2021-2022

AXI4-AVIP

USER GUIDE

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 AXI Read and Write Channels

The AXI protocol defines 5 channels:

- 2 are used for Read transactions
  - read address
  - read data
- 3 are used for Write transactions
  - Write address
  - Write data
  - Write response

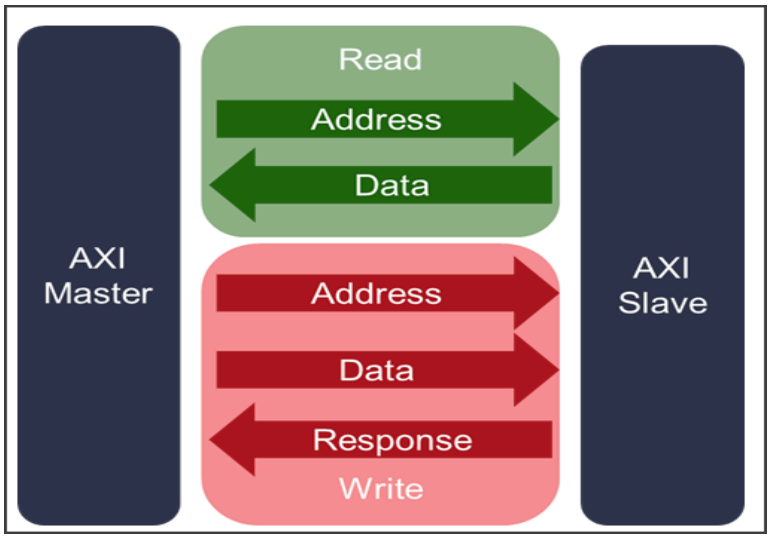


Fig 1.1 AXI Read and write Channels

The AXI is a burst-based protocol that defines the following transaction channels independently:

1. Address Read(AR)
2. Read data(R)
3. Address Write(AW)
4. Write data(W)
5. Write response(B)

1.2 AXI Read Transactions

An AXI Read transaction requires multiple transfers on the 2 Read channels.

- First, the **Address Read Channel** is sent from the Master to the Slave to set the address and some control signals.
- Then the data for this address is transmitted from the Slave to the Master on the **Read data channel**.

Note that, as per the figure below, there can be multiple data transfers per address. This type of transaction is called a **burst**.

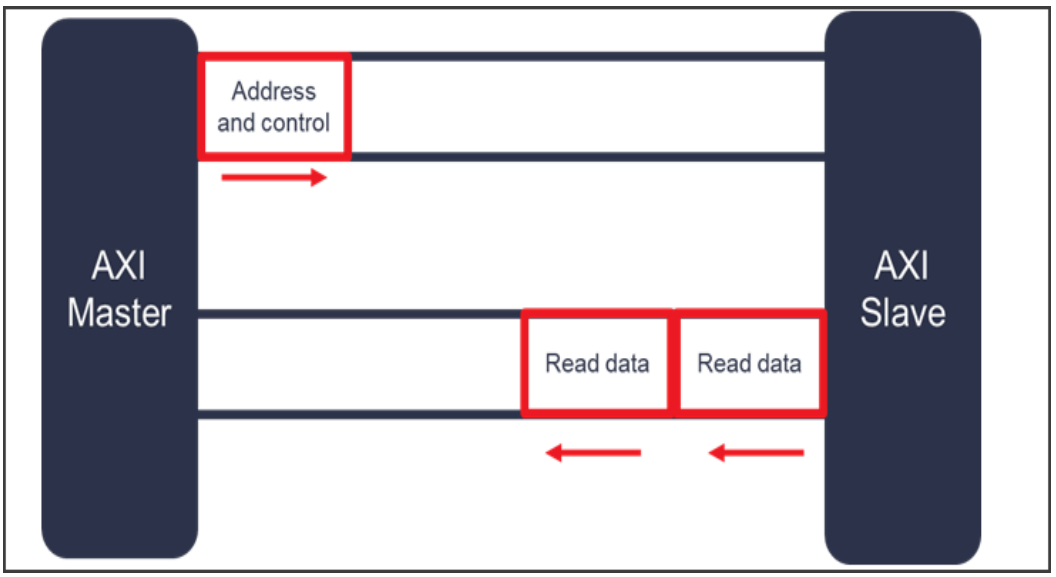


Fig 1.2 Read Channels of AXI

1.3 AXI Write Transactions

An AXI Write transaction requires multiple transfers on the 3 Read channels.

- First, the **Address Write Channel** is sent Master to the Slave to set the address and some control signals.
- Then the data for this address is transmitted Master to the Slave on the **Write data channel**.
- Finally the write response is sent from the Slave to the Master on the **Write Response Channel** to indicate if the transfer was successful.

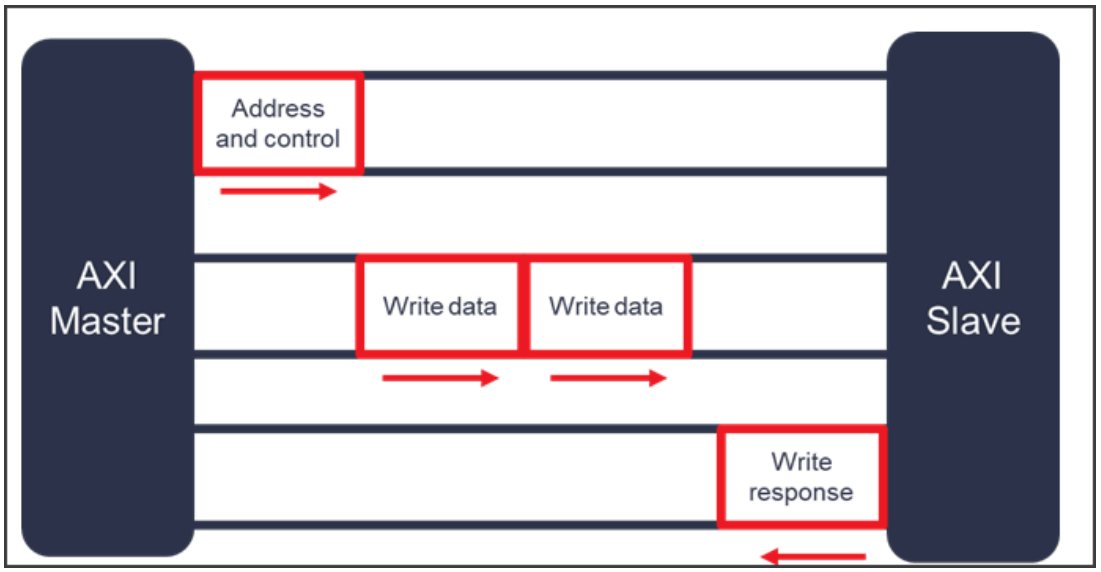


Fig 1.3 Write Channels of AXI

The possible response values on the Write Response Channel are:

- OKAY (0b00): Normal access success. Indicates that normal access has been successful
- EXOKAY (0b01): Exclusive access okay.
- SLVERR (0b10): Slave error. The slave was reached successfully but the slave wishes to return an error condition to the originating master (for example, data read not valid).
- DECERR (0b11): Decode error. Generated, typically by an interconnect component, to indicate that there is no slave at the transaction address

Note: Read transactions also have a response value but this response is transmitted as part of the Read Response Channel

1.4 axi4_compile.f file

This file contains the following things:

1. All the directories needed
2. All the packages we needed
3. All the modules are written
4. All the bfm interfaces are written
5. The axi4 interface

- ❖ If you want to add any file in the project, please add the file or folder in axi4_compile.f file to make it compiled.
- ❖ If you want to add a class-based component or object, you have to add that file in the respective package file and then make sure to mention the directory and path in axi4_compile.f file.
- ❖ If you want to add a module/interface or any static component, then mention the file name along with the path of the file.
- ❖ How to add:
    1. To include directory: +incdir+<path_of_the_folder>
    2. To include the file use file_path/file_name.extension
    3. / is used to force a new line
- ❖ Current axi4_compile.f file consists of all files directories and Packages mentioned in fig. 1.2, fig. 1.3 and fig. 1.4.

a. Directories included :

```
+incdir+../../src/globals/
+incdir+../../src/hvl_top/test/sequences/master_sequences/
+incdir+../../src/hvl_top/master/
+incdir+../../src/hdl_top/master_agent_bfm/
+incdir+../../src/hvl_top/env/virtual_sequencer/
+incdir+../../src/hvl_top/test/virtual_sequences/
+incdir+../../src/hvl_top/env
+incdir+../../src/hvl_top/slave
+incdir+../../src/hvl_top/test/sequences/slave_sequences/
+incdir+../../src/hvl_top/test
+incdir+../../src/hdl_top/slave_agent_bfm
+incdir+../../src/hdl_top/axi4_interface
```

Fig: 1.4 Directories included in axi4_compile.f file

b. Packages included:

```
../../src/globals/axi4_globals_pkg.sv
../../src/hvl_top/master/axi4_master_pkg.sv
../../src/hvl_top/slave/axi4_slave_pkg.sv
../../src/hvl_top/test/sequences/master_sequences/axi4_master_seq_pkg.sv
../../src/hvl_top/test/sequences/slave_sequences/axi4_slave_seq_pkg.sv
../../src/hvl_top/env/axi4_env_pkg.sv
../../src/hvl_top/test/virtual_sequences/axi4_virtual_seq_pkg.sv
../../src/hvl_top/test/axi4_test_pkg.sv
```

Fig: 1.5 Packages included in axi4_compile.f file

c. Static files included :

```
../../src/hdl_top/axi4_interface/axi4_if.sv
../../src/hdl_top/master_agent_bfm/axi4_master_driver_bfm.sv
../../src/hdl_top/master_agent_bfm/axi4_master_monitor_bfm.sv
../../src/hdl_top/master_agent_bfm/axi4_master_agent_bfm.sv
../../src/hdl_top/slave_agent_bfm/axi4_slave_driver_bfm.sv
../../src/hdl_top/slave_agent_bfm/axi4_slave_monitor_bfm.sv
../../src/hdl_top/slave_agent_bfm/axi4_slave_agent_bfm.sv
../../src/hdl_top/hdl_top.sv
../../src/hvl_top/hvl_top.sv
../../src/hdl_top/tb_master_assertions.sv
../../src/hdl_top/tb_slave_assertions.sv
../../src/hdl_top/master_assertions.sv
../../src/hdl_top/slave_assertions.sv
```

Fig: 1.6 static files included in axi4_compile.f file

In Makefile, we include the axi4_compile.f file to compile all the files included.
A command used: *irun -f axi4_compile.f +UVM_TEST_NAME=<test_name> +uvm_verbosity=UVM_HIGH.*

```
compile:
  make clean_compile;
  make clean_simulate;
  vlib work;
  vlog -sv \
  +acc \
  +cover \
  +fcover \
  -l axi4_compile.log \
  -f ../axi4_compile.f
```

Fig: 1.7 axi4_compile.f used in makefile for questa_sim tool

Chapter 2

ARCHITECTURE

AXI4 AVIP Testbench Architecture

The accelerated VIP has been divided into the two top modules as HVL and HDL top as shown in fig 2.1. The whole idea of using Accelerated VIP is to push the synthesizable part of the testbench into the separate top module along with the interface and it is named HDL TOP. and the unsynthesizable part is pushed into the HVL TOP it provides the ability to run the longer tests quickly. This particular testbench can be used for the simulation as well as the emulation based on mode of operation.

