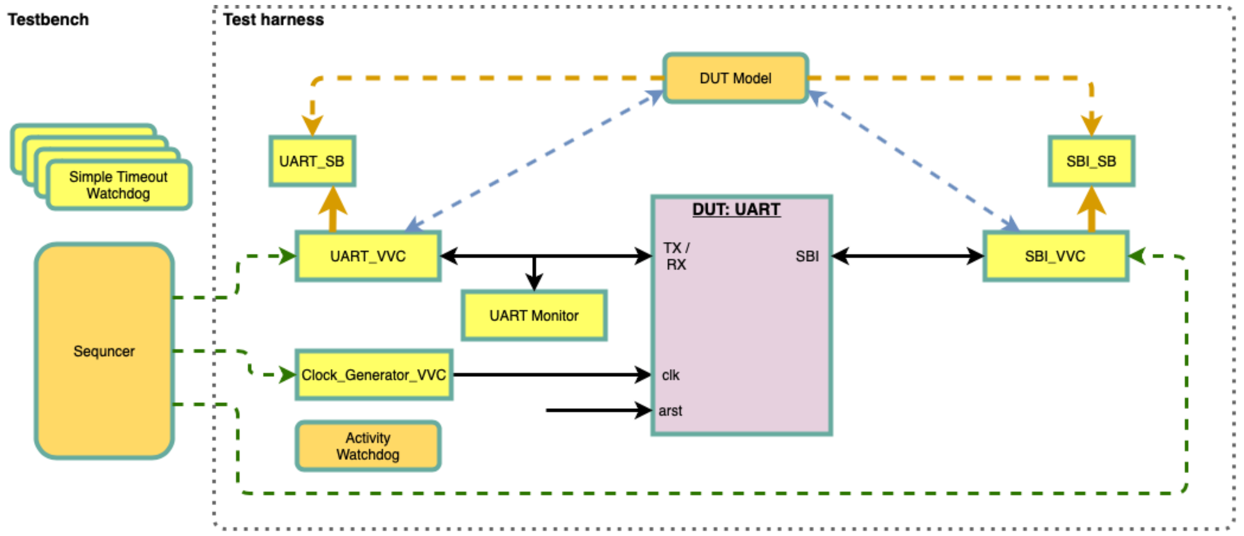
UVVM Demo Testbench Guide

This guide contains an overview of the UVVM demo testbench and demonstrates how it can be run.
In this testbench the typical VVC usage, along with some of the more advanced functionality built in to UVVM and UVVM VVC Framework, is demonstrated.
For more information see `UVVM_Essential_Mechanisms.pdf` in the `uvvm_vvc_framework/doc` folder.

Testbench Overview

The UVVM demo testbench utilizes several components in UVVM, as illustrated in Figure 1; UART, SBI and Clock Generator VVCs, UART Monitor, UART and SBI Scoreboards, Activity Watchdog, simple timeout watchdogs, DUT Model and an UART DUT. See section 1 on next page for more extensive description of the items in Figure 1.
This testbench demonstrates the usage of protocol dependent error injection, randomisation, functional coverage, protocol checker, activity watchdog and simple timeout watchdog.

Figure 1 UVVM Demo Testbench



- Green dotted lines are CDMs (command distribution methods) from the central sequencer to the VVCs.
- Blue dotted lines are DTT (direct transaction transfer) from VVCs to DUT Model.
- Black solid lines are interface connections between VVC/BFM (bus functional model) and DUT.
- Orange dotted lines are scoreboard expected receive data input.
- Orange solid lines are scoreboard actual receive data input.

1 Testbench Overview Description

Item	Location	Description
Sequencer	Testbench	Test sequencer running tests by issuing VVC commands.
Test harness	Testbench	Complete verification environment with VVCs, Model, DUT and Monitor.
UART VVC	Test harness	The UART VVC (VHDL Verification Component)
SBI VVC	Test harness	The SBI VVC (VHDL Verification Component)
UART Monitor	Test harness	The UART interface monitor reporting violations and updating the UART direct transaction transfer (DTT) signal.
DUT Model	Test harness	The UART DUT model will monitor the SBI and UART direct transaction transfer (DTT) signals and add expected data to UART or SBI scoreboard. The model is also responsible for requesting SBI VVC read commands when receiving UART DTT transmit information.
SBI Scoreboard	SBI VVC	The scoreboard for SBI read transactions. The DUT model will update this scoreboard with expected SBI read data.
UART Scoreboard	UART VVC	The scoreboard for UART receive transactions. The DUT model will update this scoreboard with expected UART receive data.
Clock Generator VVC	Test harness	The system clock generator.
DUT:UART	Test harness	The DUT with UART and SBI interfaces connected to UART VVC and SBI VVC, respectively.

