defining an “APB packet”, namely the content of a transaction sent to the APB slave; both single and 2-burst transactions are supported. The class includes some randomization constraints.

apb_parameters.svh

SystemVerilog header defining hardware-related macros that specify the parameters of the APB slave and the period of the test clock.

apb_response.sv

Class defining an “APB response packet”, namely what the APB slave is expected to output in response to a requested transaction. A method to calculate the response starting from the transaction and the bridge registers’ content is included.

hardware_top.sv

Hardware top module instantiating the DUT, the interface to access it and the clock/reset logic.

test_library.sv

UVM test library, it defines the base test class and all its daughter classes. The following actions are coded in the base class and just inherited by the other classes:

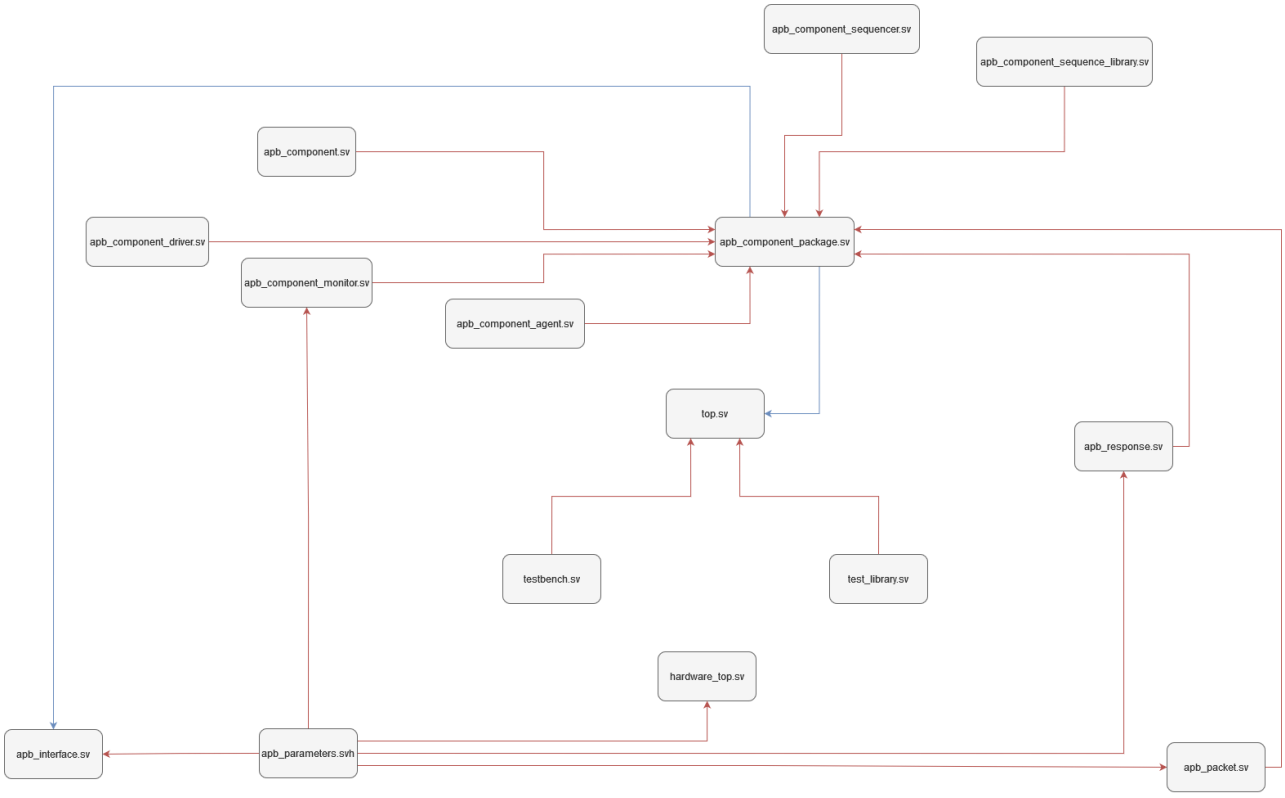
- Instantiation of the UVM testbench
- Enabling of the transaction recording
- Setting of the drain time

