VERIFICATION ARCHITECTURE DOCUMENT for WISHBONE2UART

# EVOLUTION OF DOCUMENT

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Author** | **Modifications** |
| 0.1 | 18.08.2019 | Aleksandar Komazec | Initial release |
| 0.2 | 21.08.2019 | Aleksandar Komazec | Block diagram updated |
| 0.3 | 22.08.2019 | Aleksandar Komazec | Block diagram updated |
| 0.4 | 28.08.2019 | Aleksandar Komazec | Third chapter has been updated |

# REFERENCES

**DUT documents:**

DUT’s specification documents:

1. UART\_spec.pdf
2. UART\_Protocol.pdf

DUT’s other documents:

* 16550.pdf

Protocol specification documents:

* WISHBONE\_B4\_Protocol.pdf

Protocol other documents:

* 8251-08-serial.pdf
* serial-communication-uart.pdf

# TABLE OF CONTENTS

[EVOLUTION OF DOCUMENT 2](#__RefHeading___Toc11706_1062397625)

[REFERENCES 3](#__RefHeading___Toc11708_1062397625)

[TABLE OF CONTENTS 4](#__RefHeading___Toc11710_1062397625)

[FIGURES LIST 5](#__RefHeading___Toc11712_1062397625)

[1. VERIFICATION OVERVIEW 6](#__RefHeading___Toc11714_1062397625)

[1.1 DUT SUMMARY 6](#__RefHeading___Toc11716_1062397625)

[2. ARCHITECTURE DESCRIPTION 7](#__RefHeading___Toc11718_1062397625)

[2.1 UVC DESCRIPTION 7](#__RefHeading___Toc11720_1062397625)

[2.1.1 UART UVC DESCRIPTION 7](#__RefHeading___Toc11722_1062397625)

[2.1.2 Wisbone UVC DESCRIPTION 8](#__RefHeading___Toc11806_1062397625)

[2.2 TESTBENCH BLOCK DIAGRAM 9](#__RefHeading___Toc11728_1062397625)

[2.3 TESTBENCH COMPONENT OVERVIEW 10](#__RefHeading___Toc11730_1062397625)

[2.3. TESTBENCH CONFIGURATION AND CONTROL 10](#__RefHeading___Toc11732_1062397625)

[3. TESTBENCH EXECUTION 10](#__RefHeading___Toc11734_1062397625)

[3.1 SIMULATION FLOW DECSRIPTION 10](#__RefHeading___Toc11736_1062397625)

[3.2 REGRESSION RUNNING 11](#__RefHeading___Toc11738_1062397625)

# FIGURES LIST

Table of Figures

[Image 1: UART UVC 7](#Image 1: UART UVC |graphic)

[Image 2: Wishbone UVC 8](#Image 2: Wishbone UVC |graphic)

[Image 3: Test bench block diagram 9](#Image 3: Test bench block diagram |graphic)

## VERIFICATION OVERVIEW

### DUT SUMMARY

The UART (Universal Asynchronous Receiver/Transmitter) core provides serial communication capabilities, which allow communication with modem or other external devices, like another computer using a serial cable and RS232 protocol. This core is designed to be maximally compatible with the industry standard National Semiconductors’ 16550A device

UART feature:

* Wishbone interface (32 bit/ 8 bit data bus modes (selectable) )
* Fifo only operation
* Register level and functionality compatibility with NS16550A
* Debug interface in 32-bit data bus mode

## ARCHITECTURE DESCRIPTION

## UVC DESCRIPTION

There are two UVCs:

* UART UVC
* Wishbone UVC

### 2.1.1 UART UVC DESCRIPTION

UART UVC contains two agents (UART master agent and UART slave agent).

UART slave agent is used to read data (1 start bit , 5-8 data bits, 1 parity bit (optional), 1/1.5/2 stop bits) from uart interface ( STX\_PAD\_O) using the monitor from UART slave agent. Read data will are sent to common scoreboard (Scoreboard that is common for both UVCs) .

