Verification Continuum™

ZeBu® AMBA AXI Monitor User Guide

Version Q-2021.06, June 2021

SYNOPSYS®

Contents

List of Tables

List of Figures

About This Manual

Overview

This manual describes how to use the ZeBu AMBA AMBA AXI Monitor with your design being emulated in ZeBu.

Related Documentation

For details about the ZeBu supported features and limitations, you should refer to the ZeBu Release Notes in the ZeBu documentation package which corresponds to the software version you are using.

You can find relevant information for usage of the present transactor in the training material about Using Transactors.

Synopsys Statement on Inclusivity and Diversity

Synopsys is committed to creating an inclusive environment where every employee, customer, and partner feels welcomed. We are reviewing and removing exclusionary language from our products and supporting customer-facing collateral. Our effort also includes internal initiatives to remove biased language from our engineering and working environment, including terms that are embedded in our software and IPs. At the same time, we are working to ensure that our web content and software applications are usable to people of varying abilities. You may still find examples of non-inclusive language in our software or documentation as our IPs implement industry-standard specifications that are currently under review to remove exclusionary language.

1 Introduction

AMBA AMBA AXI Monitor allows to capture traffic on ARM Advanced Micro-controller Bus Architecture (AMBA) AXI3/AXI4 hardware interface & stores the information in log file or Verdi specific FSDB file . This log file can be post-processed to extract performance statistics like bandwidth & latency or converted into a formatted transaction trace log using provided scripts. There is also support for Verdi Protocol Analyzer which shows bus traffic in graphical interface along with relationships for AXI transaction.

This section explains the following topics:

- *Overview*
- *Supported Features*

1.1 Overview

AMBA AMBA AXI Monitor provides software APIs to configure the monitor & enable/ disable the monitor at runtime.

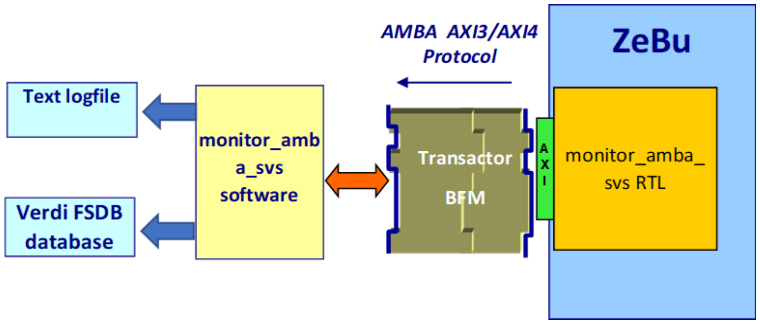


FIGURE 1. AMBA AMBA AXI Monitor Configuration

1.2 Supported Features

The following features are supported by AMBA AMBA AXI Monitor:

¢ Compliant with all ARM AMBA AXI data and address widths

¢ Supports all protocol transfer types, burst types, burst lengths and response types

¢ Support for burst-based transactions

¢ Supports multiple monitor instances

¢ Enable/disable monitor dumping at runtime

¢ Support for Verdi Protocol Analyzer (PA) for traffic monitoring

¢ Easy integration with other ZeBu transactors

1.3 FLEXlm License Features

You need the hw_xtormm_monitors FLEXlm license feature.

2 Installation

This section explains the following topics:

¢ *Installing the ZeBu AMBA AMBA AXI Monitor*

¢ *Package Description*

¢ *File Tree*

2.1 Installing the ZeBu AMBA AMBA AXI Monitor

2.1.1 Procedure

To install the AXI Master/Slave transactor, proceed as follows:

1. Make sure you have WRITE permissions on the IP directory and current directory.
2. Download the transactor compressed shell archive (`.sh`).
3. Install the ACE Lite Master/Slave transactor as follows:

```
$sh monitor_amba_svs.<version>.sh install [ZEBU_IP_ROOT]
```

where `[ZEBU_IP_ROOT]` is the path to your ZeBu IP directory:

¶ If no path is specified, the `ZEBU_IP_ROOT` environment variable is used automatically.

¶ If the path is specified and a `ZEBU_IP_ROOT` environment variable is also set, the transactor is installed at the defined path and the environment variable is ignored.

The installation process is complete and successful when the following message is displayed:

```
monitor_amba_svs v.<num> has been successfully installed.
```

If an error occurred during the installation, a message is displayed to point out the error. Here is an error message example:

```
ERROR: /auto/path/directory is not a valid directory.
```

During installation, the symbolic links are created in the following directories for an easy access from all ZeBu tools.

¢ $ZEBU_IP_ROOT/include

¢ $ZEBU_IP_ROOT/lib

¢ $ZEBU_IP_ROOT/vlog

2.2 Package Description

Once correctly installed, the monitor_amba_svs package comes with the following elements:

¢ .so library of the ZeBu AMBA AMBA AXI Monitor API

¢ Header files of the ZeBu AMBA AMBA AXI Monitor API

¢ Verilog files for the ZeBu AMBA AMBA AXI Monitor

¢ Post-processing scripts ($ZEBU_IP_ROOT/monitor_amba_svs/misc)