2 Running Simulations

The UVVM demo testbench depend on several files from UVVM to be compiled and run, all included in the UVVM repository available on GitHub.

Note that test bench and testharness are located in the `bitvis_VIP_UART/tb` folder, and that running the testbench is done from the `bitvis_VIP_UART/sim` folder:

```
vsim -c -do ../script/compile_all_and_simulate.do
```

3 Tests

The tests are defined in the testbench file `uvvm_demo_tb.vhd`, located in `bitvis_vip_uart/tb` folder.

3.1 Error Injection Test

This test demonstrates protocol dependent error injection using the UART VVC, SBI VVC and DUT.

- The UART VVC is configured with parity and stop bit error probability from 0-100% and transmit data to DUT.
- The Model receive the UART DTT signal, put any valid transfer data on the scoreboard and issue SBI VVC read request.
- The SBI VVC will put actual received data on scoreboard. Note that UART Monitor will alert when any illegal transfer is detected.
- The sequencer presents scoreboard statistics when test is done.

3.2 Randomise Test

This test demonstrates the usage of randomisation in VVC calls using the UART VVC, SBI VVC and UART DUT.

- The UART VVC is instructed to send 1 and 3 randomised data words to the DUT.
- The Model receives UART DTT signal, puts expected data on scoreboard and issues SBI VVC read request.
- The SBI VVC puts actual data on scoreboard.
- The sequencer presents scoreboard statistics when test is done.

3.3 Functional Coverage Test

This test demonstrates the usage of functional coverage in VVC calls using the UART VVC, SBI VVC and UART DUT.

- The sequencer specifies the coverage requirement.
- The UART VVC is instructed to read data from UART DUT until coverage requirement is achieved.
- The SBI VVC is instructed to send lots of randomised data.
- The Model receives SBI DTT and put expected data on scoreboard.
- The UART VVC stops reading data when coverage requirement is achieved.
- The sequencer flushes the SBI VVC command queue, and presents coverage results and scoreboard statistics.

3.4 Protocol Checker Test

This test demonstrates the usage of protocol checkers using UART VVC, SBI VVC and UART DUT.

- The UART VVC is configured with control of a bit rate protocol checker.
- The SBI VVC transmits 6 data words.
- The bit rate protocol checker is reconfigured during the transfer of the 6 words and alerts when bit rate is not within specs.
- The Model receive SBI DTT and puts expected data on scoreboard while UART VVC puts the actual data on scoreboard.
- The sequencer presents the scoreboard statistics when test is done.

3.5 Activity Watchdog Test

This test demonstrates the usage of an activity watchdog using UART VVC, SBI VVC and UART DUT.

- The SBI VVC transmits 3 data words to the DUT, and the Model receives the SBI DTT and puts expected data on scoreboard.
- The UART VVC reads data from DUT and puts actual data on scoreboard.
- All testbench activity is stalled and activity watchdog starts a timeout countdown due to VVC inactivity, and initiates an alert when timeout is reached.
- The SBI VVC transmit 3 data words to the DUT, and the Model receive the SBI DTT and put transferred data on scoreboard.
- The UART VVC read data from DUT and put received data on scoreboard.
- The sequencer presents the scoreboard statistics when test is done.

3.6 Simple Timeout Watchdog Test

This test demonstrates the usage of a simple watchdog.

- The four watchdogs are reconfigured with a new timeout value.
- Watchdog A is given the `terminate` command and stopped.
- The test sequencer stalls for a short time and verifies that watchdog B has a timeout and alerts.
- Watchdog C is tested with `extend` and `reinitialize` commands, before sequencer stalls for a moment and verifies that watchdog C has a timeout and alerts.
- Watchdog D is tested with `reinitialize` command, before test sequencer stalls and verifies that watchdog D has a timeout and alerts.

4 Additional Documentation

Additional documentation about UVVM and its features can be found under “uvvm_vvc_framework/doc/”.