UART master agent is used to send data on uart interface (SRX\_PAD\_I). This agent contains its driver and sequencer because this agent drives stimulus ( on mentioned SRX\_PAD\_I as well as modem signals (RTS\_PAD\_O, DTR\_PAD\_O, CTS\_PAD\_I, DSR\_PAD\_I, RI\_PAD\_I, DCD\_PAD\_I ) ). The monitor collects data from these signals (above mentioned in this paragraph ) and sends to the common scoreboard.

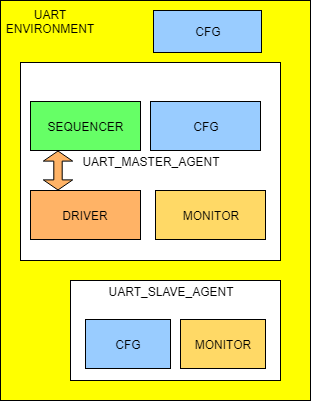
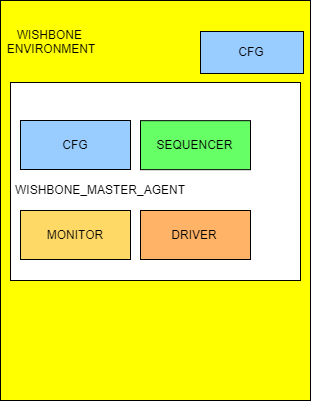
Configurations can be used to turn on and off agents.

Image 1: UART UVC

Two UART UVCs (First acts as master, and the second acts as ) can be connected to each other in order to verify UART UVC.