¢ Basic examples

2.3 File Tree

The following is an sample file tree of the ZeBu AMBA AMBA AXI Monitor after package installation:

```
BUILD_DIR/ZebuIpRoot_TD
├── alldone
├── monitor_amba_svs -> XTOR/monitor_amba_svs.$(RELEASE_STRING)
├── bin
│   └── Banner -> /remote/vgrnd20/tanuj/XTOR/td3/BUILD_DIR/
ZebuIpRoot_TD_2019_04_05/version/Banner
├── bin64 -> bin
├── doc
│   └── foss
│       └── ZX-XTOR-Library_20180911_FOSS.PDF -> ../../XTOR/
monitor_amba_svs.$(RELEASE_STRING)/doc/foss/ZX-XTOR-
Library_20180911_FOSS.PDF
├── drivers
├── gate
├── include
│   ├── monitor_amba_svs.hh -> ../XTOR/monitor_amba_svs.$(RELEASE_STRING)/
include/monitor_amba_svs.$(RELEASE_STRING).h
│   ├── CoRoutine.hh -> ../XTOR/monitor_amba_svs.$(RELEASE_STRING)/
include/CoRoutine.12.0.hh
│   ├── MPXtorBase.hh -> ../XTOR/monitor_amba_svs.$(RELEASE_STRING)/
include/MPXtorBase.12.0.hh
│   ├── svt_cr_threading.hh -> ../XTOR/monitor_amba_svs.$(RELEASE_STRING)/
include/svt_cr_threading.12.0.hh
│   ├── svt_c_runtime_cfg.hh -> ../XTOR/
monitor_amba_svs.$(RELEASE_STRING)/include/svt_c_runtime_cfg.12.0.hh
│   ├── svt_c_threading.hh -> ../XTOR/monitor_amba_svs.$(RELEASE_STRING)/
include/svt_c_threading.12.0.hh
│   ├── svt_hw_platform.hh -> ../XTOR/monitor_amba_svs.$(RELEASE_STRING)/
include/svt_hw_platform.12.0.hh
│   ├── svt_message_port.hh -> ../XTOR/monitor_amba_svs.$(RELEASE_STRING)/
include/svt_message_port.12.0.hh
```

```
│   ├── svt_pthread_threading.hh -> ../XTOR/
monitor_amba_svs.$(RELEASE_STRING)/include/svt_pthread_threading.12.0.hh
│   ├── svt_report.hh -> ../XTOR/monitor_amba_svs.$(RELEASE_STRING)/
include/svt_report.12.0.hh
│   ├── svt_report_uvm.hh -> ../XTOR/monitor_amba_svs.$(RELEASE_STRING)/
include/svt_report_uvm.12.0.hh
│   ├── svt_simulator_platform.hh -> ../XTOR/
monitor_amba_svs.$(RELEASE_STRING)/include/svt_simulator_platform.12.0.hh
│   ├── svt_systemc_threading.hh -> ../XTOR/
monitor_amba_svs.$(RELEASE_STRING)/include/svt_systemc_threading.12.0.hh
│   ├── svt_systemverilog_threading.hh -> ../XTOR/
monitor_amba_svs.$(RELEASE_STRING)/include/
svt_systemverilog_threading.12.0.hh
│   ├── svt_zebu_platform.hh -> ../XTOR/
monitor_amba_svs.$(RELEASE_STRING)/include/svt_zebu_platform.12.0.hh
│   ├── TopCoRoutine.hh -> ../XTOR/monitor_amba_svs.$(RELEASE_STRING)/
include/TopCoRoutine.12.0.hh
│   ├── TopScheduler.hh -> ../XTOR/monitor_amba_svs.$(RELEASE_STRING)/
include/TopScheduler.12.0.hh
│   ├── Xtor_defines.hh -> ../XTOR/monitor_amba_svs.$(RELEASE_STRING)/
include/Xtor_defines.12.0.hh
│   ├── Xtor.hh -> ../XTOR/monitor_amba_svs.$(RELEASE_STRING)/include/
Xtor.12.0.hh
│   ├── XtorScheduler.hh -> ../XTOR/monitor_amba_svs.$(RELEASE_STRING)/
include/XtorScheduler.12.0.hh
│   ├── ZebuIpRoot.hh -> ../XTOR/monitor_amba_svs.$(RELEASE_STRING)/
include/ZebuIpRoot.12.0.hh
│   └── ZFSDB.hh -> ../XTOR/monitor_amba_svs.$(RELEASE_STRING)/include/
ZFSDB.12.0.hh
├── lib
│   ├── libmonitor_amba.so -> ../XTOR/monitor_amba_svs.$(RELEASE_STRING)/
lib64/libmonitor_amba.$(RELEASE_STRING).so
│   ├── libZebuXtorSim.so -> ../XTOR/monitor_amba_svs.$(RELEASE_STRING)/
lib64/libZebuXtorSim.12.0.so
│   ├── libZebuXtor.so -> ../XTOR/monitor_amba_svs.$(RELEASE_STRING)/
lib64/libZebuXtor.12.0.so
│   └── libZebuXtorUVM.so -> ../XTOR/monitor_amba_svs.$(RELEASE_STRING)/
lib64/libZebuXtorUVM.12.0.so
```

```
├── lib64 -> lib
├── uc_xtor
├── version
│   └── Banner
├── vlog
│   ├── gtech
│   ├── gtech_lib.v -> ../XTOR/monitor_amba_svs.$(RELEASE_STRING)/vlog/
gtech_lib.12.0.v
│   ├── svt_dpi_globals.sv -> ../XTOR/monitor_amba_svs.$(RELEASE_STRING)/
vlog/svt_dpi_globals.12.0.sv
│   ├── svt_dpi_report_uvm.sv -> ../XTOR/
monitor_amba_svs.$(RELEASE_STRING)/vlog/svt_dpi_report_uvm.12.0.sv
│   ├── svt_dpi.sv -> ../XTOR/monitor_amba_svs.$(RELEASE_STRING)/vlog/
svt_dpi.12.0.sv
│   ├── svt_systemverilog_threading.sv -> ../XTOR/
monitor_amba_svs.$(RELEASE_STRING)/vlog/
svt_systemverilog_threading.12.0.sv
│   └── vcs
│       ├── monitor_amba_svs.v -> ../../XTOR/
monitor_amba_svs.$(RELEASE_STRING)/vlog/vcs/monitor_amba_svs.v
│       ├── vs_fifo_udpi.sv -> ../../XTOR/
monitor_amba_svs.$(RELEASE_STRING)/vlog/vcs/vs_fifo_udpi.12.0.sv
│       ├── vs_memory.sv -> ../../XTOR/monitor_amba_svs.$(RELEASE_STRING)/
vlog/vcs/vs_memory.12.0.sv
│       ├── zebu_vs_apb_master_udpi.sv -> ../../XTOR/
monitor_amba_svs.$(RELEASE_STRING)/vlog/vcs/
zebu_vs_apb_master_udpi.12.0.sv
│       ├── zebu_vs_complex_hwtosw_fifo_udpi.sv -> ../../XTOR/
monitor_amba_svs.$(RELEASE_STRING)/vlog/vcs/
zebu_vs_complex_hwtosw_fifo_udpi.12.0.sv
│       ├── zebu_vs_complex_rst_and_clk_udpi.sv -> ../../XTOR/
monitor_amba_svs.$(RELEASE_STRING)/vlog/vcs/
zebu_vs_complex_rst_and_clk_udpi.12.0.sv
│       ├── zebu_vs_complex_swtohw_fifo_udpi.sv -> ../../XTOR/
monitor_amba_svs.$(RELEASE_STRING)/vlog/vcs/
zebu_vs_complex_swtohw_fifo_udpi.12.0.sv
```

```
|        ├── zebu_vs_from_sw_fifo.sv -> ../../XTOR/
monitor_amba_svs.$(RELEASE_STRING)/vlog/vcs/zebu_vs_from_sw_fifo.12.0.sv
|        ├── zebu_vs_simple_hwtosw_fifo_udpi.sv -> ../../XTOR/
monitor_amba_svs.$(RELEASE_STRING)/vlog/vcs/
zebu_vs_simple_hwtosw_fifo_udpi.12.0.sv
|        ├── zebu_vs_simple_rst_and_clk_udpi.sv -> ../../XTOR/
monitor_amba_svs.$(RELEASE_STRING)/vlog/vcs/
zebu_vs_simple_rst_and_clk_udpi.12.0.sv
|        ├── zebu_vs_simple_swtohw_fifo_udpi.sv -> ../../XTOR/
monitor_amba_svs.$(RELEASE_STRING)/vlog/vcs/
zebu_vs_simple_swtohw_fifo_udpi.12.0.sv
|        └── zebu_vs_to_sw_fifo.sv -> ../../XTOR/
monitor_amba_svs.$(RELEASE_STRING)/vlog/vcs/zebu_vs_to_sw_fifo.12.0.sv
└── XTOR
    └── monitor_amba_svs.$(RELEASE_STRING)
        ├── doc
        │   └── foss
        │       └── ZX-XTOR-Library_20180911_FOSS.PDF
        ├── example
        │   ├── multiple_xtor
        │   │   ├── rtl
        │   │   │   ├── axi_slave_dut.v
        │   │   │   ├── axi_slave_memory_ctrl.v
        │   │   │   ├── dut.v
        │   │   │   └── zebu_sram2Mx.v
        │   │   ├── software
        │   │   │   ├── main.cc
        │   │   │   ├── Makefile
        │   │   │   ├── MemAccess.cc
        │   │   │   └── MemAccess.hh
        │   │   └── zebu
        │   │       ├── designFeatures
        │   │       ├── env
        │   │       │   ├── AXIRAM.utf
        │   │       │   └── xtor_32.v
        │     │       └── Makefile
        │     ├── no_xtor_master_dut_slave_dut
        │     │   ├── rtl
        │     │   │   ├── axi_slave_dut.v
        │     │   │   ├── axi_slave_memory_ctrl.v
        │     │   │   ├── dut.v
        │     │   │   └── zebu_sram2Mx.v
        │     │   ├── src
        │     │   │   ├── xtor_32.v
        │     │   │   └── xtor_64.v
        │     │   └── zebu
        │     │       └── Makefile
        │     ├── one_master_xtor_slave_dut
        │     │   ├── rtl
        │     │   │   ├── axi_slave_dut.v
        │     │   │   ├── axi_slave_memory_ctrl.v
        │     │   │   ├── dut.v
        │     │   │   └── zebu_sram2Mx.v
        │     │   ├── software
        │     │   │   ├── main.cc
        │     │   │   ├── main.o
        │     │   │   ├── Makefile
        │     │   │   ├── MemAccess.cc
        │     │   │   ├── MemAccess.hh
        │     │   │   ├── MemAccess.o
        │     │   │   └── theTest
        │     │   └── zebu
        │     │       ├── designFeatures
        │     │       ├── env
        │     │       │   ├── AXIRAM.utf
        │     │       │   └── xtor_32.v
        │     │       └── Makefile
        │     └── README
        ├── include
        │   ├── monitor_amba_svs.hh ->
monitor_amba_svs.$(RELEASE_STRING).hh
        │   ├── monitor_amba_svs.$(RELEASE_STRING).hh
        │   ├── CoRoutine.12.0.hh
        │   ├── CoRoutine.hh -> CoRoutine.12.0.hh
        │   ├── MPXtorBase.12.0.hh
        │   ├── MPXtorBase.hh -> MPXtorBase.12.0.hh
        │   ├── svt_cr_threading.12.0.hh
        │   ├── svt_cr_threading.hh -> svt_cr_threading.12.0.hh
        │   ├── svt_c_runtime_cfg.12.0.hh
        │   ├── svt_c_runtime_cfg.hh -> svt_c_runtime_cfg.12.0.hh
        │   ├── svt_c_threading.12.0.hh
        │   ├── svt_c_threading.hh -> svt_c_threading.12.0.hh
        │   ├── svt_hw_platform.12.0.hh