HVL TOP has the design which is untimed and the transactions flow from both master virtual sequence and slave virtual sequence onto the AXI4 I/F through the BFM Proxy and BFM and gets the data from monitor BFM and uses the data to do checks using scoreboard and coverage

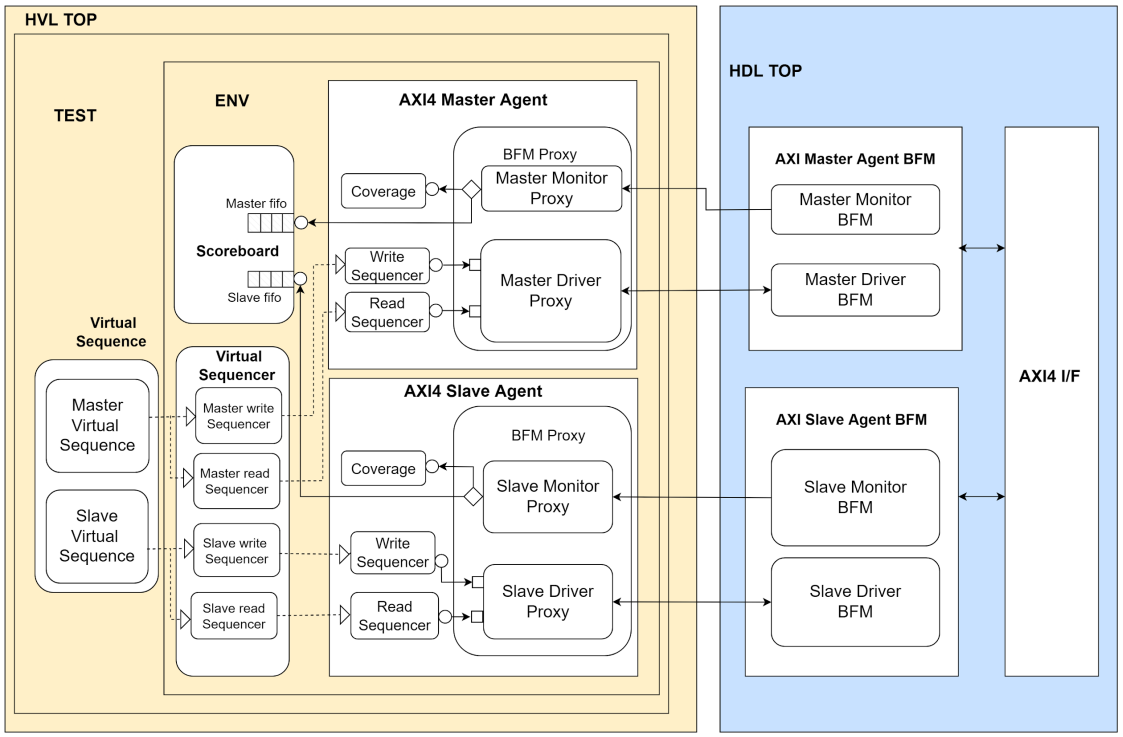


Fig 2.1 AXI4 AVIP Testbench Architecture

HDL TOP consists of the design part which is timed and synthesizable, Clock and reset signals are generated in the HDL TOP. Bus Functional Models (BFMs) i.e. synthesizable

parts of drivers and monitors are present in HDL TOP, BFMs also have the back pointers to their proxy to call non-blocking methods which are defined in the proxy.

We have the tasks and functions within the drivers and monitors which are called by the driver and monitor proxy inside the HVL. This is how the data is transferred between the HVL TOP and HDL TOP.

HDL and HVL use transaction-based communication to enable the information-rich transactions and since the clock is generated within the HDL TOP inside the emulator it allows the emulator to run at full speed.

Chapter 3

Steps to run Test Cases

3.1 Git steps

1. Checking for git, open the terminal type the command

   git version

   The output will either tell you which version of Git is installed or alert you that git is an unknown command. If it's an unknown command, install Git using the following link guide to install git in other platforms
2. Copy the ssh public key and do the clone of the axi4_avip repository in the terminal

   git@github.com:*mirafra-software-technologies*/axi4_avip.git

   git clone git@github.com:mirafra-software-technologies/axi4_avip.git

   After cloning, change the directory to the cloned repository

   cd axi4_avip
3. After cloning you will be in the main branch i.e, the production branch

   git branch
4. Do the pull for the cloned repository to be in sync CD

   git pull origin main
5. Fetch all branches in the spi_avip repository

   git fetch
6. Check all branches present in the spi_avip repository

   git branch -a
7. To switch from the main branch to another branch

   git checkout origin <branch_name>
8. Do the pull for the cloned repository to be in sync

   git pull origin <branch_name>

Note: To run any test case you should be inside the cloned directory i.e, axi4_avip [axi4_avip is considered as root path]

3.2 Mentor's Questasim

1. Change the directory to questasim directory where the makefile is present

   Path for the mentioned directory is spi_avip/sim/questasim

   Note: To Compile, simulate, regression and for coverage you must be in the specified path i.e, *axi4_avip/sim/questasim*

2. To view the usage for running test cases, type the command

   make

   Fig 3.1 shows the usage to compile, simulate, and regression

```
To compile use:
make compile

To simulate individual test:
make simulate test=<test_name> uvm_verbosity=<VERBOSITY_LEVEL>

Example::
make simulate test=base_test uvm_verbosity=UVM_HIGH

To run regression:
make regression testlist_name=<regression_testlist_name.list>

Example::
make regression testlist_name=axi4_transfers_regression.list
```

Fig 3.1 Usage of the make command

3.2.1 Compilation

1. Use the following command to compile

   make compile

2. Open the log file axi4_compile.log to view the compiled files
   gvim axi4_compile.log

3.2.2 Simulation

1. After compilation, use the following command to simulate individual test cases

   make simulate test=<test_name> uvm_verbosity=<VERBOSITY_LEVEL>

   Example:
   make simulate test=axi4_blocking_8b_write_read_test uvm_verbosity=UVM_HIGH
   Note: You can find all the test case names in the path given below *axi4_avip/src/hvl_top/testlists/axi4_transfers_regression.list*

2. To view the log file
   gvim <test_name>/<test_name>.log

   Ex: *gvim axi4_blocking_8b_write_read_test/axi4_blocking_8b_write_read_test.log*

   Note: The path for the log file will be displayed in the simulation report along with the name of the simulated test

3. To view waveform

   vsim -view <test_name>/waveform.wlf &

   Ex: *vsim -view axi4_blocking_8b_write_read_test/waveform.wlf &*

   Note: The command to view the waveform will be displayed in the simulation report along with the name of the simulated test