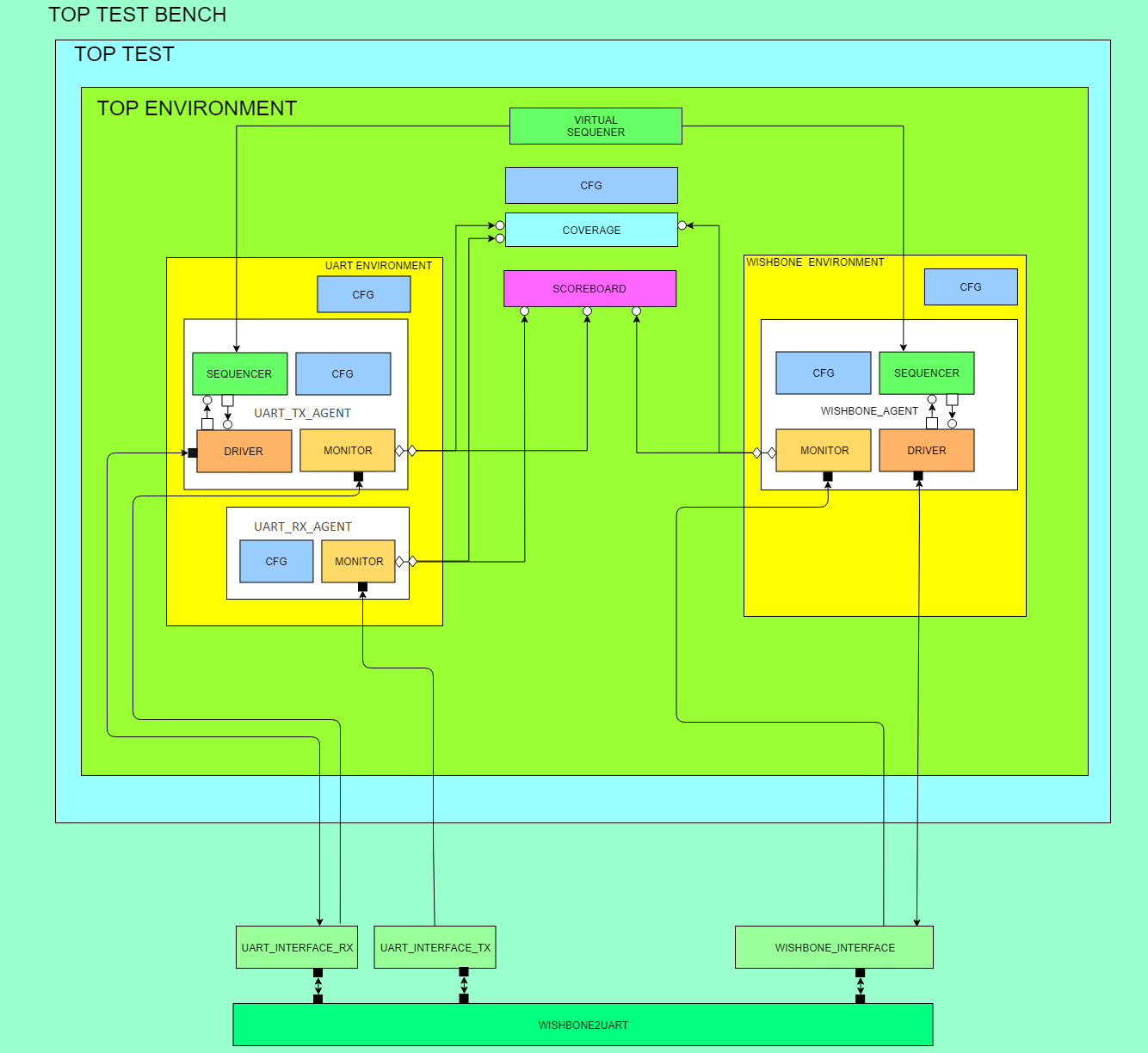
### 2.1.2 Wisbone UVC DESCRIPTION

Wishbone UVC contains 1 agent. In order to verify DUT, Wishbone agent acts as master in order to send configure DUT’s registers, send informations to Wishbone interface (WB\_DAT\_I data bus) and reads information from Wishbone interface (WB\_DAT\_O).

Image 2: Wishbone UVC

In order to verify Wishbone UVC, two Wishbone UVCs can be connected (The first Wishbone UVC is master, and the second is slave) to each other.

### TESTBENCH BLOCK DIAGRAM

Image 3: Test bench block diagram

### TESTBENCH COMPONENT OVERVIEW

*Test bench contains next components:*

* ***Wishbone UVC,*** *It consists of master agent and configuration.*
* ***UART UVC,*** It consists of master and slave agents and configuration
* ***Configuration class (top, in TEST\_BENCH),*** *This class contains pointers to UART and WB configuration classes and a configuration fields . These fields are set in the test. With these fields uart frame can be set, the data size, parity bit, number of stop bits and some other fields.*
* ***Scoreboard,*** *It receives transactions using uvm analysis ports captured by wishbone and uart monitors, stores them in the queue, predicts data and contains data checkers for comparing data from wishbone and uart interfaces.*
* ***Virtual sequencer,*** Virtual sequencer has pointers to the wishbone and uart sequencers. It can manipulates with these sequencers in order to pass all desired scenarios. One example is to use UART UVC master sequence to write to Receiver Fifo and after that Wishbone UVC master sequence to configure DUT’s registers and read data from Receiver Fifo.

### 2.3. TESTBENCH CONFIGURATION AND CONTROL

Configuration bits are:

* agent\_is\_active = If value is 1, it exists otherwise does not exist (It can be used in UART UVC to disable the slave agent when the master agent is needed)
* uvc\_is\_active = if value is 1, it UVC exists otherwise doesn’t exist.
* has\_checker =If value is 1, checkers exist otherwise they doesn’t exist. Some checkers can be disabled in order to speed up the simulation.
* has\_coverage = If value is 1, coverage exists otherwise they it doesn’t exist.
* uart\_data\_size = It can be 5,6,7 or 8 bits.
* has\_parity\_bit = Because parity bit is optional, It can be disabled or enabled it.
* number\_of\_stop\_bits = 1, 1.5, or 2 stop bits.