        │   ├── svt_hw_platform.hh -> svt_hw_platform.12.0.hh
        │   ├── svt_message_port.12.0.hh
        │   ├── svt_message_port.hh -> svt_message_port.12.0.hh
        │   ├── svt_pthread_threading.12.0.hh
        │   ├── svt_pthread_threading.hh -> svt_pthread_threading.12.0.hh
        │   ├── svt_report.12.0.hh
        │   ├── svt_report.hh -> svt_report.12.0.hh
        │   ├── svt_report_uvm.12.0.hh
        │   ├── svt_report_uvm.hh -> svt_report_uvm.12.0.hh
        │   ├── svt_simulator_platform.12.0.hh
        │   ├── svt_simulator_platform.hh -> svt_simulator_platform.12.0.hh
        │   ├── svt_systemc_threading.12.0.hh
        │   ├── svt_systemc_threading.hh -> svt_systemc_threading.12.0.hh
        │   ├── svt_systemverilog_threading.12.0.hh
        │   ├── svt_systemverilog_threading.hh ->
svt_systemverilog_threading.12.0.hh
        │   ├── svt_zebu_platform.12.0.hh
        │   ├── svt_zebu_platform.hh -> svt_zebu_platform.12.0.hh
        │   ├── TopCoRoutine.12.0.hh
        │   ├── TopCoRoutine.hh -> TopCoRoutine.12.0.hh
        │   ├── TopScheduler.12.0.hh
        │   ├── TopScheduler.hh -> TopScheduler.12.0.hh
        │   ├── Xtor.12.0.hh
        │   ├── Xtor_defines.12.0.hh
        │   ├── Xtor_defines.hh -> Xtor_defines.12.0.hh
        │   ├── Xtor.hh -> Xtor.12.0.hh
        │   ├── XtorScheduler.12.0.hh
        │   ├── XtorScheduler.hh -> XtorScheduler.12.0.hh
        │   ├── ZebuIpRoot.12.0.hh
        │   ├── ZebuIpRoot.hh -> ZebuIpRoot.12.0.hh
        │   ├── ZFSDB.12.0.hh
        │   └── ZFSDB.hh -> ZFSDB.12.0.hh
        ├── lib -> lib64
        ├── lib64
        │   ├── libmonitor_amba.so -> libmonitor_amba.$(RELEASE_STRING).so
        │   ├── libmonitor_amba.$(RELEASE_STRING).so
        │   ├── libZebuXtor.12.0.so
        │   ├── libZebuXtorSim.12.0.so
        │   ├── libZebuXtorSim.so -> libZebuXtorSim.12.0.so
        │   ├── libZebuXtor.so -> libZebuXtor.12.0.so
        │   ├── libZebuXtorUVM.12.0.so
        │   └── libZebuXtorUVM.so -> libZebuXtorUVM.12.0.so
        ├── misc
        │   ├── extension_files
        │   │   └── AXI4
        │   │       └── latest
        │   │           └── pa
        │   │               ├── extension.xml
        │   │               ├── icons
        │   │               │   ├── ace_bus_address.png
        │   │               │   ├── ace_bus_data.png
        │           │    │                   │    ├── ace_bus.png
        │           │    │                   │    ├── ace_bus_response.png
        │           │    │                   │    ├── ace_coherent_read_transaction.png
        │           │    │                   │    ├── ace_coherent_transaction.png
        │           │    │                   │    ├── ace_coherent_write_transaction.png
        │           │    │                   │    ├── ace_snoop_transaction.png
        │           │    │                   │    ├── ace_transaction.png
        │           │    │                   │    ├── axi_bus.png
        │           │    │                   │    ├── axi_bus_read_address.png
        │           │    │                   │    ├── axi_bus_read_data.png
        │           │    │                   │    ├── axi_bus_read_response.png
        │           │    │                   │    ├── axi_bus_write_address.png
        │           │    │                   │    ├── axi_bus_write_data.png
        │           │    │                   │    ├── axi_bus_write_response.png
        │           │    │                   │    ├── axi_master_data_stream_transaction.png
        │           │    │                   │    ├── axi_master_read_transaction.png
        │           │    │                   │    ├── axi_master_transaction.png
        │           │    │                   │    ├── axi_master_write_transaction.png
        │           │    │                   │    ├── axi_read_transaction.png
        │           │    │                   │    ├── axi_slave_data_stream_transaction.png
        │           │    │                   │    ├── axi_slave_read_transaction.png
        │           │    │                   │    ├── axi_slave_transaction.png
        │           │    │                   │    ├── axi_slave_write_transaction.png
        │           │    │                   │    ├── axi_stream_bus_data.png
        │           │    │                   │    ├── axi_svt_xml.gif
        │           │    │                   │    ├── axi_transaction.png
        │           │    │                   │    ├── axi_write_transaction.png
        │           │    │                   │    ├── cache_line_state.png
        │           │    │                   │    └── matchTransactions.png
        │           │    │                   └── performance
        │           │    │                        └── metrics
        │           │    │                             ├── axi4_zebu_ctrans_bandwidth.tcl
        │           │    │                             ├── axi4_zebu_ctrans_byte_count.tcl
```

```
        │   │                       ├──
axi4_zebu_ctrans_read_request_count.tcl
        │   │                       ├──
axi4_zebu_ctrans_write_request_count.tcl
        │   │                       ├── axi4_zebu_trans_byte_count.tcl
        │   │                       ├── axi4_zebu_trans_read_latency.tcl
        │   │                       └── axi4_zebu_write_latency.tcl
        │   ├── perf_post_process_script
        │   │   ├── aximaster_svs.py
        │   │   ├── base_svs.py
        │   │   ├── example
        │   │   │   ├── axi3.log
        │   │   │   ├── axi4.log
        │   │   │   ├── Makefile
        │   │   │   └── README
        │   │   └── perf_stats_gen_svs.py
        │   └── post_process_script
        │       ├── aximaster.py
        │       ├── base.py
        │       ├── example
        │       │   ├── axi3.log
        │       │   ├── axi4.log
        │       │   └── Makefile
        │       └── log2trace.py
        └── vlog
            ├── gtech
            ├── gtech_lib.12.0.v
            ├── gtech_lib.v -> gtech_lib.12.0.v
            ├── svt_dpi.12.0.sv
            ├── svt_dpi_globals.12.0.sv
            ├── svt_dpi_globals.sv -> svt_dpi_globals.12.0.sv
            ├── svt_dpi_report_uvm.12.0.sv
            ├── svt_dpi_report_uvm.sv -> svt_dpi_report_uvm.12.0.sv
            ├── svt_dpi.sv -> svt_dpi.12.0.sv
            ├── svt_systemverilog_threading.12.0.sv
            ├── svt_systemverilog_threading.sv ->
svt_systemverilog_threading.12.0.sv
            └── vcs
                ├── monitor_amba_svs.v
                ├── vs_fifo_udpi.12.0.sv
                ├── vs_fifo_udpi.sv -> vs_fifo_udpi.12.0.sv
                ├── vs_memory.12.0.sv
                ├── vs_memory.sv -> vs_memory.12.0.sv
                ├── zebu_vs_apb_master_udpi.12.0.sv
                ├── zebu_vs_apb_master_udpi.sv ->
zebu_vs_apb_master_udpi.12.0.sv
                ├── zebu_vs_complex_hwtosw_fifo_udpi.12.0.sv
                ├── zebu_vs_complex_hwtosw_fifo_udpi.sv ->
zebu_vs_complex_hwtosw_fifo_udpi.12.0.sv
                ├── zebu_vs_complex_rst_and_clk_udpi.12.0.sv
                ├── zebu_vs_complex_rst_and_clk_udpi.sv ->
zebu_vs_complex_rst_and_clk_udpi.12.0.sv
                ├── zebu_vs_complex_swtohw_fifo_udpi.12.0.sv
                ├── zebu_vs_complex_swtohw_fifo_udpi.sv ->
zebu_vs_complex_swtohw_fifo_udpi.12.0.sv
                ├── zebu_vs_from_sw_fifo.12.0.sv
               ├── zebu_vs_from_sw_fifo.sv -> zebu_vs_from_sw_fifo.12.0.sv
                ├── zebu_vs_simple_hwtosw_fifo_udpi.12.0.sv
                ├── zebu_vs_simple_hwtosw_fifo_udpi.sv ->
zebu_vs_simple_hwtosw_fifo_udpi.12.0.sv
                ├── zebu_vs_simple_rst_and_clk_udpi.12.0.sv
                ├── zebu_vs_simple_rst_and_clk_udpi.sv ->
zebu_vs_simple_rst_and_clk_udpi.12.0.sv
                ├── zebu_vs_simple_swtohw_fifo_udpi.12.0.sv
                ├── zebu_vs_simple_swtohw_fifo_udpi.sv ->
zebu_vs_simple_swtohw_fifo_udpi.12.0.sv
                ├── zebu_vs_to_sw_fifo.12.0.sv
                └── zebu_vs_to_sw_fifo.sv -> zebu_vs_to_sw_fifo.12.0.sv
```