4. As you run the above command, the new WLF Questasim window will appear as shown in fig 3.2

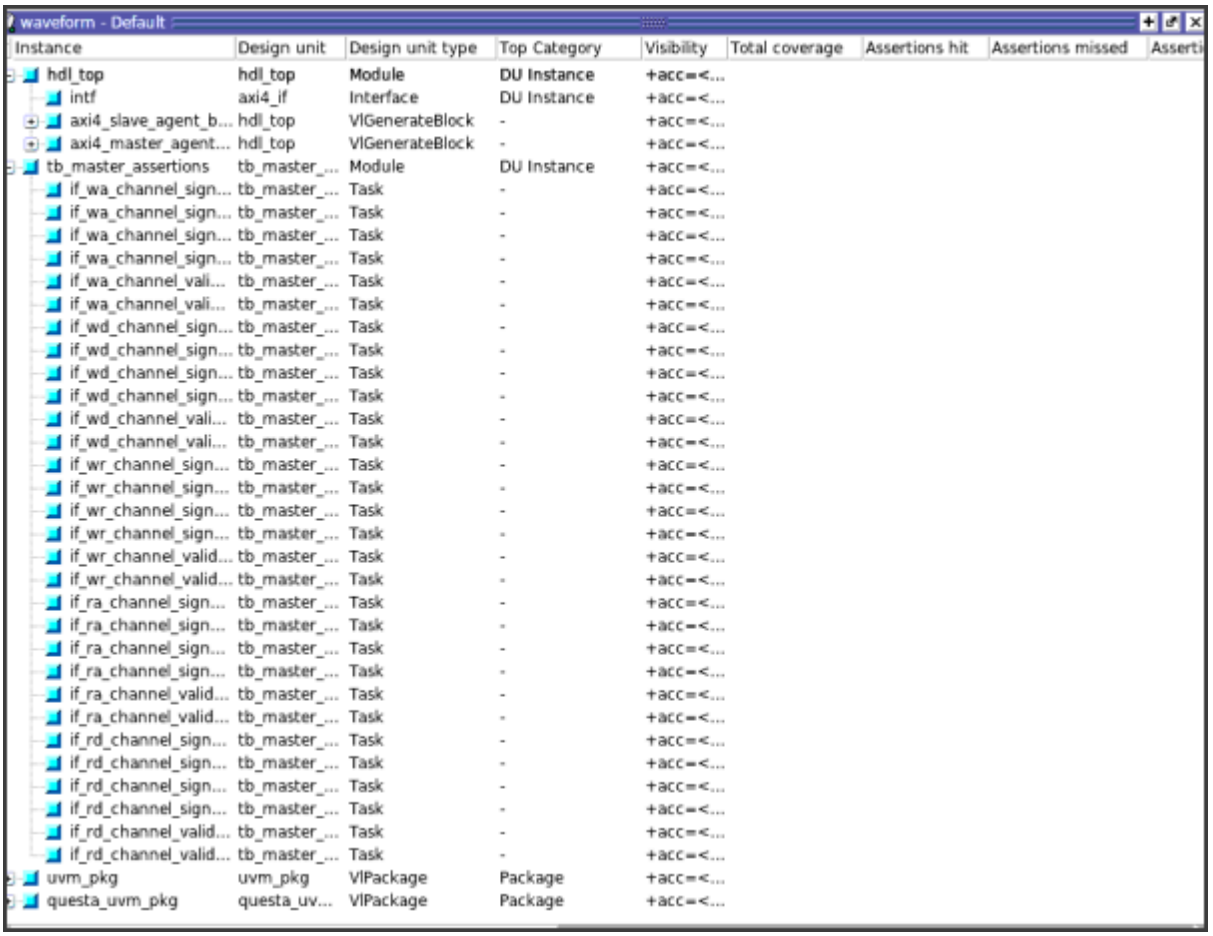


Fig 3.2 Questasim WLF window

5. Right-click on **intf** and select Add Wave as shown in image 3.3 to add the signals to the wave window

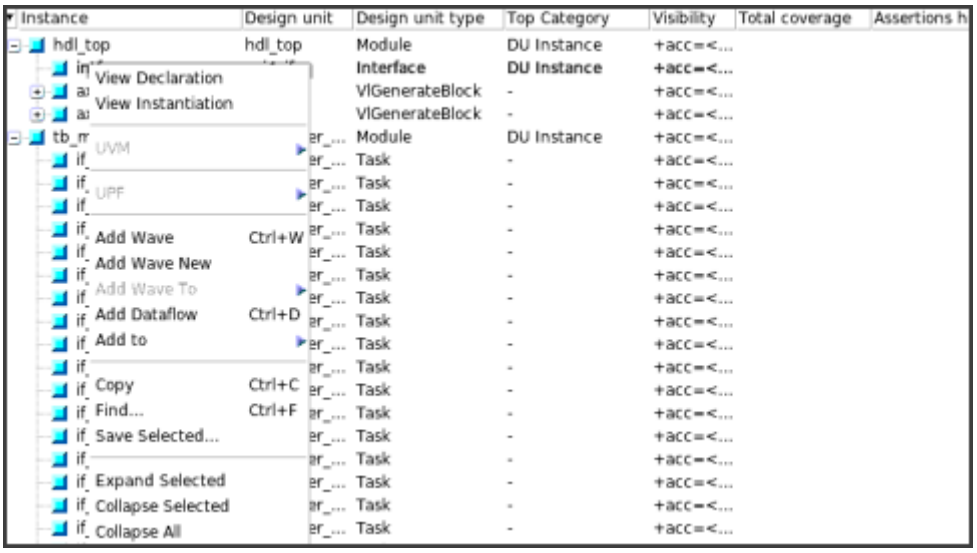


Fig 3.3 Screenshot of adding waves in wave window

6. After adding wave, click on Wave window as shown in the Fig 3.3

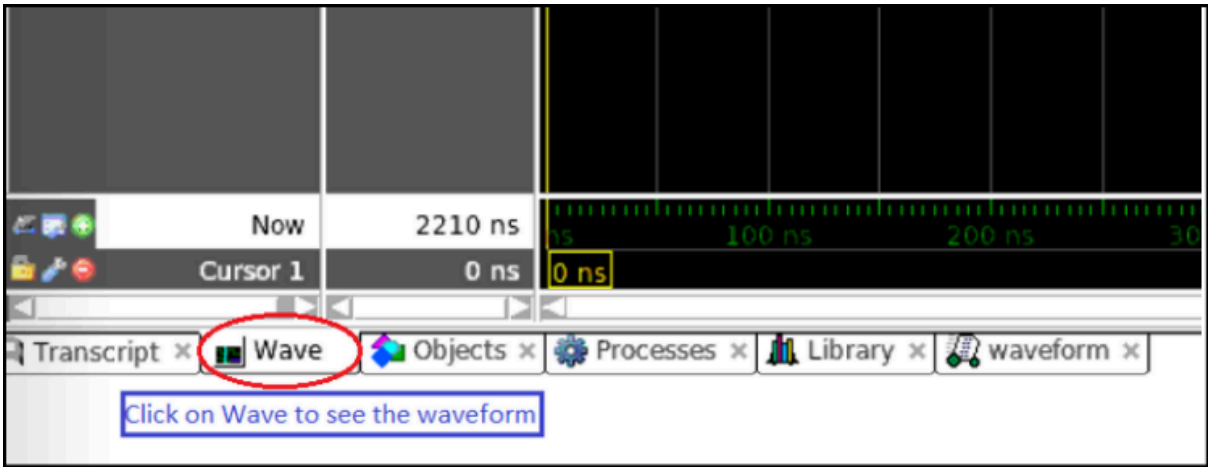


Fig 3.4 Wave window

7. Click as shown in fig 3.4 to unlock the waveform window

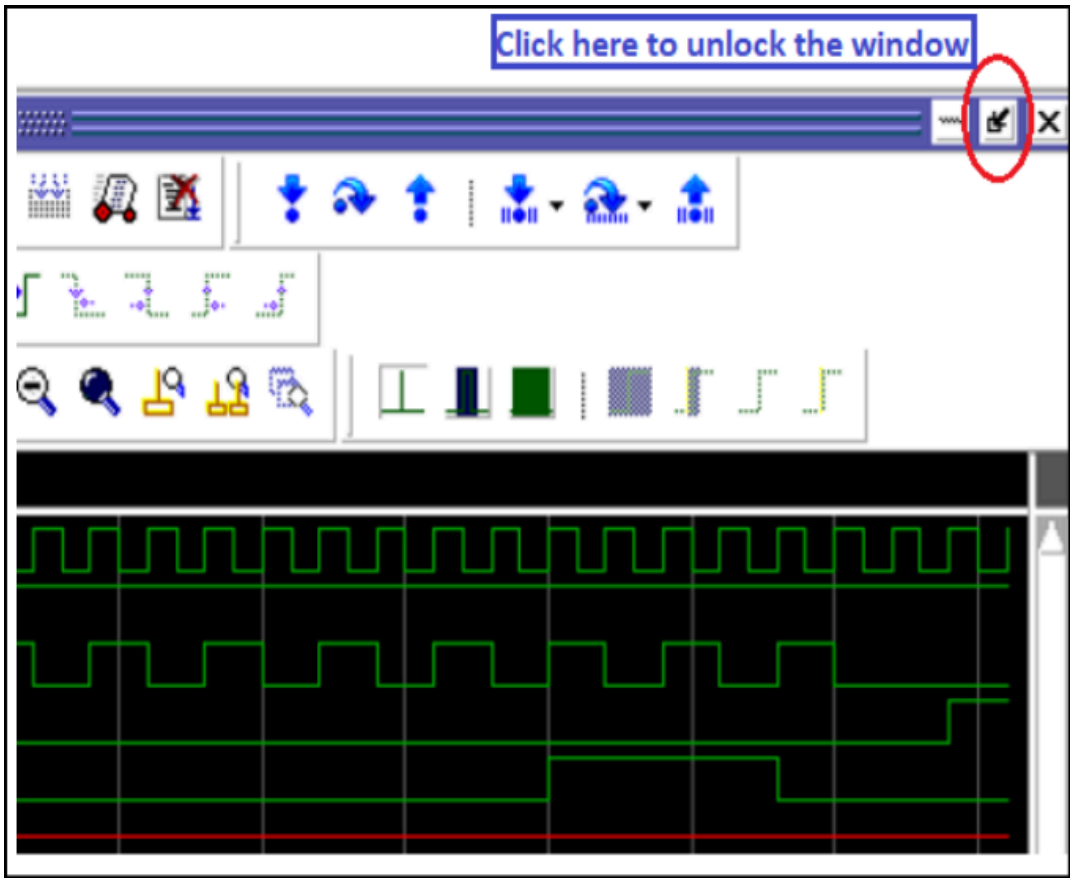


Fig 3.5 Screenshot of unlocking the wave window

8. You will be able to get a separate wave window as shown in the Fig 3.5

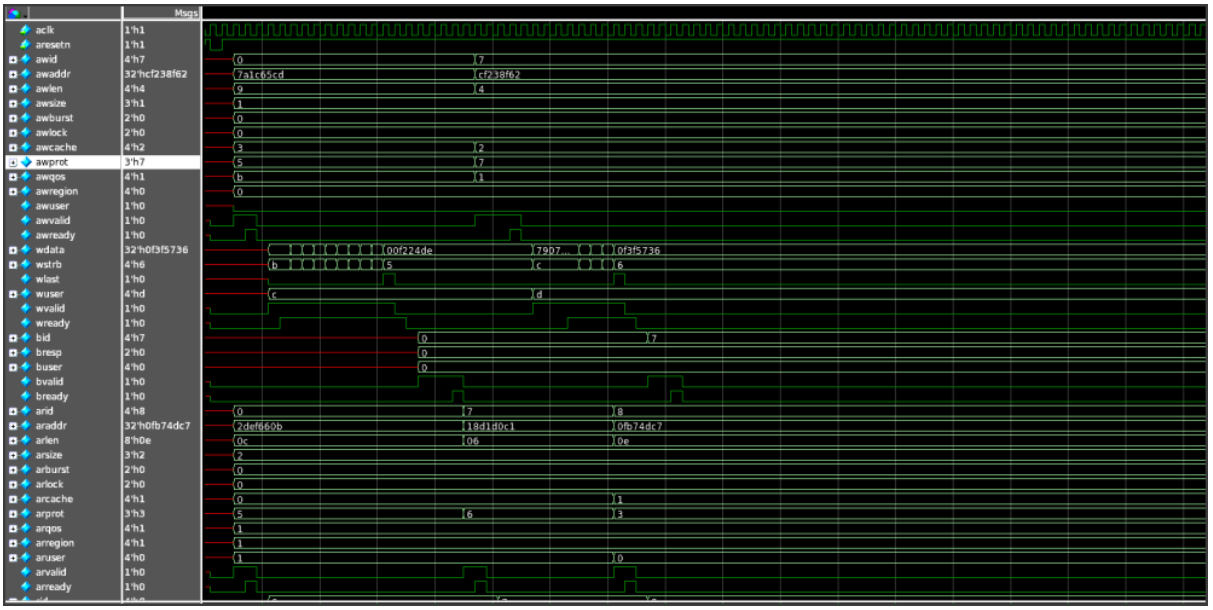


Fig 3.6 Screenshot of the unlocked wave window with signals

9. Click on the icon signal toggle leaf name marked in fig 3.6 to see the signals as shown

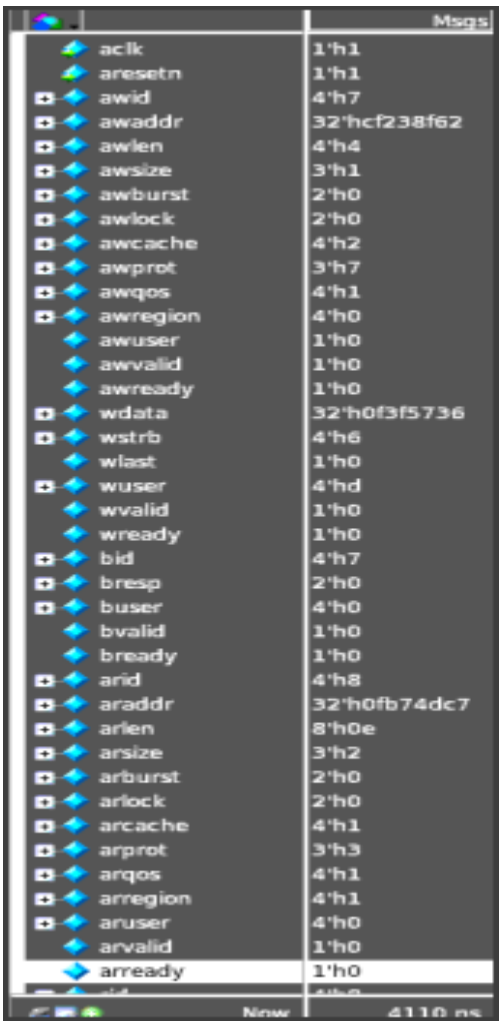


Fig 3.7 Screenshot showing way to see the name of signals

10. For the analysis of waveform, go through the link below

    Waveform Viewer

3.2.3 Regression

1. To run regression for all test cases

   make regression testlist_name=<regression_testlist_name.list>

   Ex: *make regression testlist_name=axi4_transfers_regression.list*

   Note: You can find all the test case names in the path given below
   axi4_avip/src/hvl_top/testlists/axi4_transfers_regression.list

2. After regression, you can view the individual files as shown fig 3.7

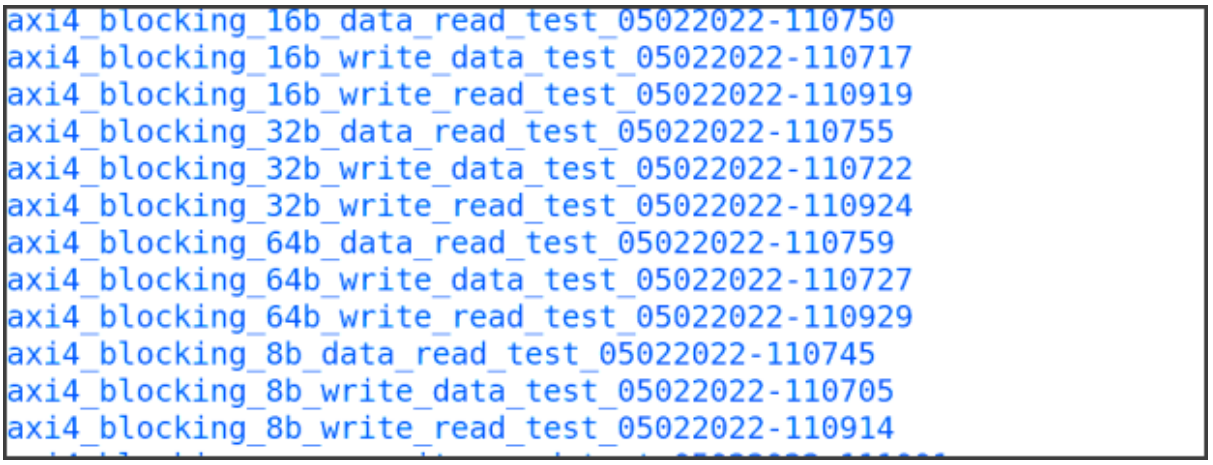


Fig 3.8 Files in questasim after the regression

3. To view the log files of individual tests, select the interested test case file, go inside that directory

   Ex: Interested in the test case *axi4_blocking_8b_write_read_test*

   Go inside the directory of interested testcase with the date

   axi4_blocking_8b_write_read_test_05022022-110914

   Inside this directory, you will be able to find the log file of the interested test case

   axi4_blocking_8b_write_read_test.log

   Path:
axi4_blocking_8b_write_read_test_05022022-110914/axi4_blocking_8b_write_read_test.log

3.2.4 Coverage

1. To see coverage

a. After simulating

For the individual test, use the command firefox

axi4_blocking_8b_write_read_test/html_cov_report/index.html &

Ex: *firefox axi4_blocking_8b_write_read_test/html_cov_report/index.html &*

Note: The command to see the coverage will be displayed in the simulation report along with the name of the simulated test

b. **After the regression,**

- To view the coverages of all test cases, type the below command

```
firefox merged_cov_html_report/index.html &
```

firefox merged_cov_html_report/index.html &

Note: The command to see the coverage will be displayed in the simulation report along with the name of the simulated test

- To view the coverage for individual test cases

  See the list of files generated after regression, which is shown in fig 3.7.

  Select the interested test case file, go inside that directory

  Ex:

  Interested in the test case *axi4_blocking_8b_write_read_test*

  Go inside the directory of the interested test case with the date

  axi4_blocking_8b_write_read_test_05022022-110914

  Inside this directory, you will be able to find the html coverage file of the interested test case

  html_cov_report/

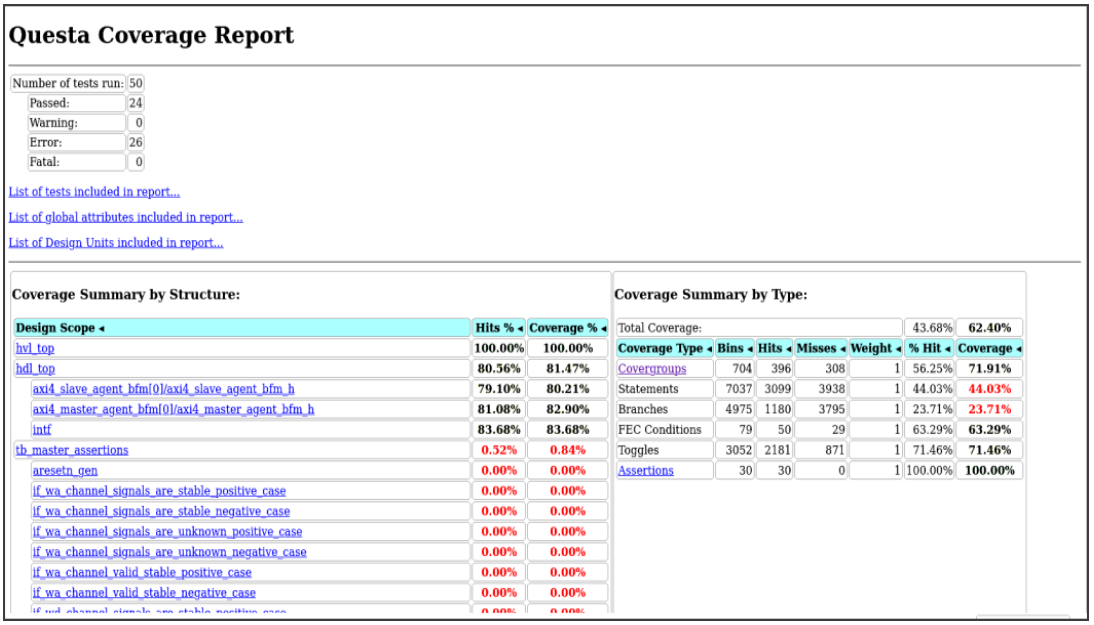
  Inside it would be the HTML file

  covsummary.html

  Command to view coverage report for the above test case will be

  firefox axi4_blocking_8b_write_read_test_05022022-110914/html_cov_report/covsummary.html

2. The coverage report window appears as shown in fig 3.8



Questa Coverage Report

Number of tests run:	50
Passed:	24
Warning:	0
Error:	26
Fatal:	0

List of tests included in report...
List of global attributes included in report...
List of Design Units included in report...