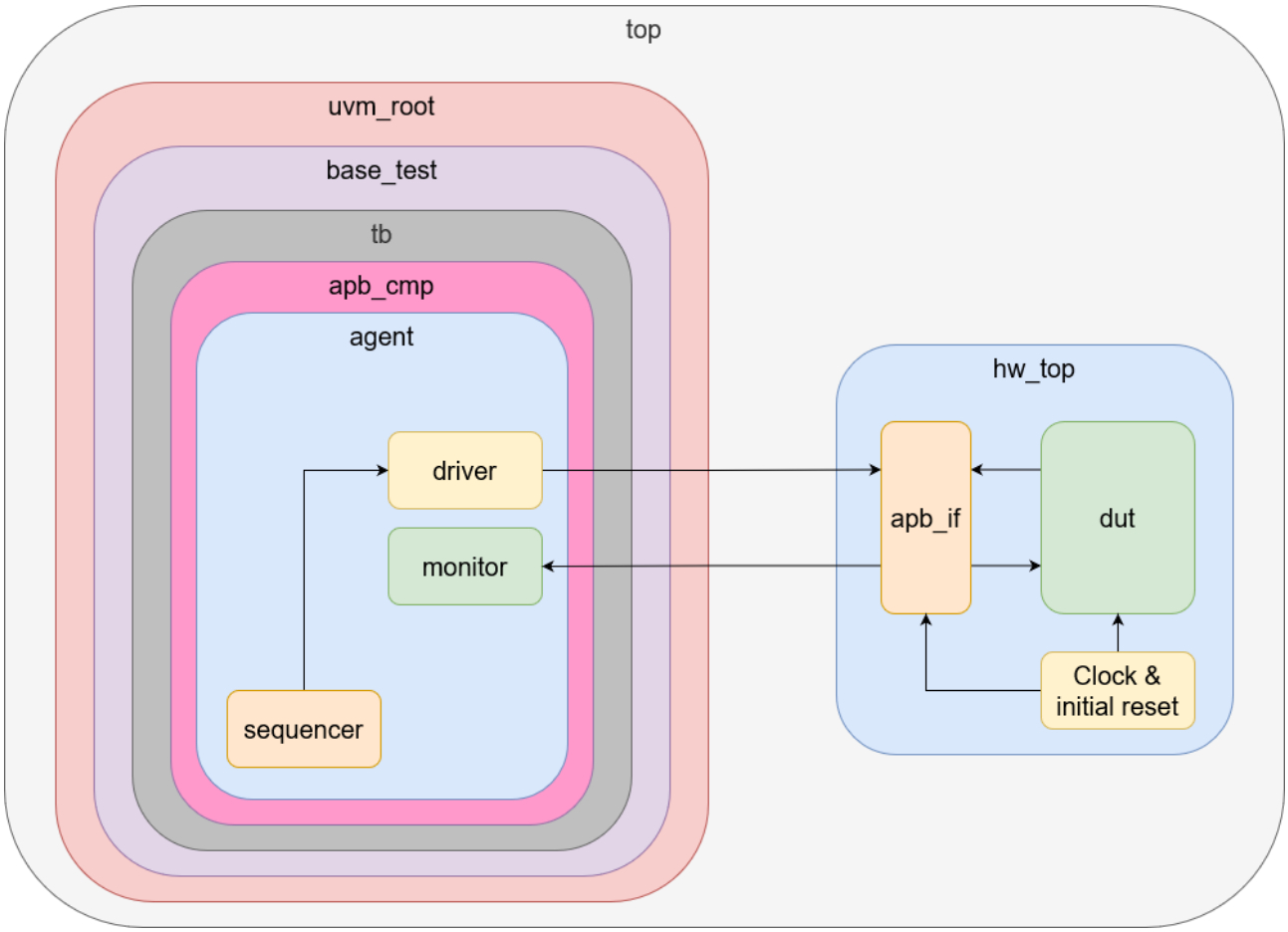
testbench.sv

UVM testbench class, it simply creates the only component instance.

top.sv

Top module file: it instantiates the hardware top, associates the only component’s driver & monitor virtual interfaces to the actual interface inside the hardware top and calls the `run_test` procedure with the base test class (the actual test to run must be defined by the `+UVM_TESTNAME` command line parameter).


apb_component_sequencer.sv
apb_component_sequence_library.sv
apb_component.sv
apb_component_driver.sv
apb_component_package.sv
apb_component_monitor.sv
apb_component_agent.sv
top.sv
apb_response.sv
testbench.sv
test_library.sv
hardware_top.sv
apb_interface.sv
apb_parameters.svh
apb_packet.sv


top
uvm_root
base_test
tb
apb_cmp
agent
driver
monitor
sequencer
hw_top
apb_if
dut
Clock &
initial reset