3 Features

The AMBA AMBA AXI Monitor transactor supports the following features:

¢ *Log Dumping*

¢ *Log2trace Utility*

¢ *perf_stats_gen_svs utility*

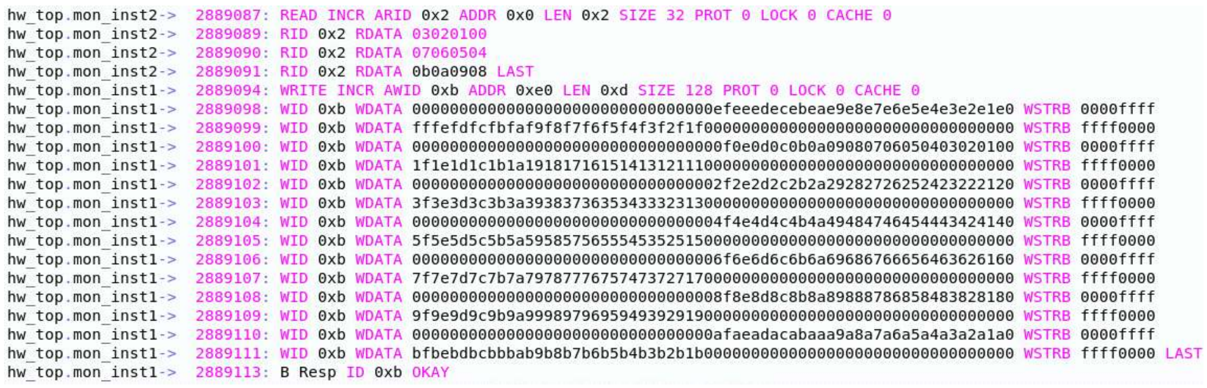
¢ *Verdi Protocol Analyzer (PA)*

3.1 Log Dumping

AMBA AMBA AXI Monitor captures traffic on AXI3/AXI4 interface and prints the information in a log file.

AMBA AMBA AXI Monitor generates separate log files for each instance of monitor. Also, it generates a combined log file containing information for all instances.

The following figure illustrates a sample log file:



```
hw_top.mon_inst2->  2889087: READ INCR ARID 0x2 ADDR 0x0 LEN 0x2 SIZE 32 PROT 0 LOCK 0 CACHE 0
hw_top.mon_inst2->  2889089: RID 0x2 RDATA 03020100
hw_top.mon_inst2->  2889090: RID 0x2 RDATA 07060504
hw_top.mon_inst2->  2889091: RID 0x2 RDATA 0b0a0908 LAST
hw_top.mon_inst1->  2889094: WRITE INCR AWID 0xb ADDR 0xe0 LEN 0xd SIZE 128 PROT 0 LOCK 0 CACHE 0
hw_top.mon_inst1->  2889098: WID 0xb WDATA 00000000000000000000000000000000efeeedecebeae9e8e7e6e5e4e3e2e1e0 WSTRB 0000ffff
hw_top.mon_inst1->  2889099: WID 0xb WDATA fffefdfcfbfaf9f8f7f6f5f4f3f2f1f000000000000000000000000000000000 WSTRB ffff0000
hw_top.mon_inst1->  2889100: WID 0xb WDATA 000000000000000000000000000000000f0e0d0c0b0a09080706050403020100 WSTRB 0000ffff
hw_top.mon_inst1->  2889101: WID 0xb WDATA 1f1e1d1c1b1a1918171615141312111000000000000000000000000000000000 WSTRB ffff0000
hw_top.mon_inst1->  2889102: WID 0xb WDATA 000000000000000000000000000000002f2e2d2c2b2a29282726252423222120 WSTRB 0000ffff
hw_top.mon_inst1->  2889103: WID 0xb WDATA 3f3e3d3c3b3a3938373635343332313000000000000000000000000000000000 WSTRB ffff0000
hw_top.mon_inst1->  2889104: WID 0xb WDATA 000000000000000000000000000000004f4e4d4c4b4a49484746454443424140 WSTRB 0000ffff
hw_top.mon_inst1->  2889105: WID 0xb WDATA 5f5e5d5c5b5a5958575655545352515000000000000000000000000000000000 WSTRB ffff0000
hw_top.mon_inst1->  2889106: WID 0xb WDATA 000000000000000000000000000000006f6e6d6c6b6a69686766656463626160 WSTRB 0000ffff
hw_top.mon_inst1->  2889107: WID 0xb WDATA 7f7e7d7c7b7a7978777675747372717000000000000000000000000000000000 WSTRB ffff0000
hw_top.mon_inst1->  2889108: WID 0xb WDATA 000000000000000000000000000000008f8e8d8c8b8a89888786858483828180 WSTRB 0000ffff
hw_top.mon_inst1->  2889109: WID 0xb WDATA 9f9e9d9c9b9a9998979695949392919000000000000000000000000000000000 WSTRB ffff0000
hw_top.mon_inst1->  2889110: WID 0xb WDATA 00000000000000000000000000000000afaeadacabaaa9a8a7a6a5a4a3a2a1a0 WSTRB 0000ffff
hw_top.mon_inst1->  2889111: WID 0xb WDATA bfbebdbcbbbab9b8b7b6b5b4b3b2b1b000000000000000000000000000000000 WSTRB ffff0000 LAST
hw_top.mon_inst1->  2889113: B Resp ID 0xb OKAY
```

FIGURE 2. Printing the Log

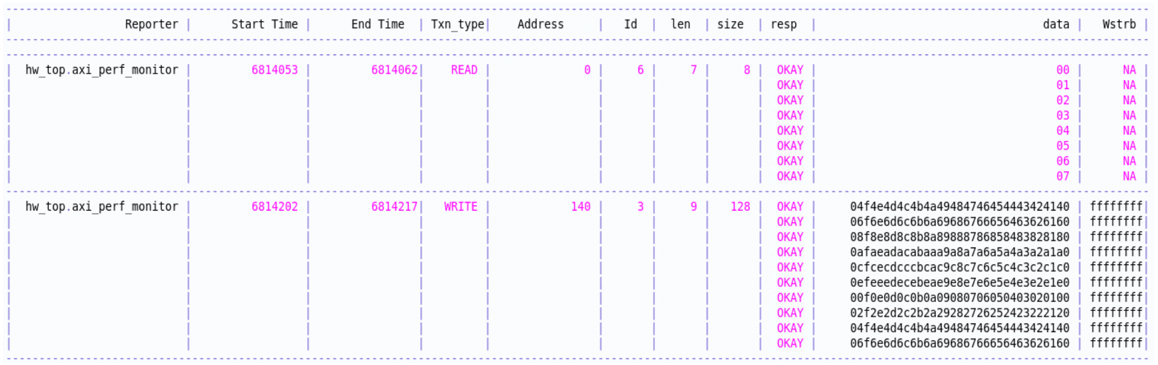
Each line in the log file corresponds to bus activity and shows the hierarchal path of the monitor instance from where it was captured.

Each log file is named according to its unique ID that is passed to Monitor constructor in the testbench. You can also override the log name for a particular instance using the set_log_name() API.