Coverage Summary by Structure:

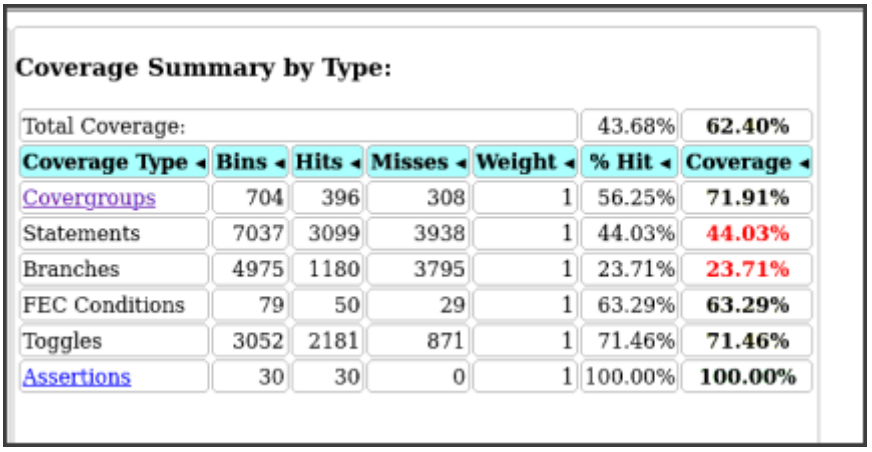
Design Scope	Hits %	Coverage %
hvl_top	100.00%	100.00%
hdl_top	80.56%	81.47%
axi4_slave_agent_bfm[0]/axi4_slave_agent_bfm_h	79.10%	80.21%
axi4_master_agent_bfm[0]/axi4_master_agent_bfm_h	81.08%	82.90%
intf	83.68%	83.68%
tb_master_assertions	0.52%	0.84%
aresetn_gen	0.00%	0.00%
if_wa_channel_signals_are_stable_positive_case	0.00%	0.00%
if_wa_channel_signals_are_stable_negative_case	0.00%	0.00%
if_wa_channel_signals_are_unknown_positive_case	0.00%	0.00%
if_wa_channel_signals_are_unknown_negative_case	0.00%	0.00%
if_wa_channel_valid_stable_positive_case	0.00%	0.00%
if_wa_channel_valid_stable_negative_case	0.00%	0.00%

Coverage Summary by Type:

Total Coverage:					43.68%	62.40%
Coverage Type	Bins	Hits	Misses	Weight	% Hit	Coverage
Covergroups	704	396	308	1	56.25%	71.91%
Statements	7037	3099	3938	1	44.03%	44.03%
Branches	4975	1180	3795	1	23.71%	23.71%
FEC Conditions	79	50	29	1	63.29%	63.29%
Toggles	3052	2181	871	1	71.46%	71.46%
Assertions	30	30	0	1	100.00%	100.00%

Fig 3.9 Coverage Report

3. Scroll down to the coverage summary by type and click on covergroups shown in fig 3.9.

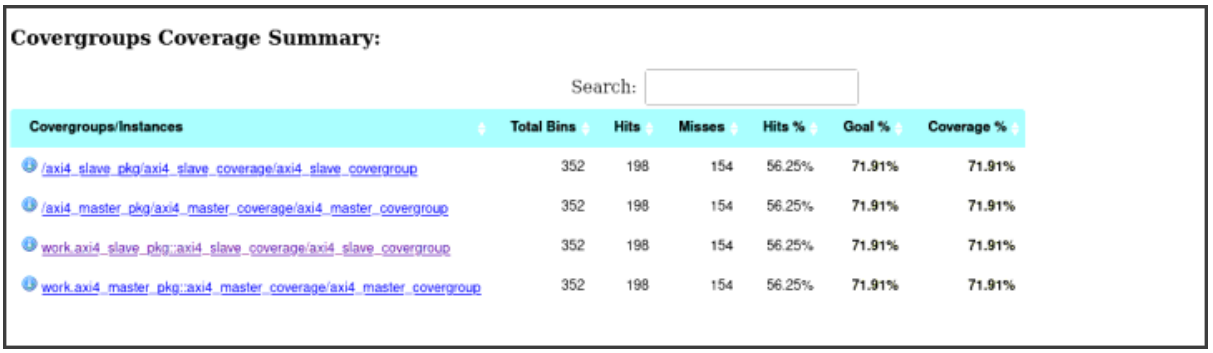


Coverage Summary by Type:

Total Coverage:					43.68%	**62.40%**
Coverage Type	**Bins**	**Hits**	**Misses**	**Weight**	**% Hit**	**Coverage**
Covergroups	704	396	308	1	56.25%	**71.91%**
Statements	7037	3099	3938	1	44.03%	**44.03%**
Branches	4975	1180	3795	1	23.71%	**23.71%**
FEC Conditions	79	50	29	1	63.29%	**63.29%**
Toggles	3052	2181	871	1	71.46%	**71.46%**
Assertions	30	30	0	1	100.00%	**100.00%**

3.10 Screenshot of opening covergroups

4. After opening coevrgroup you will be able to see the summary.click on as shown in the fig 3.10 to master and slave covergroup



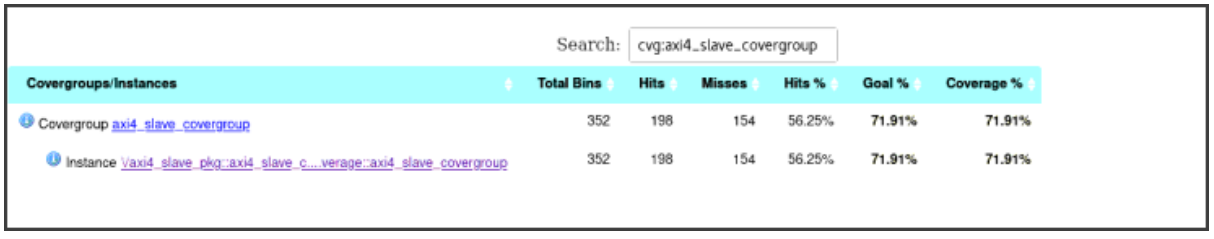
Covergroups Coverage Summary:

Search:

Covergroups/Instances	Total Bins	Hits	Misses	Hits %	Goal %	Coverage %
/axi4_slave_pkg/axi4_slave_coverage/axi4_slave_covergroup	352	198	154	56.25%	71.91%	71.91%
/axi4_master_pkg/axi4_master_coverage/axi4_master_covergroup	352	198	154	56.25%	71.91%	71.91%
work.axi4_slave_pkg::axi4_slave_coverage/axi4_slave_covergroup	352	198	154	56.25%	71.91%	71.91%
work.axi4_master_pkg::axi4_master_coverage/axi4_master_covergroup	352	198	154	56.25%	71.91%	71.91%

3.11 Screenshot of opening slave covergroup

5. If clicked on slave covergroup, further it opens to another window, again click on the slave covergroup as shown in fig 3.11

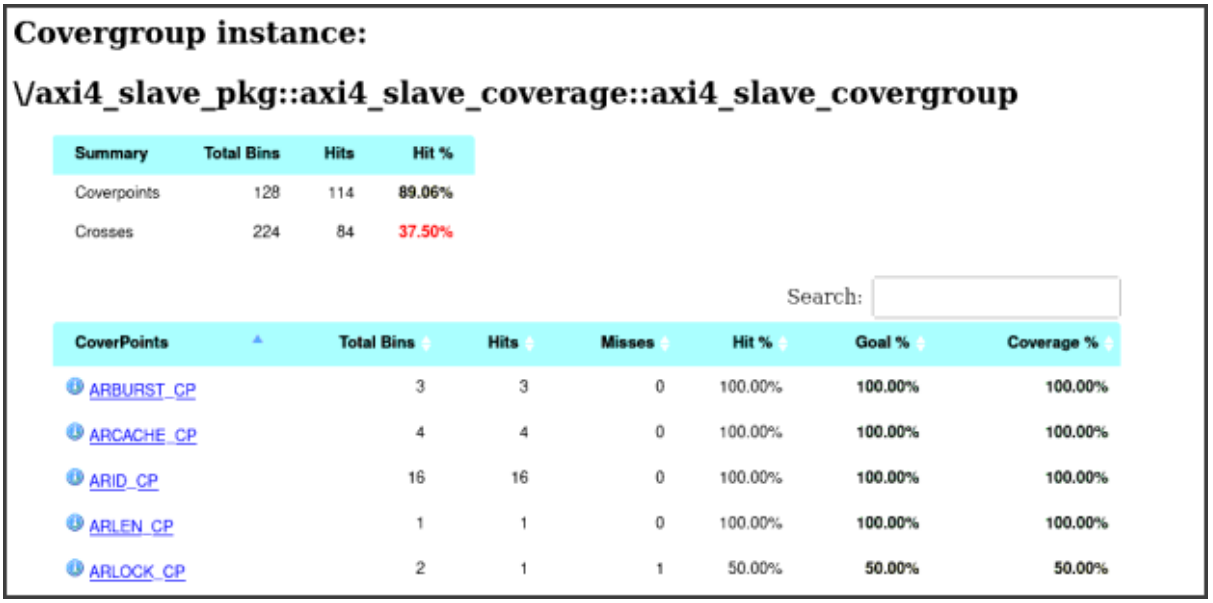


Search: cvg:axi4_slave_covergroup

Covergroups/Instances	Total Bins	Hits	Misses	Hits %	Goal %	Coverage %
Covergroup axi4_slave_covergroup	352	198	154	56.25%	71.91%	71.91%
Instance \/axi4_slave_pkg::axi4_slave_c...verage::axi4_slave_covergroup	352	198	154	56.25%	71.91%	71.91%

3.12 Shows way to open slave covergroup coverage report

6. Further, you will be able to see coverpoints and crosses as shown in fig 3.12



Covergroup instance:

\/axi4_slave_pkg::axi4_slave_coverage::axi4_slave_covergroup

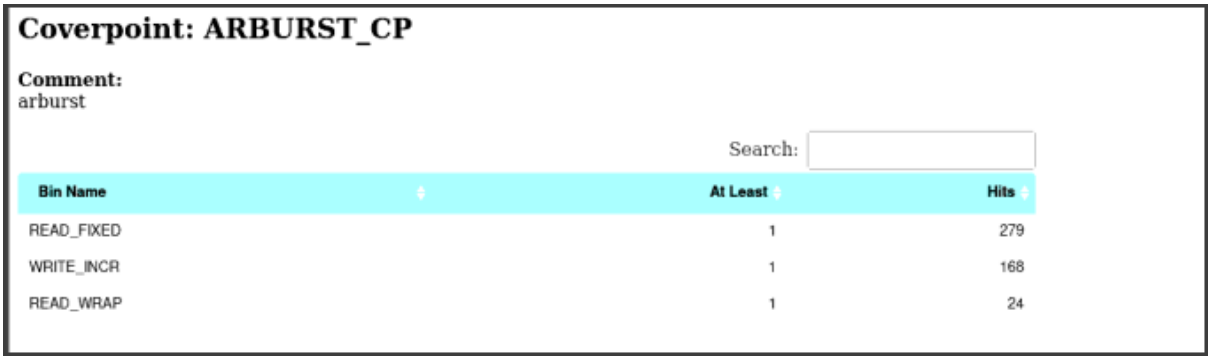
Summary	Total Bins	Hits	Hit %
Coverpoints	128	114	89.06%
Crosses	224	84	37.50%

Search:

CoverPoints	Total Bins	Hits	Misses	Hit %	Goal %	Coverage %
ARBURST_CP	3	3	0	100.00%	100.00%	100.00%
ARCACHE_CP	4	4	0	100.00%	100.00%	100.00%
ARID_CP	16	16	0	100.00%	100.00%	100.00%
ARLEN_CP	1	1	0	100.00%	100.00%	100.00%
ARLOCK_CP	2	1	1	50.00%	50.00%	50.00%

Fig 3.13 Screenshot of Coverpints and cross cover points

7. Click on individual coverpoints and crosses to see the bins hit, here AR_BURST_CP is an individual coverpoint in Fig 3.13



Coverpoint: ARBURST_CP

Comment:
arburst

Search:

Bin Name	At Least	Hits
READ_FIXED	1	279
WRITE_INCR	1	168
READ_WRAP	1	24

Fig. 3.14 ARBURST_CP coverpoint report

8. For the analysis of the coverage report, click on the link Coverage Debug

Chapter 4

Debug Tips

As design complexity continues to increase, it is contributing to new challenges in verification and debugging. Fortunately, new solutions and methodologies (such as UVM) have emerged to address growing design complexity. Yet, even with the productivity gains that can be achieved with the adoption of UVM, newer debugging challenges specifically related to UVM need to be addressed.