## TESTBENCH EXECUTION

The tool that is used for simulation running is Xcelium Parallel Logic Simulation.

The tool that is used for code coverage is Integrated Metrics Center.

Directory structure contains the main folder WB2UART.

The design is written using verilog language and the placed into rtl folder (This folder can be found in WB2UART).

Function verification files can be found into WB2UART/verif/system verilog/uvc\_lib.

There are 2 different uvcs (wishbone\_uvc and uart\_uvc).

Each of uvc folders contain :example makefile sv

In example folder:

test\_wishbone\_uvc\_base.sv,wishbone\_uvc\_env\_top.sv,test\_wishbone\_uvc\_example.sv wishbone\_uvc\_tb\_top.sv,wishbone\_uvc\_cfg\_top.sv,wishbone\_uvc\_test\_pkg.sv , wishbone\_uvc\_env\_top\_pkg.sv

In makefile folder:

Makefile,probe.tcl

In sv folder:

wishbone\_uvc\_agent\_cfg.sv,wishbone\_uvc\_if.sv,wishbone\_uvc\_agent\_slave.sv, wishbone\_uvc\_item.sv,wishbone\_uvc\_agent.sv,wishbone\_uvc\_cfg.sv wishbone\_uvc\_monitor.sv,wishbone\_uvc\_common.sv,wishbone\_uvc\_pkg.sv, wishbone\_uvc\_cov.sv,wishbone\_uvc\_driver\_slave.sv,wishbone\_uvc\_seq\_lib\_slave.sv, wishbone\_uvc\_driver.sv,wishbone\_uvc\_seq\_lib.sv,wishbone\_uvc\_env.sv,wishbone\_uvc\_sequencer.sv

*Top environment can be found at \wb2uart\verif\systemverilog\env*

*There are files:*

*wb2uart\_common.sv,wb2uart\_env\_top.sv,wb2uart\_env\_top\_pkg.sv, wb2uart\_scoreboard.sv, wb2uart\_top\_cfg.sv, wb2uart\_virtual\_sequencer.sv*

*Virtual sequences can be found at \wb2uart\verif\systemverilog\env\virt\_seq\_lib:*

*wb2uart\_virtual\_seq\_register\_access.sv,wb2uart\_virtual\_sequence\_base.sv*

*wb2uart\_virtual\_sequence\_lib.sv*

*Top module can be found at /wb2uart\verif\systemverilog\tb\wb2uart\_tb\_top.sv*

*Test can be found at \akomazec\wb2uart\verif\systemverilog\tests:*

*wb2uart\_test\_pkg.sv,wb2uart\_test\_base.sv,wb2uart\_register\_access\_test\_tc.sv*

### SIMULATION FLOW DESCRIPTION

Script that is used for compilation and simulation running is Makefile script

Make file contains next commands:

run\_gui :

* $(SIM) is irun, irun is a command that run step by step compilation hdl, systemC(ncvlog, ncvhdl, ncsc), elaboration(ncelab) and simulation (ncsim)
* $(INCDIRS) to include funtional verification sv files (sv and example)
* $(PACKAGES) to include packages
* $(OPTIONS) in options verbosity,seed,access,timescale, destination of design under test can be changed
* $(COVERAGE\_OPTIONS) options related to coverage
* $(DEBUG\_OPTIONS) options related to debug
* -gui gui mode provides many options in order to debug

run\_batch :

* $(SIM) $(INCDIRS) $(PACKAGES) $(OPTIONS) $(COVERAGE\_OPTIONS) (DEBUG\_OPTIONS)

Explanation is same as the above

* $(RUN\_TCL) runs probe.tcl script (This script create top test bench file)

clean :

* rm -rf INCA\* \*.key \*.log \*.diag default\* cov\_work \*.err .simvis\* \*.history \*xcelium\*

It deletes all files above (.key, .log, …)

help : This command prints Makefile help menu in order to help user how to use other commands.

### REGRESSION RUNNING

*Description of the regression running, simulation results evaluation, and regression report generation.*