3.2 Log2trace Utility

The log2trace utility processes monitor-generated log into a formatted transaction trace file. This trace file shows all the bus-activity related to a transaction.

The following figure illustrates a sample trace file:



```
-------------------------------------------------------------------------------------------------------------------------------------------------------
|               Reporter |     Start Time |     End Time  | Txn_type|   Address      |  Id  |  len  | size  | resp  |                              data |  Wstrb |
-------------------------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------------------------------
|  hw_top.axi_perf_monitor |        6814053 |       6814062|    READ |              0 |    6 |     7 |     8 |  OKAY |                                00 |     NA |
|                          |                |              |         |                |      |       |       |  OKAY |                                01 |     NA |
|                          |                |              |         |                |      |       |       |  OKAY |                                02 |     NA |
|                          |                |              |         |                |      |       |       |  OKAY |                                03 |     NA |
|                          |                |              |         |                |      |       |       |  OKAY |                                04 |     NA |
|                          |                |              |         |                |      |       |       |  OKAY |                                05 |     NA |
|                          |                |              |         |                |      |       |       |  OKAY |                                06 |     NA |
|                          |                |              |         |                |      |       |       |  OKAY |                                07 |     NA |
-------------------------------------------------------------------------------------------------------------------------------------------------------
|  hw_top.axi_perf_monitor |        6814202 |       6814217|   WRITE |            140 |    3 |     9 |   128 |  OKAY |  04f4e4d4c4b4a49484746454443424140 | ffffffff|
|                          |                |              |         |                |      |       |       |  OKAY |  06f6e6d6c6b6a69686766656463626160 | ffffffff|
|                          |                |              |         |                |      |       |       |  OKAY |  08f8e8d8c8b8a89888786858483828180 | ffffffff|
|                          |                |              |         |                |      |       |       |  OKAY |  0afaeadacabaaa9a8a7a6a5a4a3a2a1a0 | ffffffff|
|                          |                |              |         |                |      |       |       |  OKAY |  0cfcecdcccbcac9c8c7c6c5c4c3c2c1c0 | ffffffff|
|                          |                |              |         |                |      |       |       |  OKAY |  0efeeedecebeae9e8e7e6e5e4e3e2e1e0 | ffffffff|
|                          |                |              |         |                |      |       |       |  OKAY |  00f0e0d0c0b0a09080706050403020100 | ffffffff|
|                          |                |              |         |                |      |       |       |  OKAY |  02f2e2d2c2b2a29282726252423222120 | ffffffff|
|                          |                |              |         |                |      |       |       |  OKAY |  04f4e4d4c4b4a49484746454443424140 | ffffffff|
|                          |                |              |         |                |      |       |       |  OKAY |  06f6e6d6c6b6a69686766656463626160 | ffffffff|
-------------------------------------------------------------------------------------------------------------------------------------------------------
```

FIGURE 3. Log2trace Utility

Script Usage

```
$ZEBU_IP_ROOT/monitor_amba_svs/misc/post_process_script/log2trace.py
--file <input logfile> --datawidth 128
```

where

- data width is <width of data bus>
- Formatted trace file generated is zebu_vs_trace.log

3.3 perf_stats_gen_svs utility

This section explains the following topics:

- Purpose
- Definitions
- Use Model

3.3.1 Purpose

The perf_stats_gen_svs utility extracts performance stats from monitor generated log. It generates a perf. summary log, which displays the following information:

- Read/Write Latency for each txn: is generated in the perf_<(read/write)>_latency.log
- No. of bytes on RDATA/WDATA channel: Is generated in the perf_<(read/write)>_bytes.log
- Read/Write Bandwidth: Is generated in the perf_<(read/write)>_bandwidth.log
- Read/Write Bandwidth graph: Is available in the <(read/write)>_bandwidth.png
- Read/Write Outstanding txns: Is available in the perf_<(read/write)>_outstanding.log

3.3.2 Definitions

The definition of performance statistics is described as below:

- Latency (Read/Write)
  - Read Latency: cycle difference between Read Request & first Read Response.
- Write Latency: cycle difference between last Write Data Beat & Write Response.
- Bandwidth (Read/Write)
  - number of bytes (AxSIZE) on RDATA/WDATA per cycles
  - Two methods:
    - Running Average: Moves through the read/write bytes in log and computes the running average by adding the current number of bytes

to the sum of the preceding byte count and dividing by the total cycles.

® Window Average: Computes average bandwidth for each window of "n" cycles, where n is input provided by user

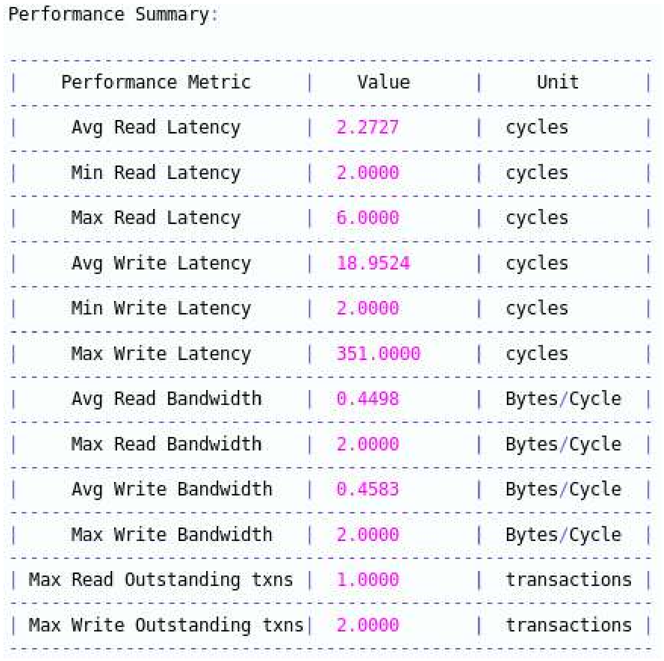
¢ Outstanding transactions

¶ Write: number of write txns waiting for write response

¶ Read :number of read txns waiting for read response

3.3.3 Sample Files

The following figure illustrates the Performance Summary log:



```
Performance Summary:

------------------------------------------------------------
|    Performance Metric     |     Value     |     Unit      |
------------------------------------------------------------
|     Avg Read Latency      |  2.2727       |  cycles       |
------------------------------------------------------------
|     Min Read Latency      |  2.0000       |  cycles       |
------------------------------------------------------------
|     Max Read Latency      |  6.0000       |  cycles       |
------------------------------------------------------------
|     Avg Write Latency     |  18.9524      |  cycles       |
------------------------------------------------------------
|     Min Write Latency     |  2.0000       |  cycles       |
------------------------------------------------------------
|     Max Write Latency     |  351.0000     |  cycles       |
------------------------------------------------------------
|     Avg Read Bandwidth    |  0.4498       |  Bytes/Cycle  |
------------------------------------------------------------
|     Max Read Bandwidth    |  2.0000       |  Bytes/Cycle  |
------------------------------------------------------------
|     Avg Write Bandwidth   |  0.4583       |  Bytes/Cycle  |
------------------------------------------------------------
|     Max Write Bandwidth   |  2.0000       |  Bytes/Cycle  |
------------------------------------------------------------
| Max Read Outstanding txns |  1.0000       |  transactions |
------------------------------------------------------------
| Max Write Outstanding txns|  2.0000       |  transactions |
------------------------------------------------------------
```

FIGURE 4. Performance Summary Log

The following figure illustrates the Write Bandwidth chart using the window method:

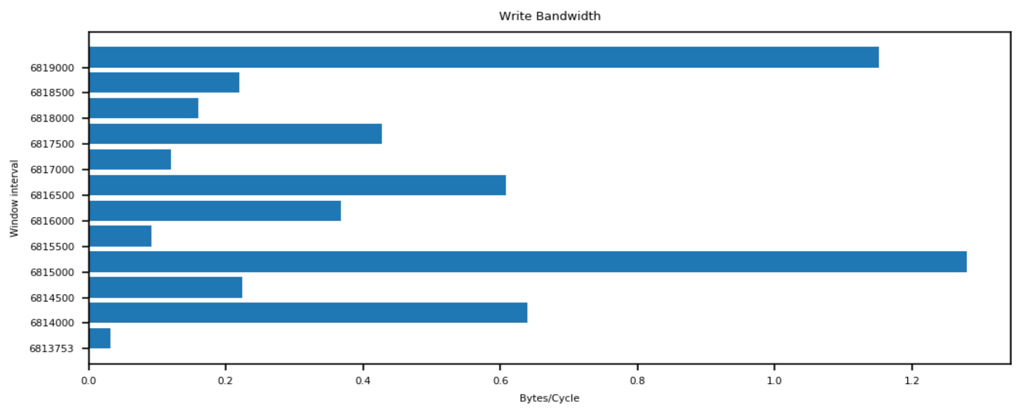


FIGURE 5. Write Bandwidth (window method)