Here **axi4_blocking_32b_write_read_test** has been used as an example test case in order to show the below debugging flow of the axi4 protocol and all the info has been run using **UVM_HIGH** verbosity

4.1 AXI4 Debugging Flow

Initially, open with a log file which is inside the test folder that has been run and then follow the below procedure in order to have a debug flow.

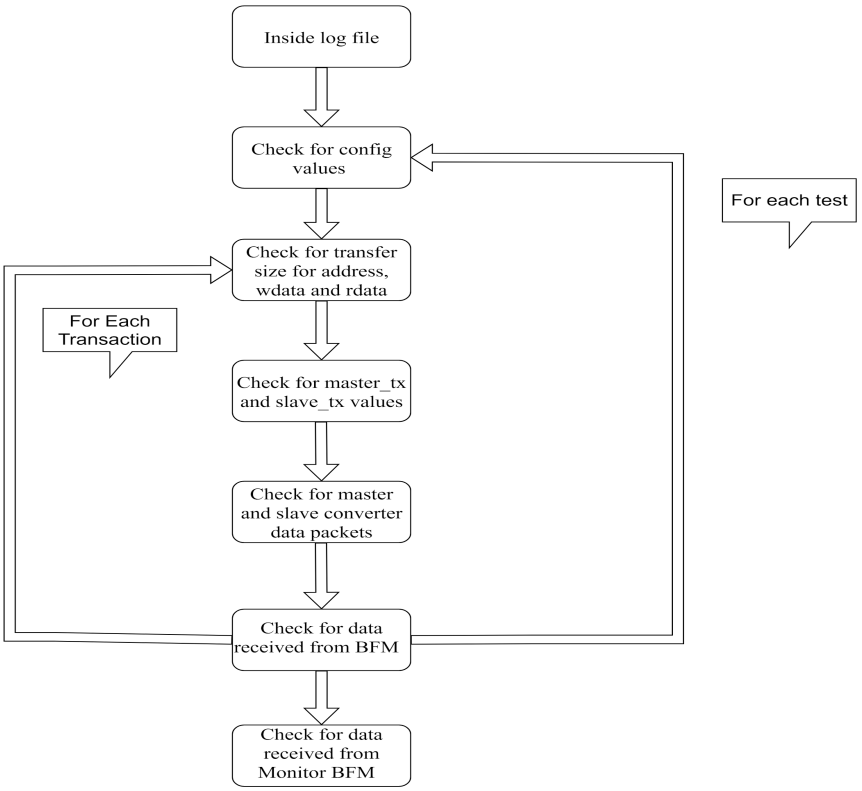


Fig 4.1 Debugging flow

4.2 Check for Configuration Values:

At this stage, the user is trying to check for all the values related to the master agent, slave agent, and environment configurations that have been generated from the test.

4.2.1(a) Master agent configurations

Master agent configurations Includes

Table 4.1 Master Configurations

Name	Type	Default value	Description
is_active	enum	UVM_ACTIVE	It will be used for configuring an agent as an active agent means it has sequencer, driver and monitor or passive agent which has monitor only.
has_coverage	integer	‘d1	Used for enabling the master agent coverage
master_min_addr_range_array[0]	integer	Associative array	Which indicates the minimum address range for master
master_max_addr_range_array[0]	integer	Associative array	Which indicates the maximum address range for master

```
UVM_INFO ../../src/hvl_top/test/axi4_base_test.sv(122) @ 0: uvm_test_top [axi4_blocking_32b_write_read_test]
AXI4_MASTER_CONFIG[0]
----------------------------------------------------------------------------
Name                                Type                      Size  Value
----------------------------------------------------------------------------
axi4_master_agent_cfg_h[0]          axi4_master_agent_config  -     @497
  is_active                         string                    10    UVM_ACTIVE
  has_coverage                      integral                  1     1
  master_min_addr_range_array[0]    integral                  32    'h0
  master_max_addr_range_array[0]    integral                  32    'hfff
  wait_count_write_address_channel  integral                  32    'd0
  wait_count_write_data_channel     integral                  32    'd0
  wait_count_read_address_channel   integral                  32    'd0
  outstanding_write_tx              integral                  32    'd0
  outstanding_read_tx               integral                  32    'd0
----------------------------------------------------------------------------
```

Fig 4.2 master_agent_config values

Figure 4.2 shows the different config values that has been set in master agent config class

4.2.1(b) Slave agent configurations

Slave agent configurations Includes

Table 4.2 Slave configurations

Configurations	conditions for the configurations
is_active	It will be used for configuring an agent as an active agent means it has sequencer, driver and monitor or passive agent which has monitor only.
has_coverage	Which indicates the coverage connection
master_min_addr_range_array[0]	Which indicates the minimum address range for master
master_max_addr_range_array[0]	Which indicate the maximum address range for master
slave_id	Shows how many slaves are connected

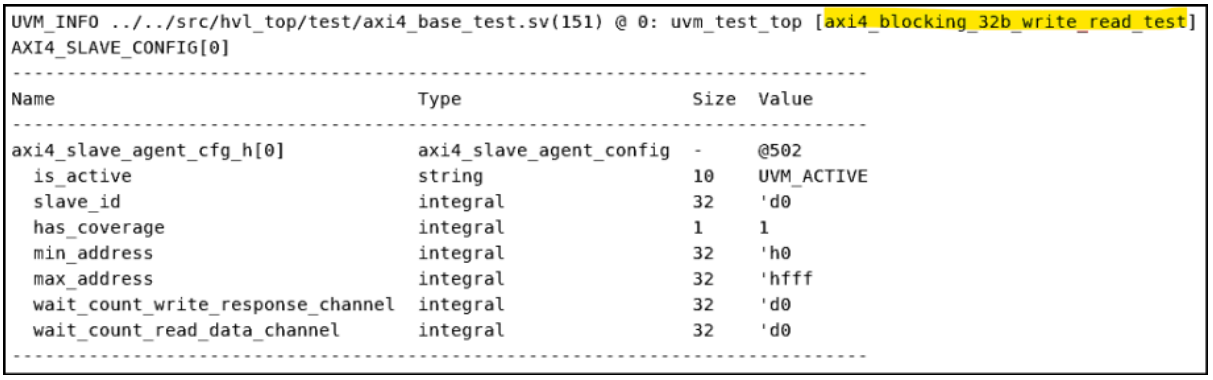


Fig 4.3 slave_agent_config values

Figure 4.3 shows the different config values that has been set in slave agent config class

4.2.1(c) Environment configuration

Environment configuration includes

Table 4.3 environment configurations

Configurations	conditions for the configurations
has_scoreboard	which tells how many scoreboards are connected to env. Which has to be at least 1
has_virtual_seqr	which tells how many virtual seqr are connected to env Which has to be at least 1
No_of_slaves	Tells how many slaves are connected Which shouldn't be 0
No_of_slaves	Tells how many masters are connected Which shouldn't be 0

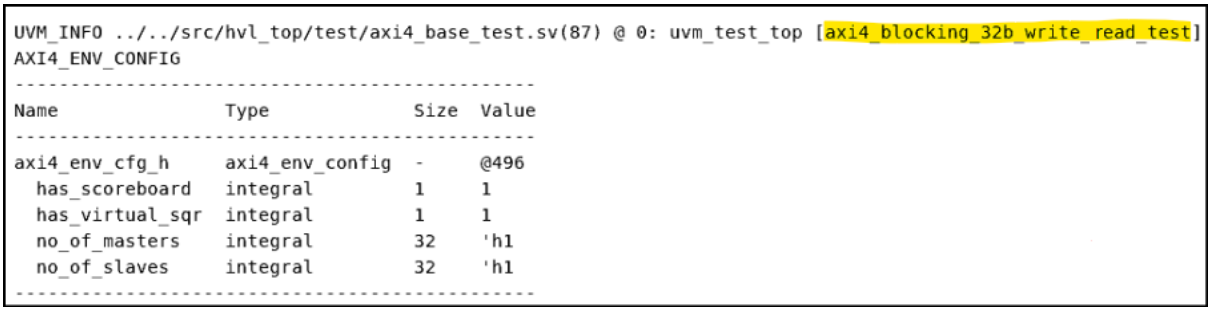


Fig 4.4 env_config values

Figure 4.4 shows the different config values that have been set in env config class

4.3 Check for transaction values

Once the config values are correct then check for the address and data to be transmitted from master_tx class as well from slave_tx class
In master_tx_class check for the transaction type, there are write type and read type and also check size, length, id address and the data

```
# Name                                  Type             Size  Value
# ------------------------------------------------------------------------------
# axi4_master_tx                        axi4_master_tx   -     @2799
#   tx_type                             string           5     WRITE
#   awid                                string           6     AWID_0
#   awaddr                              integral         32    'h49ebbc96
#   awlen                               integral         8     'd11
#   awsize                              string           12    WRITE_1_BYTE
#   awburst                             string           11    WRITE_FIXED
#   awlock                              string           19    WRITE_NORMAL_ACCESS
#   awcache                             string           16    WRITE_MODIFIABLE
#   awprot                              string           24    WRITE_NORMAL_SECURE_DATA
#   awqos                               integral         4     'h5
#   wait_count_write_address_channel    integral         32    'h5
#   wdata[0]                            integral         32    'h87d22cb6
#   wdata[1]                            integral         32    'hdd0a8202
#   wdata[2]                            integral         32    'h3253c5c9
#   wdata[3]                            integral         32    'hdc07c39a
#   wdata[4]                            integral         32    'h5a7bb08d
#   wdata[5]                            integral         32    'h818cbe22
#   wdata[6]                            integral         32    'h693491ae
#   wdata[7]                            integral         32    'h3a4fdf0d
#   wdata[8]                            integral         32    'hf4f3a456
#   wdata[9]                            integral         32    'h8199fff3
#   wdata[10]                           integral         32    'h620aa2e8
#   wdata[11]                           integral         32    'h791b91db
#   wstrb[0]                            integral         4     'h4
#   wstrb[1]                            integral         4     'h8
#   wstrb[2]                            integral         4     'h1
```

Fig 4.5 master_tx write type transaction values

```
----------------------------------------------------------------------
Name                 Type           Size  Value
----------------------------------------------------------------------
req                  axi4_slave_tx  -     @2479
  tx_type            string         4     READ
  arid               string         6     ARID_1
  araddr             integral       32    'h36a74f3f
  arlen              integral       8     'd41
  arsize             string         12    READ_4_BYTES
  arburst            string         9     READ_INCR
  arlock             string         18    READ_NORMAL_ACCESS
  arcache            string         15    READ_BUFFERABLE
  arprot             string         23    READ_NORMAL_SECURE_DATA
  arqos              integral       1     'h0
  rid                string         6     RID_12
  rdata[0]           integral       32    'hde227c86
  rdata[1]           integral       32    'h257d9d56
  rdata[2]           integral       32    'h12b0e22c
  rdata[3]           integral       32    'h30b69d1f
  rdata[4]           integral       32    'h8500a373
  rdata[5]           integral       32    'h98fb89a6
  rdata[6]           integral       32    'h950a8044
  rdata[7]           integral       32    'h60af72e
  rdata[8]           integral       32    'heb25094f
  rdata[9]           integral       32    'h1cbdd5ce
  rdata[10]          integral       32    'h8ba6b1c1
  rdata[11]          integral       32    'h141c27cd
```

No. of read transfer = length+1 = 41+1 = 42

Fig 4.6 slave_tx read type transaction values

4.4 Check for data received from driver BFM

Once the data has been randomized and sent to master or slave BFMs. The master driver BFM will drive the write transaction and the slave samples the data depending on the configurations of the master and similarly slave driver BFM will drive the data and response and sample the write data depending on the configurations of a slave.