The following figure illustrates the Write Bandwidth chart using the running average method:

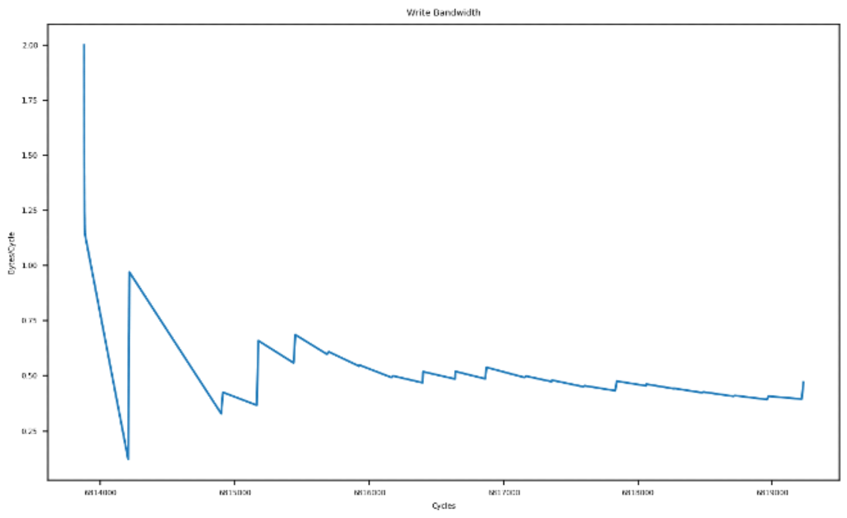


FIGURE 6. Write Bandwidth (running average method)

3.3.4 Use Model

The following is the use model for the perf_stats_gen_svs utility:

```
$ZEBU_IP_ROOT/monitor_amba_svs/misc/perf_post_process_script/
perf_stats_gen_svs.py --file <input logfile>
```

By default, summary log shows perf. metrics in terms of cycles. Also, by default, the script calculates bandwidth by using running avg method:

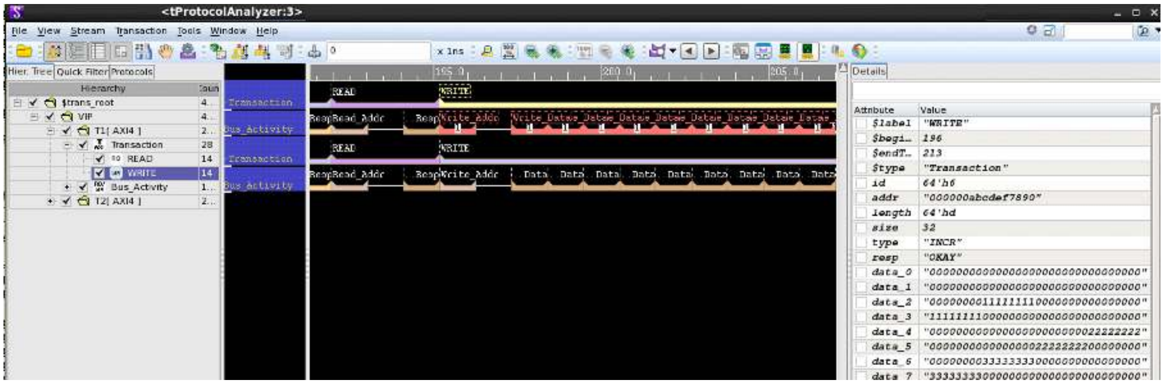
Optional arguments

¢ --window <window size in cycles>: to calculate moving window bandwidth, instead of default running avg bandwidth.

¢ --clk_freq <emulation frequency in Mhz>: to view summary log in terms of time/freq instead of cycles.

3.4 Verdi Protocol Analyzer (PA)

Verdi Protocol Analyzer (PA) shows bus traffic in a GUI, along with relationships for a AXI transaction.



Attribute	Value
$label	"WRITE"
$begi..	196
$endT..	213
$type	"Transaction"
id	64'h6
addr	"000000abcdef7890"
length	64'hd
size	32
type	"INCR"
resp	"OKAY"
data_0	"00000000000000000000000000000000"
data_1	"00000000000000000000000000000000"
data_2	"00000000111111110000000000000000"
data_3	"11111111000000000000000000000000"
data_4	"00000000000000000000000022222222"
data_5	"00000000000000002222222200000000"
data_6	"00000000333333330000000000000000"
data_7	"33333333000000000000000000000000"

FIGURE 7. Protocol Analyzer GUI

PA database is generated in the form of transactional fsdb. Use the following command to open the same:

```
verdi -lca -ssf "pa_axi.fsdb" -workMode protocolDebug
```

3.5 Timestamp Support

With zCei clock ports, monitor reported clock cycle count is considered as a reference for AMBA txns. However, the RTL clock feature support in ZeBu, provides simulation-like timestamping in emulation as well. If the design uses RTL clock and has timestamp feature enabled in UTF, it is now possible to obtain a global timestamp that represents time in presence of different clocks in emulation & can be used instead of clock counters traditionally used.

The following are the prerequisites to enable global timestamping in AMBA Monitor:

¢ Design is compiled using RTL clock (in a single clock group), which is implemented by using either clockDelayPort or # delays. It also requires the following UTF command:

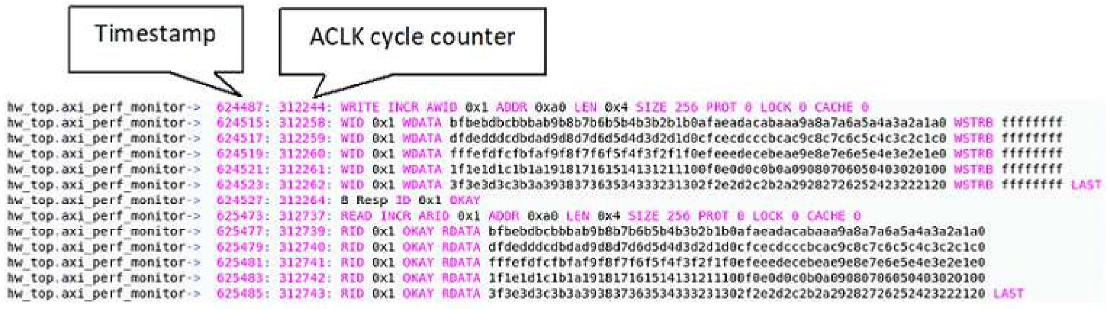
```
clock_delay -module {hw_top} -debug true
```

¢ Timestamping is enabled in UTF file by using following command:

```
zemi3 -timestamp true
```

¢ Is used only for emulation, not VCS simulation.