The master driver BFM will print the write task data which has been driven by the master and write response sampled data and similarly, slave driver BFM will print which has been driven by the slave and sampled data. In the end both the master BFM and slave BFM data have to be the same.

```
# UVM INFO ../../src/hvl_top/master//axi4_master_driver_proxy.sv(198) @ 390: uvm_test_top.axi4_env_h.
 axi4_master_agent_h[0].axi4_master_drv_proxy_h [axi4_master_driver_proxy] WRITE_TASK::Response Received_r
eq_write_packet =
#  ------------------------------------------------------------------------------------------
# Name                                Type            Size  Value
# ------------------------------------------------------------------------------------------
# axi4_master_tx                      axi4_master_tx  -     @2734
#   tx_type                           string          5     WRITE
#   awid                              string          7     AWID_13
#   awaddr                            integral        32    'hd106f6a8
#   awlen                             integral        8     'd3
#   awsize                            string          13    WRITE_4_BYTES
#   awburst                           string          11    WRITE_FIXED
#   awlock                            string          19    WRITE_NORMAL_ACCESS
#   awcache                           string          16    WRITE_BUFFERABLE
#   awprot                            string          28    WRITE_PRIVILEGED_SECURE_DATA
#   awqos                             integral        4     'hb
#   wait_count_write_address_channel  integral        32    'h3
#   wdata[0]                          integral        32    'hb1219df8
#   wdata[1]                          integral        32    'hb98b8576
#   wdata[2]                          integral        32    'hc96d52b6
#   wdata[3]                          integral        32    'he73f9ab2
#   wstrb[0]                          integral        4     'hf
#   wstrb[1]                          integral        4     'hf
#   wstrb[2]                          integral        4     'hf
#   wstrb[3]                          integral        4     'hf
#   wait_count_write_data_channel     integral        32    'h0
#   bid                               string          6     BID_13
#   bresp                             string          10    WRITE_OKAY
#   no_of_wait_states                 integral        32    'd0
#   wait_count_write_response_channel integral        32    'h0
#   transfer_type                     string          14    BLOCKING_WRITE
# ------------------------------------------------------------------------------------------
```

Fig 4.7 master bfm driven and sampled write data

```
# UVM_INFO ../../src/hvl_top/master//axi4_master_driver_proxy.sv(419) @ 330: uvm_test_top.axi4_env_h.
axi4_master_agent_h[0].axi4_master_drv_proxy_h [axi4_master_driver_proxy] READ_TASK::Response_received_re
q_read_packet =
#  ------------------------------------------------------------------------------------------------
# Name                                Type            Size  Value
# ------------------------------------------------------------------------------------------------
# axi4_master_tx                      axi4_master_tx  -     @2705
#   tx_type                           string          4     READ
#   arid                              string          7     ARID_10
#   araddr                            integral        32    'hcfaba61b
#   arlen                             integral        8     'd7
#   arsize                            string          12    READ_4_BYTES
#   arburst                           string          9     READ_INCR
#   arlock                            string          18    READ_NORMAL_ACCESS
#   arcache                           string          13    READ_ALLOCATE
#   arprot                            string          37    READ_PRIVILEGED_NONSECURE_INSTRUCTION
#   arqos                             integral        1     'h1
#   wait_count_read_address_channel   integral        32    'h2
#   rid                               string          6     RID_10
#   rdata[0]                          integral        32    'h6f7a4f86
#   rdata[1]                          integral        32    'h6920ce56
#   rdata[2]                          integral        32    'h8ad4c12c
#   rdata[3]                          integral        32    'h6afdcd1f
#   rdata[4]                          integral        32    'h88ee3373
#   rdata[5]                          integral        32    'h4221b2a6
#   rdata[6]                          integral        32    'hc4647144
#   rdata[7]                          integral        32    'hf7e5ed2e
#   rresp                             string          9     READ_OKAY
#   ruser                             integral        1     'h0
#   no_of_wait_states                 integral        32    'd0
#   wait_count_read_data_channel      integral        32    'h0
#   transfer_type                     string          14    BLOCKING_WRITE
```

Fig 4.8 master bfm driven and sampled read data

```
UVM_INFO ../../src/hvl_top/slave/axi4_slave_driver_proxy.sv(271) @ 390: uvm_test_top.axi4_env_h.axi4_slave_agent_h[0].
axi4_slave_drv_proxy_h [DEBUG_SLAVE_WDATA_PROXY] AFTER :: COMBINED WRITE CHANNEL PACKET
  ------------------------------------------------------------
Name           Type           Size  Value
------------------------------------------------------------
axi4_slave_tx  axi4_slave_tx  -     @2730
  tx_type      string         5     WRITE
  awid         string         7     AWID_13
  awaddr       integral       32    'hd106f6a8
  awlen        integral       8     'd3
  awsize       string         13    WRITE_4_BYTES
  awburst      string         11    WRITE_FIXED
  awlock       string         19    WRITE_NORMAL_ACCESS
  awcache      string         16    WRITE_BUFFERABLE
  awprot       string         24    WRITE_NORMAL_SECURE_DATA
  awqos        integral       1     'h0
  wdata[0]     integral       32    'hb1219df8
  wdata[1]     integral       32    'hb98b8576
  wdata[2]     integral       32    'hc96d52b6
  wdata[3]     integral       32    'he73f9ab2
  wstrb[0]     integral       4     'hf
  wstrb[1]     integral       4     'hf
  wstrb[2]     integral       4     'hf
  wstrb[3]     integral       4     'hf
  wlast        integral       1     1
  wuser        integral       4     'd0
  bid          string         6     BID_13
  bresp        string         10    WRITE_OKAY
  buser        integral       4     'd0
------------------------------------------------------------
```

Fig 4.9 slave bfm driven and sampled write data

```
UVM_INFO ../../src/hvl_top/slave/axi4_slave_driver_proxy.sv(381) @ 330: uvm_test_top.axi4_env_h.axi4_slave_agent_h[0].
axi4_slave_drv_proxy_h [DEBUG_SLAVE_RDATA_PROXY] AFTER :: COMBINED READ CHANNEL PACKET
  ---------------------------------------------------------------
Name                  Type           Size  Value
---------------------------------------------------------------
axi4_slave_tx         axi4_slave_tx  -     @2701
  tx_type             string         4     READ
  arid                string         7     ARID_10
  araddr              integral       32    'hcfaba61b
  arlen               integral       8     'd7
  arsize              string         12    READ_4_BYTES
  arburst             string         9     READ_INCR
  arlock              string         18    READ_NORMAL_ACCESS
  arcache             string         15    READ_BUFFERABLE
  arprot              string         23    READ_NORMAL_SECURE_DATA
  arqos               integral       1     'h0
  rid                 string         6     RID_10
  rdata[0]            integral       32    'h6f7a4f86
  rdata[1]            integral       32    'h6920ce56
  rdata[2]            integral       32    'h8ad4c12c
  rdata[3]            integral       32    'h6afdcd1f
  rdata[4]            integral       32    'h88ee3373
  rdata[5]            integral       32    'h4221b2a6
  rdata[6]            integral       32    'hc4647144
  rdata[7]            integral       32    'hf7e5ed2e
  rresp               string         9     READ_OKAY
  ruser               integral       4     'h0
  no_of_wait_states   integral       32    'h0
---------------------------------------------------------------
```

Fig 4.10 slave bfm driven and sampled read data

fig 4.9 and 4.10 shows the data with respect to the write and read task of master
The fig 4.10 and 4.11 shows the values with respect to write and read task of slave

4.6 Check for data received from monitor BFM to proxy

Once the data has been driven or sampled from master and slave driver, monitor will capture the data for all the write and read channels and it will print the sampled data for each of the 5 channels.

Check the values for all the channels in Master Monitor.

A. Sampled data from monitor bfm : write address channel

```
UVM_INFO ../../src/hvl_top/master//axi4_master_monitor_proxy.sv(159) @ 110: uvm_test_top.axi4_env_h.axi4_master_agent_h[0].
axi4_master_mon_proxy_h [axi4_master_monitor_proxy] Packet received from axi4_write_address clone packet is

  ---------------------------------------------------------------------------------------
 Name                                  Type            Size  Value
 ---------------------------------------------------------------------------------------
 axi4_master_tx                        axi4_master_tx  -     @2649
   tx_type                             string          5     WRITE
   awid                                string          7     AWID_13
   awaddr                              integral        32    'hd106f6a8
   awlen                               integral        8     'd3
   awsize                              string          13    WRITE_4_BYTES
   awburst                             string          11    WRITE_FIXED
   awlock                              string          19    WRITE_NORMAL_ACCESS
   awcache                             string          16    WRITE_BUFFERABLE
   awprot                              string          28    WRITE_PRIVILEGED_SECURE_DATA
   awqos                               integral        4     'h0
   wait_count_write_address_channel    integral        32    'h0
   wait_count_write_data_channel       integral        32    'h0
   bid                                 string          5     BID_0
   bresp                               string          10    WRITE_OKAY
   no_of_wait_states                   integral        32    'd0
   wait_count_write_response_channel   integral        32    'h0
   transfer_type                       string          14    BLOCKING_WRITE
 ---------------------------------------------------------------------------------------
```

Fig 4.11 write address channel

B. Sampled data from monitor bfm : write data channel

```
UVM_INFO ../../src/hvl_top/master//axi4_master_monitor_proxy.sv(191) @ 250: uvm_test_top.axi4_env_h.axi4_master_agent_h[0].
axi4_master_mon_proxy_h [axi4_master_monitor_proxy] Packet received from axi4_write_data clone packet is

  ---------------------------------------------------------------------------------------
 Name                                  Type            Size  Value
 ---------------------------------------------------------------------------------------
 axi4_master_tx                        axi4_master_tx  -     @2669
   tx_type                             string          5     WRITE
   awid                                string          7     AWID_13
   awaddr                              integral        32    'hd106f6a8
   awlen                               integral        8     'd3
   awsize                              string          13    WRITE_4_BYTES
   awburst                             string          11    WRITE_FIXED
   awlock                              string          19    WRITE_NORMAL_ACCESS
   awcache                             string          16    WRITE_BUFFERABLE
   awprot                              string          28    WRITE_PRIVILEGED_SECURE_DATA
   awqos                               integral        4     'h0
   wait_count_write_address_channel    integral        32    'h0
   wdata[0]                            integral        32    'hb1219df8
   wdata[1]                            integral        32    'hb98b8576
   wdata[2]                            integral        32    'hc96d52b6
   wdata[3]                            integral        32    'he73f9ab2
   wstrb[0]                            integral        4     'hf
   wstrb[1]                            integral        4     'hf
   wstrb[2]                            integral        4     'hf
   wstrb[3]                            integral        4     'hf
   wait_count_write_data_channel       integral        32    'h0
   bid                                 string          5     BID_0
   bresp                               string          10    WRITE_OKAY
   no_of_wait_states                   integral        32    'd0
   wait_count_write_response_channel   integral        32    'h0
   transfer_type                       string          14    BLOCKING_WRITE
 ---------------------------------------------------------------------------------------
```

Fig 4.12 write data channel

C. Sampled data from monitor bfm : write response channel

```
UVM_INFO ../../src/hvl_top/master//axi4_master_monitor_proxy.sv(220) @ 390: uvm_test_top.axi4_env_h.axi4_master_agent_h[0].
axi4_master_mon_proxy_h [axi4_master_monitor_proxy] Packet received from axi4_write_response clone packet is
 ----------------------------------------------------------------------------------------
Name                                  Type            Size  Value
----------------------------------------------------------------------------------------
axi4_master_tx                        axi4_master_tx  -     @2746
  tx_type                             string          5     WRITE
  awid                                string          7     AWID_13
  awaddr                              integral        32    'hd106f6a8
  awlen                               integral        8     'd3
  awsize                              string          13    WRITE_4_BYTES
  awburst                             string          11    WRITE_FIXED
  awlock                              string          19    WRITE_NORMAL_ACCESS
  awcache                             string          16    WRITE_BUFFERABLE
  awprot                              string          28    WRITE_PRIVILEGED_SECURE_DATA
  awqos                               integral        4     'h0
  wait_count_write_address_channel    integral        32    'h0
  wdata[0]                            integral        32    'hb1219df8
  wdata[1]                            integral        32    'hb98b8576
  wdata[2]                            integral        32    'hc96d52b6
  wdata[3]                            integral        32    'he73f9ab2
  wstrb[0]                            integral        4     'hf
  wstrb[1]                            integral        4     'hf
  wstrb[2]                            integral        4     'hf
  wstrb[3]                            integral        4     'hf
  wait_count_write_data_channel       integral        32    'h0
  bid                                 string          6     BID_13
  bresp                               string          10    WRITE_OKAY
  no_of_wait_states                   integral        32    'd0
  wait_count_write_response_channel   integral        32    'h0
  transfer_type                       string          14    BLOCKING_WRITE
----------------------------------------------------------------------------------------
```

Fig 4.13 write response channel

D. Sampled data from monitor bfm : read address channel

```
UVM_INFO ../../src/hvl_top/master//axi4_master_monitor_proxy.sv(244) @ 90: uvm_test_top.axi4_env_h.axi4_master_agent_h[0].
axi4_master_mon_proxy_h [axi4_master_monitor_proxy] Packet received from axi4_read_address clone packet is
 ----------------------------------------------------------------------------------------
Name                                  Type            Size  Value
----------------------------------------------------------------------------------------
axi4_master_tx                        axi4_master_tx  -     @2623
  tx_type                             string          4     READ
  arid                                string          7     ARID_10
  araddr                              integral        32    'hcfaba61b
  arlen                               integral        8     'd7
  arsize                              string          12    READ_4_BYTES
  arburst                             string          9     READ_INCR
  arlock                              string          18    READ_NORMAL_ACCESS
  arcache                             string          13    READ_ALLOCATE
  arprot                              string          37    READ_PRIVILEGED_NONSECURE_INSTRUCTION
  arqos                               integral        1     'h1
  wait_count_read_address_channel     integral        32    'h0
  rid                                 string          5     RID_0
  rresp                               string          9     READ_OKAY
  ruser                               integral        1     'h0
  no_of_wait_states                   integral        32    'd0
  wait_count_read_data_channel        integral        32    'h0
  transfer_type                       string          14    BLOCKING_WRITE
----------------------------------------------------------------------------------------
```

Fig 4.14 read address channel

E. Sampled data from monitor bfm : read data channel

```
UVM_INFO ../../src/hvl_top/master//axi4_master_monitor_proxy.sv(270) @ 310: uvm_test_top.axi4_env_h.axi4_master_agent_h[0].
axi4_master_mon_proxy_h [axi4_master_monitor_proxy] Packet received from axi4_read_data clone packet is
 ----------------------------------------------------------------------------------------------
Name                                Type            Size  Value
----------------------------------------------------------------------------------------------
axi4_master_tx                      axi4_master_tx  -     @2693
  tx_type                           string          4     READ
  arid                              string          7     ARID_10
  araddr                            integral        32    'hcfaba61b
  arlen                             integral        8     'd7
  arsize                            string          12    READ_4_BYTES
  arburst                           string          9     READ_INCR
  arlock                            string          18    READ_NORMAL_ACCESS
  arcache                           string          13    READ_ALLOCATE
  arprot                            string          37    READ_PRIVILEGED_NONSECURE_INSTRUCTION
  arqos                             integral        1     'h1
  wait_count_read_address_channel   integral        32    'h0
  rid                               string          6     RID_10
  rdata[0]                          integral        32    'h6f7a4f86
  rdata[1]                          integral        32    'h6920ce56
  rdata[2]                          integral        32    'h8ad4c12c
  rdata[3]                          integral        32    'h6afdcd1f
  rdata[4]                          integral        32    'h88ee3373
  rdata[5]                          integral        32    'h4221b2a6
  rdata[6]                          integral        32    'hc4647144
  rdata[7]                          integral        32    'hf7e5ed2e
  rresp                             string          9     READ_OKAY
  ruser                             integral        1     'h0
  no_of_wait_states                 integral        32    'd0
  wait_count_read_data_channel      integral        32    'h0
  transfer_type                     string          14    BLOCKING_WRITE
```

Fig 4.15 read data channel

Similarly the slave monitor will have 5 tasks and for each task needs to check the values.

4.7 Scoreboard Checks:

And finally we have scoreboard checks which basically compare the rid from master and the slave, raddr of the master and the slave, rlen of the master and the slave, rsize of the master and the slave, and another transfer signal from the master and the slave as shown below:-

```
# UVM_INFO ../../src/hvl_top/env/axi4_scoreboard.sv(598) @ 90: uvm_test_top.axi4_env_h.axi4_scoreboard_h [axi4_scoreboard] axi4_a
rid from master and slave is equal
# UVM_INFO ../../src/hvl_top/env/axi4_scoreboard.sv(599) @ 90: uvm_test_top.axi4_env_h.axi4_scoreboard_h [SB_arID_MATCHED] Master
 arID = 'h0 and Slave arID = 'h0
# UVM_INFO ../../src/hvl_top/env/axi4_scoreboard.sv(608) @ 90: uvm_test_top.axi4_env_h.axi4_scoreboard_h [axi4_scoreboard] axi4_a
raddr from master and slave is equal
# UVM_INFO ../../src/hvl_top/env/axi4_scoreboard.sv(609) @ 90: uvm_test_top.axi4_env_h.axi4_scoreboard_h [SB_arADDR_MATCHED] Mast
er arADDR = 'h2def660b and Slave arADDR = 'h2def660b
# UVM_INFO ../../src/hvl_top/env/axi4_scoreboard.sv(618) @ 90: uvm_test_top.axi4_env_h.axi4_scoreboard_h [axi4_scoreboard] axi4_a
rlen from master and slave is equal
# UVM_INFO ../../src/hvl_top/env/axi4_scoreboard.sv(619) @ 90: uvm_test_top.axi4_env_h.axi4_scoreboard_h [SB_arlen_MATCHED] Maste
r arlen = 'hc and Slave arlen = 'hc
# UVM_INFO ../../src/hvl_top/env/axi4_scoreboard.sv(628) @ 90: uvm_test_top.axi4_env_h.axi4_scoreboard_h [axi4_scoreboard] axi4_a
rsize from master and slave is equal
# UVM_INFO ../../src/hvl_top/env/axi4_scoreboard.sv(629) @ 90: uvm_test_top.axi4_env_h.axi4_scoreboard_h [SB_arsize_MATCHED] Mast
er arsize = 'h2 and Slave arsize = 'h2
# UVM_INFO ../../src/hvl_top/env/axi4_scoreboard.sv(638) @ 90: uvm_test_top.axi4_env_h.axi4_scoreboard_h [axi4_scoreboard] axi4_a
rburst from master and slave is equal
# UVM_INFO ../../src/hvl_top/env/axi4_scoreboard.sv(639) @ 90: uvm_test_top.axi4_env_h.axi4_scoreboard_h [SB_arburst_MATCHED] Mas
ter arburst = 'h0 and Slave arburst = 'h0
```

Fig 4.16 scoreboard_checks

4.8 Coverage Debug:

Coverage is a metric which basically tells how much percentage of verification has been done to the dut.

Go to the log file, Here it will get you the complete master and slave coverage for the particular test we are running.

```
UVM_INFO ../../src/hvl_top/master//axi4_master_coverage.sv(235) @ 4110: uvm_test_top.axi4_env_h.axi4_master_agent_h[0].axi4_mas
er_cov_h [axi4_master_coverage] AXI4 Master Agent Coverage = 23.88 %
```

Fig 4.17 coverage for master

```
UVM_INFO ../../src/hvl_top/slave/axi4_slave_coverage.sv(235) @ 4110: uvm_test_top.axi4_env_h.axi4_slave_agent_h[0].axi4_slave_c
_h [axi4_slave_coverage] AXI4 Slave Agent Coverage = 23.88 %
```

Fig 4.18 coverage for slave

For individual bins checking goto the below html file :-
firefox axi4_blocking_write_read_test/html_cov_report/index.html &

Inside that check for covergroups in the coverage summary then check for the instance created for master and slave coverage

Covergroups Coverage Summary:

Search:

Covergroups/Instances	Total Bins	Hits	Misses	Hits %	Goal %	Coverage %
/axi4_slave_pkg/axi4_slave_coverage/axi4_slave_covergroup	1136	43	1093	3.78%	23.87%	23.87%
/axi4_master_pkg/axi4_master_coverage/axi4_master_covergroup	1136	43	1093	3.78%	23.87%	23.87%
work.axi4_slave_pkg::axi4_slave_coverage/axi4_slave_covergroup	1136	43	1093	3.78%	23.87%	23.87%
work.axi4_master_pkg::axi4_master_coverage/axi4_master_covergroup	1136	43	1093	3.78%	23.87%	23.87%

Fig 4.19 master and slave coverage

Questa Covergroup Coverage Report

Search: cvg:axi4_master_covergroup

Covergroups/Instances	Total Bins	Hits	Misses	Hits %	Goal %	Coverage %
Covergroup axi4_master_covergroup	1136	43	1093	3.78%	23.87%	23.87%
Instance \/axi4_master_pkg::axi4_master...erage::axi4_master_covergroup	1136	43	1093	3.78%	23.87%	23.87%

Fig 4.20 instance of cover group

Then click on the master covergroup instance to check the individual bins which are hitted and missed. And here you can even check cross coverages between write burst, write length, write size, read burst, read length,and read size.

CoverPoints	Total Bins	Hits	Misses	Hit %	Goal %	Coverage %
ARBURST_CP	3	1	2	33.33%	33.33%	33.33%
ARCACHE_CP	4	2	2	50.00%	50.00%	50.00%
ARID_CP	16	3	13	18.75%	18.75%	18.75%
ARLEN_CP	9	1	8	11.11%	11.11%	11.11%
ARLOCK_CP	2	1	1	50.00%	50.00%	50.00%
ARPROT_CP	8	4	4	50.00%	50.00%	50.00%
ARSIZE_CP	8	2	6	25.00%	25.00%	25.00%
AWBURST_CP	3	1	2	33.33%	33.33%	33.33%
AWCACHE_CP	4	3	1	75.00%	75.00%	75.00%
AWID_CP	16	2	14	12.50%	12.50%	12.50%
AWLEN_CP	9	1	8	11.11%	11.11%	11.11%
AWLOCK_CP	2	1	1	50.00%	50.00%	50.00%

Fig 4.21 master_coverage coverpoint

Figure 4.21 shows all the coverpoints included in master coverage

Crosses	Total Bins	Hits	Misses	Hit %	Goal %	Coverage %
ARBURST_CP_X_ARLEN_CP_X_ARSIZE_CP	216	1	215	0.46%	0.46%	0.46%
ARLENGTH_CP_X_ARSIZE_X_ARBURST	216	1	215	0.46%	0.46%	0.46%
AWBURST_CP_X_AWLEN_CP_X_AWSIZE_CP	216	1	215	0.46%	0.46%	0.46%
AWLENGTH_CP_X_AWSIZE_X_AWBURST	216	1	215	0.46%	0.46%	0.46%
BID_CP_X_BRESP_CP	64	2	62	3.12%	3.12%	3.12%
RID_CP_X_RRESP_CP	64	2	62	3.12%	3.12%	3.12%

Fig 4.22 master_coverage crosses coverpoints

Figure 4.22 shows all the cross coverpoints included in master coverage

If you click on the slave covergroup instance to check the individual bins which are hitted and missed. And here you can even check cross coverages between write burst, write length, write size, read burst, read length,and read size.

CoverPoints	Total Bins	Hits	Misses	Hit %	Goal %	Coverage %
ARBURST_CP	3	1	2	33.33%	33.33%	33.33%
ARCACHE_CP	4	2	2	50.00%	50.00%	50.00%
ARID_CP	16	3	13	18.75%	18.75%	18.75%
ARLEN_CP	9	1	8	11.11%	11.11%	11.11%
ARLOCK_CP	2	1	1	50.00%	50.00%	50.00%
ARPROT_CP	8	4	4	50.00%	50.00%	50.00%
ARSIZE_CP	8	2	6	25.00%	25.00%	25.00%
AWBURST_CP	3	1	2	33.33%	33.33%	33.33%
AWCACHE_CP	4	3	1	75.00%	75.00%	75.00%
AWID_CP	16	2	14	12.50%	12.50%	12.50%
AWLEN_CP	9	1	8	11.11%	11.11%	11.11%
AWLOCK_CP	2	1	1	50.00%	50.00%	50.00%

Fig 4.23 salve_coverage coverpoint

Figure 4.23 shows all the cover points included in slave coverage

Crosses	Total Bins	Hits	Misses	Hit %	Goal %	Coverage %
ARBURST_CP_X_ARLEN_CP_X_ARSIZE_CP	216	1	215	0.46%	0.46%	0.46%
ARLENGTH_CP_X_ARSIZE_X_ARBURST	216	1	215	0.46%	0.46%	0.46%
AWBURST_CP_X_AWLEN_CP_X_AWSIZE_CP	216	1	215	0.46%	0.46%	0.46%
AWLENGTH_CP_X_AWSIZE_X_AWBURST	216	1	215	0.46%	0.46%	0.46%
BID_CP_X_BRESP_CP	64	2	62	3.12%	3.12%	3.12%
RID_CP_X_RRESP_CP	64	2	62	3.12%	3.12%	3.12%

Fig 4.24 slave_coverage crosses cover points

Figure 4.24 shows all the cross cover points included in slave coverage

4.9 Waveform Viewer

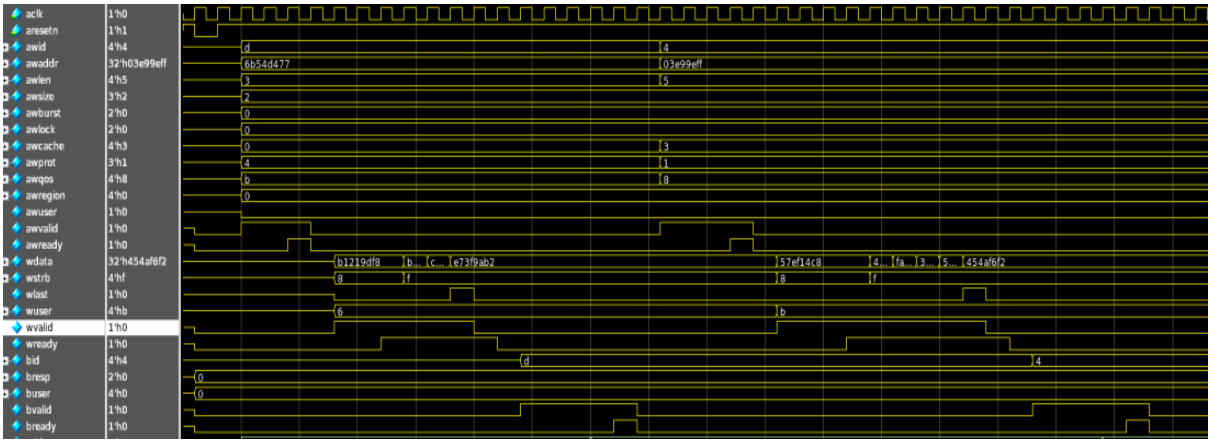


Fig 4.25 waveform for 32 blocking write transfers

1. In the waveform, initially check for the generation of the system clock(aclk), after every 10ns it will be toggled as shown in figure 4.26. Once the aclk is done check for the reset condition(Active low reset) if the reset is low all the signals in different channels should be in unknown state.
2. Once the reset is high at the next posedge of aclk master can initiates the request to the slave by asserting all the write and read address channel signals