If above prerequisites are met, you can configure global timestamping feature in AMBA monitor by using the setParam(USE_GLOBAL_TIME,1) API before enabling the monitor. Else, AMBA Monitor uses traditional clock counters. The following figure illustrates sample logs showing timestamp, when the timestamping feature is enabled.



```
hw_top.axi_perf_monitor->  624487: 312244: WRITE INCR AWID 0x1 ADDR 0xa0 LEN 0x4 SIZE 256 PROT 0 LOCK 0 CACHE 0
hw_top.axi_perf_monitor->  624515: 312258: WID 0x1 WDATA bfbebdbcbbbab9b8b7b6b5b4b3b2b1b0afaeadacabaaa9a8a7a6a5a4a3a2a1a0 WSTRB ffffffff
hw_top.axi_perf_monitor->  624517: 312259: WID 0x1 WDATA dfdedddcdbdad9d8d7d6d5d4d3d2d1d0cfcecdcccbcac9c8c7c6c5c4c3c2c1c0 WSTRB ffffffff
hw_top.axi_perf_monitor->  624519: 312260: WID 0x1 WDATA fffefdfcfbfaf9f8f7f6f5f4f3f2f1f0efeeedecebeae9e8e7e6e5e4e3e2e1e0 WSTRB ffffffff
hw_top.axi_perf_monitor->  624521: 312261: WID 0x1 WDATA 1f1e1d1c1b1a191817161514131211100f0e0d0c0b0a09080706050403020100 WSTRB ffffffff
hw_top.axi_perf_monitor->  624523: 312262: WID 0x1 WDATA 3f3e3d3c3b3a393837363534333231302f2e2d2c2b2a292827262524232221 20 WSTRB ffffffff LAST
hw_top.axi_perf_monitor->  624527: 312264: B Resp ID 0x1 OKAY
hw_top.axi_perf_monitor->  625473: 312737: READ INCR ARID 0x1 ADDR 0xa0 LEN 0x4 SIZE 256 PROT 0 LOCK 0 CACHE 0
hw_top.axi_perf_monitor->  625477: 312739: RID 0x1 OKAY RDATA bfbebdbcbbbab9b8b7b6b5b4b3b2b1b0afaeadacabaaa9a8a7a6a5a4a3a2a1a0
hw_top.axi_perf_monitor->  625479: 312740: RID 0x1 OKAY RDATA dfdedddcdbdad9d8d7d6d5d4d3d2d1d0cfcecdcccbcac9c8c7c6c5c4c3c2c1c0
hw_top.axi_perf_monitor->  625481: 312741: RID 0x1 OKAY RDATA fffefdfcfbfaf9f8f7f6f5f4f3f2f1f0efeeedecebeae9e8e7e6e5e4e3e2e1e0
hw_top.axi_perf_monitor->  625483: 312742: RID 0x1 OKAY RDATA 1f1e1d1c1b1a191817161514131211100f0e0d0c0b0a09080706050403020100
hw_top.axi_perf_monitor->  625485: 312743: RID 0x1 OKAY RDATA 3f3e3d3c3b3a393837363534333231302f2e2d2c2b2a29282726252423222120 LAST
```

In the above figure, the first value is timestamp and second value is ACLK cycle counter.

3.6 Reporting Binary Database

Using binary database reduces impact of PA on runtime performance.

AMBA Monitor can be configured to report binary database only. During runtime, monitor reports a binary database that is not user-readable. Once runtime is over, this database can be post-processed by monitor to generate logs and/or Protocol Analyzer (PA) fsdb. Emulator host machine is not required to run post-process executable.

To enable reporting of binary database, use BINARY_ONLY mode in monitor's constructor in testbench.

The following is an example of binary reporting feature:

```
monitor_amba_svs* mon = new
monitor_amba_svs(1,<path>,AXI3,BINARY_ONLY,<runtime object>)


Following Makefile target can be used to post-process binary database after
runtime is over.


binary_postprocess:
                -ln -s $(ZEBU_IP_ROOT)/monitor_amba_svs/bin/amba_postprocess
$(rundir)
                cd $(rundir) && LD_LIBRARY_PATH=$$VERDI_HOME/share/
fsdbTrans_API/lib/linux64:$$LD_LIBRARY_PATH &&  ./amba_postprocess
OUTPUT_MODE=FSDB_LOG
```

where OUTPUT_MODE can be:

- LOG_ONLY
- FSDB_ONLY
- FSDB_LOG

3.7 Callbacks

AMBA Monitor supports callbacks. These callbacks can be registered in testbench by user.

AMBA monitor supports callbacks, which are registered in testbench. Whenever, a request or transaction completion is seen on monitor interface, a callback is triggered and logic is specified for dynamic score-boarding using these callbacks.

TABLE 1 AMBA AXI Monitor Callbacks

Callback	**Condition when triggered**	**Argument description**
registerWaddrCB (void(*waddrCB)(void* xtor,AXIMChanInfo info), void* context = NULL)	Write request (AW)	xtor : pointer to xtor class object if "context" argument is not set by user, otherwise set to "context" provided info : channel info received, refer to monitor_amba_defines_svs.hh for enum definition context : reference to context that user can set (optional).
registerWriteTxnCB(void(*wrTxnCB)(void* xtor,AXIMChanInfo info ,AXIMData *data,AXIMResp *resp) , void* context = NULL)	Write transaction completion (AW + W … W + B)	xtor : pointer to xtor class object if "context" argument is not set by user, otherwise set to "context" provided info : channel info received, refer to monitor_amba_defines_svs.hh for enum definition data : AXIMData holding data received for Write transaction ,refer to monitor_amba_defines_svs.hh. resp : reference to AXIMResp class that contains Resp ,refer to monitor_amba_defines_svs.hh context : reference to context that user can set (optional).

TABLE 1 AMBA AXI Monitor Callbacks

Callback	**Condition when triggered**	**Argument description**
registerReadTxnCB(void(*rdTxnCB)(void* xtor,AXIMChanInfo info ,AXIMData *data,AXIMResp *resp) , void* context = NULL)	Read request (AR)	xtor : pointer to xtor class object if "context" argument is not set by user, otherwise set to "context" provided data : AXIMData holding data received for Write transaction ,refer to monitor_amba_defines_svs.hh. resp : reference to AXIMResp class that contains Resp ,refer to monitor_amba_defines_svs.hh. context : reference to context that user can set (optional).
registerRaddrCB(void(*raddrCB)(void* xtor,AXIMChanInfo info), void* context = NULL)	Read transaction completion (AR + R ... R)	xtor : pointer to xtor class object if "context" argument is not set by user, otherwise set to "context" provided info : channel info received, refer to monitor_amba_defines_svs.hh for enum definition context : reference to context that user can set (optional).

3.8 Filtering

The AMBA AXI Monitor supports following filtering options, which can be configured using setInstanceParam() API.

TABLE 2 AMBA AXI Monitor Filtering Options

setInstanceParam(config , value)		**Description**
config	**value**	
DISABLE_RD_TXN	1	If true, Read txns are not dumped
DISABLE_WR_TXN	1	If true, Write txns are not dumped
RADDR_FILTER_MIN	address (maximum 64 bit supported)	If set, Read txns with addr less than this value are filtered out
RADDR_FILTER_MAX		If set, Read txns with addr more than this value are filtered out
WADDR_FILTER_MIN		If set, Write txns with addr less than this value are filtered out
WADDR_FILTER_MAX		If set, Write txns with addr more than this value are filtered out

3.9 Other Configurations

AMBA AXI Monitor provides additional configuration parameters, which can be used using the setParam API and AXIMONITOR_PARAM enum defined in the monitor_amba_svs include file.

To enable a parameter, use the `setParam (PARAMETER, 1)` API in the testbench.

TABLE 3 Other AMBA AXI Monitor Configurations

PARAMETER	**Function**
LOG_DUMPING_MODE	Used to set log dumping mode, can be filled using the LOG_DUMP_MODE enum defined in the monitor_amba_svs include file.
ENABLE_FORCE_CLOSE	If enabled, close() API stops dumping without waiting for completion of pending transactions.
USE_GLOBAL_TIME	If enabled, timestamping is enabled. Please refer to 3.5 for detailed explanation.
IGNORE_M_BRESP	If enabled, callbacks ignore write response for maturation of Write transaction callback(registerWriteTxnCB).
DISABLE_LOG_BUFFERI NG	If enabled, no buffering is used for dumping into log files.
RAW_LOG_DUMPING	If enabled, transaction completion logic is not used in monitor. Only logs can be dumped in this mode.

4 Hardware Interface

The following table lists the AMBA AMBA AXI Monitor, monitor_amba_svs, signals.