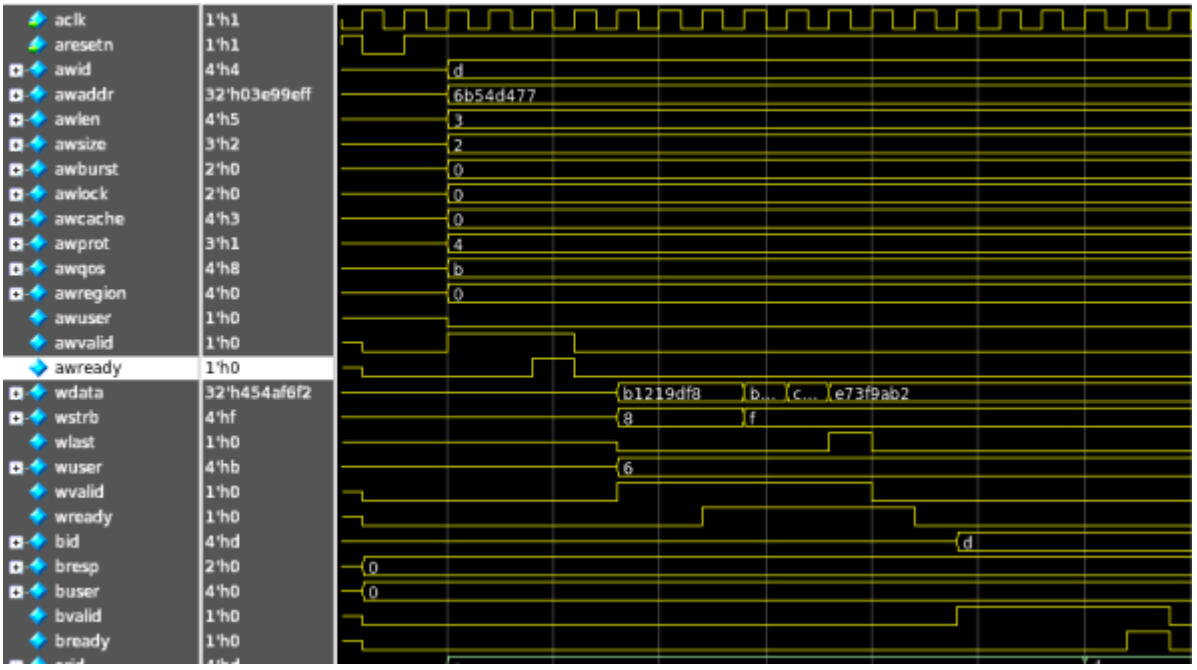


Fig 4.26 wave form for 32 blocking write transfers with initial req

3. The fig 4.27 shows the blocking write transfers since it is of blocking type the write data channel and response channel have to wait to complete the write address phase by master.
4. Once the master completes the write address phase, the master starts sending the write data followed by a response.
5. Once the master completes all 3 channels for req1 then it can start 2nd transfer for reads as well as shown in fig 4.28.

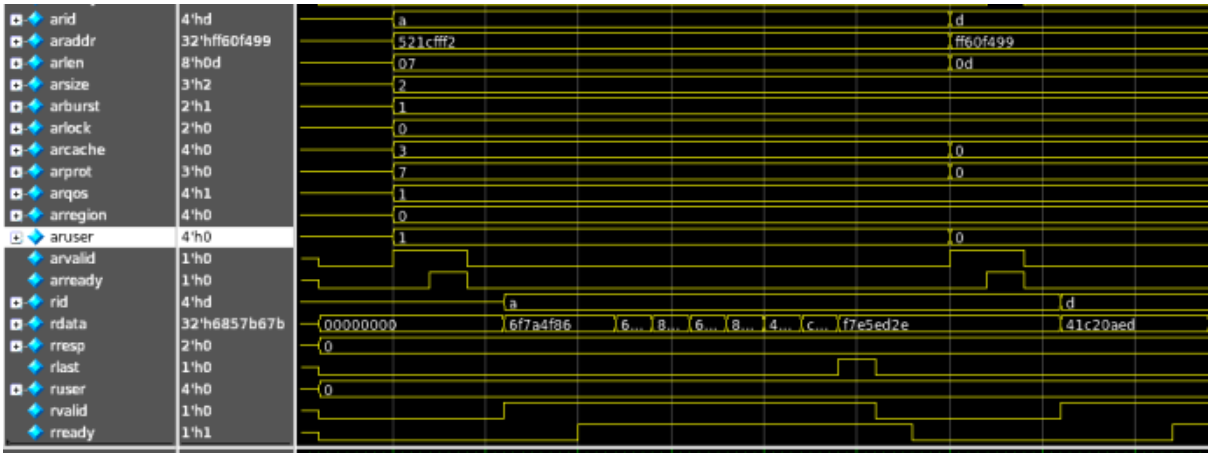


Fig 4.27 wave form for 32 blocking Read transfers with initial req

6. In the write data channel, check for the size of the transfer and length of the burst and wlast will be asserted to be high at the end of the transfer.
7. Once write data is done check for the response for each transaction

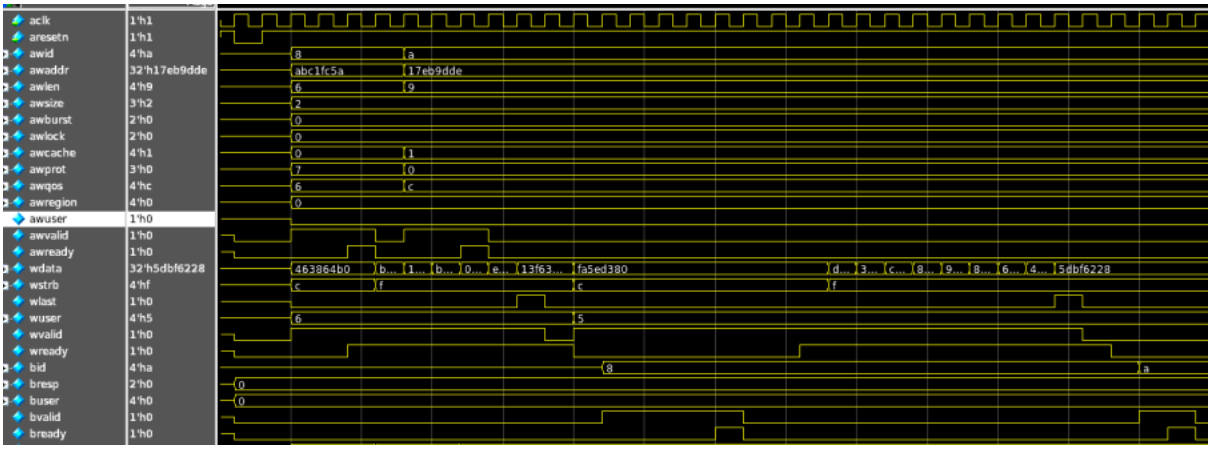


Fig 4.28 waveform for 32 nonblocking write transfers with outstanding transfers

1. The fig 4.29 shows the non-blocking write transfers since it is of non-blocking type all three 3 channels will start at a time and run parallely.
2. Due to that master can initiate multiple outstanding transfers without waiting for the written data and response to complete

3. Since all are independent channels each channel has there own handshaking valid ready mechanism. In order to ensure that data loss or incorrect data will not be transferred.

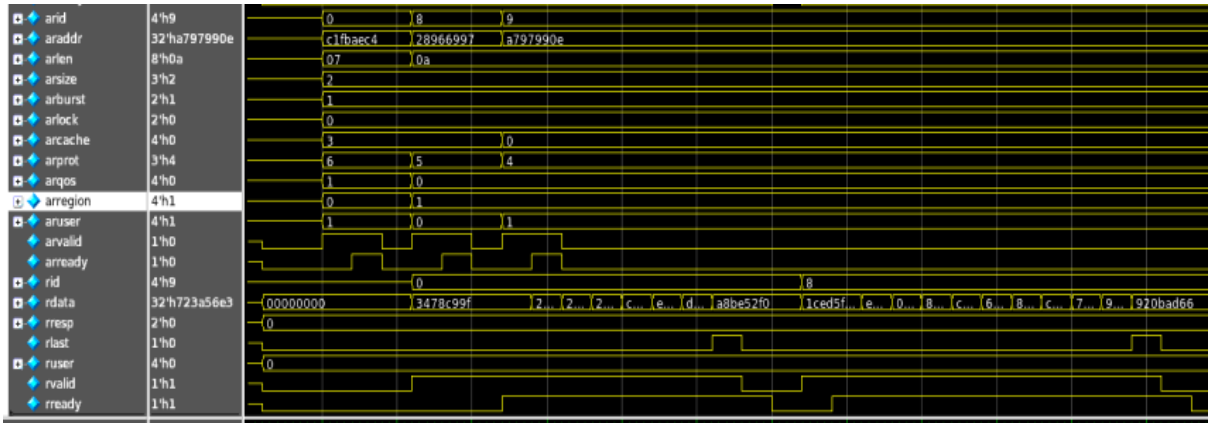


Fig 4.29 wave form for 32 non blocking read transfers with outstanding transfers

4. Based on the read address channel requested by master slave will send the read data using read data channel and response based on attributes on read address channel signals as shown in fig 4.30

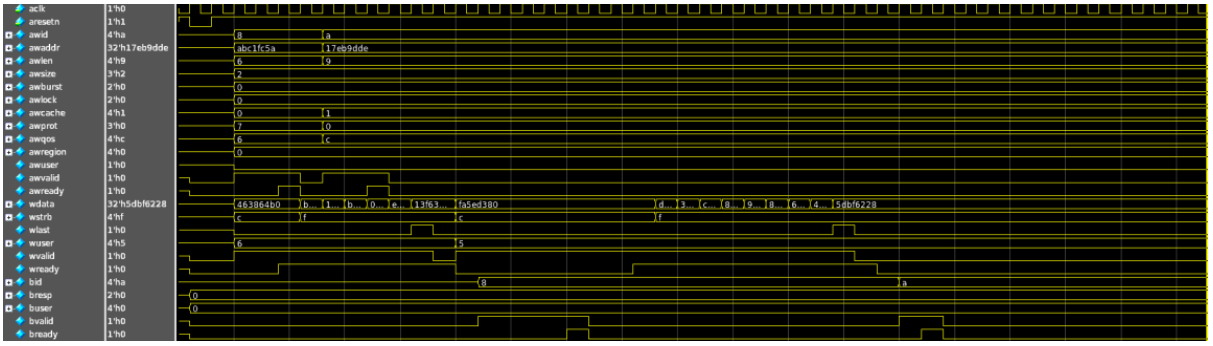


Fig 4.30 complete wave form for 32 non blocking write transfers with outstanding transfers

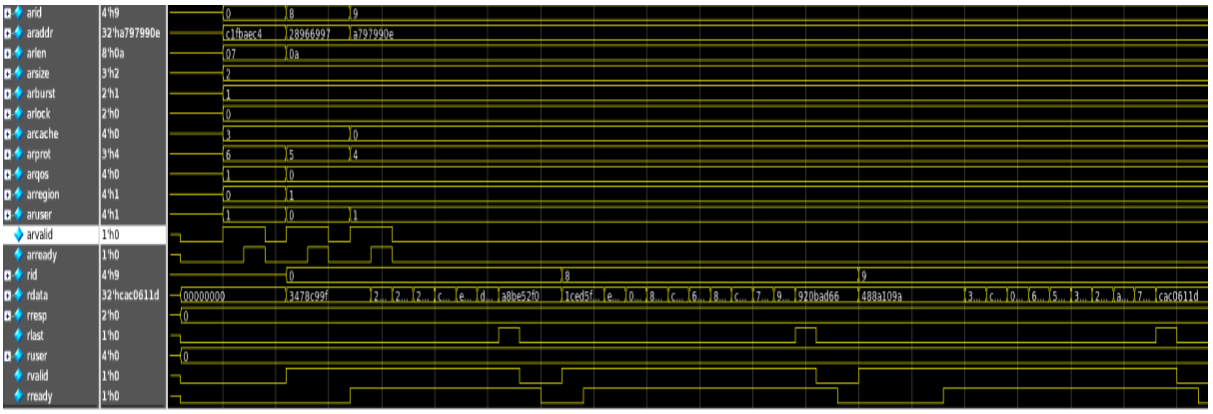


Fig 4.31 complete wave form for 32 non blocking read transfers with outstanding transfers

Chapter 5

References

Git Hub guidlines

amba_axi_and_ace_protocol_spec