TABLE 4 AMBA Monitor Signals

Symbol	Size in bits	Direction	Description
ACLK	1	Input	AXI clock
ARESETN	1		AXI reset signal driven by DUT
AWADDR	ADDR_WIDTH		AW address
AWVALID	1		AW valid
AWID, AWLEN, AWSIZE, AWBURST, AWLOCK, AWCACHE,			
AWPROT	ID_WIDTH, 8 , 3 , 2, 4 ,3		AW control signals:
AWREADY	1		Slave AWREADY signal. Indicates that the
Slave is ready to receive AW channel			
ARADDR	ADDR_WIDTH		AR address
ARVALID	1		AR valid
ARID, ARLEN, ARSIZE, ARBURST, ARLOCK, ARCACHE,			
ARPROT	ID WIDTH, 8, 3 , 2, 4 ,3		AR control signals

TABLE 4 AMBA Monitor Signals

Symbol	Size in bits	Direction	Description
ARREADY	1		Slave ARREADY signal. Indicates that the
Slave is ready to receive AR channel			
WDATA	DATA_WIDTH		Data to write
WID	ID_WIDTH		AXI3 only. ID value
WSTRB	DATA_WIDTH/8		Byte-lane of the data to write
WVALID	1		W valid
WLAST	1		Indicates last data of a burst.
WREADY	1		Indicates that the Slave is ready to receive data
BRESP	2		Response from the Slave
BID	ID_WIDTH		ID value of the response
BVALID	1		W valid
BREADY	1		Master indicates if ready to get response
RDATA	DATA_WIDTH		Read data
RID	ID_WIDTH	Input	ID value of the response
RRESP	2		Response from the Slave
RVALID	1		R valid
RLAST	1		Indicates last data of a burst
RREADY	1		Master indicates if ready to get response
AWQOS,AWREGION	4,4		AXI4 only. Read address channel QOS & Region control
ARQOS,ARREGION	4,4		AXI4 only. Write address channel QOS & Region control

TABLE 4 AMBA Monitor Signals

Symbol	Size in bits	Direction	Description
AWUSER,ARUSER,WUSER	USER_WIDTH		AXI4 only. User signal for AW, AR & W channel respectively.
BUSER, RUSER	USER_WIDTH		AXI4 only. User signal for B & R channel respectively
AWSNOOP,AWDOMAIN,AWBAR	3,2,2		ACELite/ACE only. Write address channel Snoop, Domain & Barrier control

You can connect the monitor_amba_svs to AXI3, AXI4 or ACELite interface and configure it using the testbench. Signals that are not applicable can be tied down to zero. For example , if bus type is AXI3, all AXI4 & ACELite signals can be hard wired to zero.

5 Software Interface

This section explains the following APIs in the AMBA AMBA AXI Monitor transactor:

- *monitor_amba_svs()*
- *init()*
- *enable()*
- *close()*
- *set_log_name()*
- *setParam()*

5.1 monitor_amba_svs()

Used to construct monitor_amba_svs handle:

The following is the syntax for the monitor_amba_svs:

```
monitor_amba_svs(int instance, const char* scope, PROTOCOL_MODE
protocol_mode, DUMP_MODE dump_mode)
```

Parameters:

¢ **int instance**: instance number that is used in monitor_amba_svs instance in hw_top

¢ **const char* scope**: heirarchical path to monitor_amba_svs instance

¢ **PROTOCOL_MODE protocol_mode**: AMBA protocol of the bus on which monitor_amba_svs is connected, see PROTOCOL_MODE enum

¢ **DUMP_MODE dump_mode**: Dumping mode. For more options, see DUMP_MODE enum

¢ **svt_c_runtime_cfg runtime**: svt runtime handle. For usage, see the example

5.2 init()

Used to initialize monitor_amba_svs.

The following is the syntax for init() method:

```
static void init(int argc,char* argv[], const char* fsdbFile);
```

Parameters:

- **int argc** : argument count, for usage see examples
- **char* argv** : argument value, for usage see examples
- **const char* fsdbFile**: name of the PA fsdb generated

5.3 enable()

Used to enable/disable monitor at runtime.

The following is the syntax for enable() method:

```
void enable(int _enable) ;
```

Parameters:

¢ int _enable: use (0) to disable or (1) to enable dumping

5.4 close()

Used to close monitor_amba_svs.

The following is the syntax for close() method:

```
void close()
```

5.5 set_log_name()

Used to set name of monitor_amba_svs log. If this API is not used, default name is considered.

```
void set_log_name(char* logfile_name)
```

Parameters:

¢ std::string logfile_name: name of logfile to be used for this monitor instance

5.6 setParam()

Used to set monitor_amba_svs parameters to a specified value.

```
static void (AXIMONITOR_PARAM param,int value) ;
```

Parameters:

¢ **param**: Name of parameter to be set. For more information, see AXIMONITOR_PARAM enum.

¢ **value**: value to be assigned to above parameter.

6 Example

Given below is AMBA AMBA AXI Monitor connected with ZEMI3 AXIMaster transactor .

Hardware top:

```
`define DATA_WIDTH       256
`define ADDR_WIDTH       32
`define ID_WIDTH         4
`define WSTRB_WIDTH      `DATA_WIDTH >> 3
`define BLEN_WIDTH       4
`define XTOR_DEBUG_WIDTH 10

module hw_top();

// Parameter Declaration
parameter CLOCK_PERIOD     = 20;
parameter RST_PERIOD       = 1000;

parameter DATA_WIDTH       = `DATA_WIDTH  ;
parameter ADDR_WIDTH       = `ADDR_WIDTH  ;
parameter ID_WIDTH         = `ID_WIDTH    ;
parameter WSTRB_WIDTH      = `WSTRB_WIDTH ;
parameter BLEN_WIDTH       = `BLEN_WIDTH  ;
parameter XTOR_DEBUG_WIDTH = `XTOR_DEBUG_WIDTH;

// AXI Bus Signals

// Write Data Channel
wire  [DATA_WIDTH-1:0]     AXI_WDATA;
wire  [ID_WIDTH-1:0]       AXI_WID;
wire  [WSTRB_WIDTH-1:0]    AXI_WSTRB;
wire                       AXI_WLAST;
wire                       AXI_WVALID;
wire                       AXI_WREADY;

// Write Addr Channel
wire  [ADDR_WIDTH-1:0]     AXI_AWADDR;
wire  [ID_WIDTH-1:0]       AXI_AWID;
wire  [3:0]                AXI_AWLEN;
wire  [2:0]                AXI_AWSIZE;
wire  [2:0]                AXI_AWPROT;
wire  [1:0]                AXI_AWBURST;
wire  [1:0]                AXI_AWLOCK;
wire  [3:0]                AXI_AWCACHE;
wire                       AXI_AWVALID;
wire                       AXI_AWREADY;

// Write Resp channel
wire  [ID_WIDTH-1:0]       AXI_BID;
wire  [1:0]                AXI_BRESP;
wire                       AXI_BVALID;
wire                       AXI_BREADY;

// Read Addr Channel
wire  [ADDR_WIDTH-1:0]     AXI_ARADDR;
wire  [ID_WIDTH-1:0]       AXI_ARID;
wire  [3:0]                AXI_ARLEN;
wire  [2:0]                AXI_ARSIZE;
wire  [2:0]                AXI_ARPROT;
wire  [1:0]                AXI_ARBURST;
wire  [1:0]                AXI_ARLOCK;
wire  [3:0]                AXI_ARCACHE;
wire                       AXI_ARVALID;
wire                           AXI_ARREADY;

// Read Resp Channel
wire   [DATA_WIDTH-1:0]        AXI_RDATA;
wire   [ID_WIDTH-1:0]          AXI_RID;
wire   [1:0]                   AXI_RRESP;
wire                           AXI_RLAST;
wire                           AXI_RVALID;
wire                           AXI_RREADY;

// Clock and reset handling
reg                            reset;
`ifdef COEMU
    wire                       clk_local;
    wire                       rst_local;
    wire                       resetcounter;
    reg [15:0]                 resetcnt;

    zceiClockPort zceiClockPort_clk_i (
        .cclock(clk_local),
        .cresetn(rst_local)
      );

    assign resetcounter = ~ rst_local;

    always@ (posedge clk_local or posedge resetcounter)
    begin
        if(resetcounter) begin
            resetcnt <= 16'h0000;
            reset <= 1'b0;
        end else if (resetcnt[6] == 1'b1) begin
            reset <= 1'b1;
        end else begin
```

```
            resetcnt <= resetcnt + 1;
            reset <= 1'b0;
        end
    end
`else
    reg                     clk_local;
    always begin
        forever #(CLOCK_PERIOD/2)
            clk_local = ~clk_local;
    end

    initial
    begin
        reset                        = 0;
        clk_local                    = 0;
        #RST_PERIOD;
        reset                        = 1;
    end
`endif

bit [XTOR_DEBUG_WIDTH-1 :0] xtor_info;

xtor_amba_master_axi3_svs #(  .DATA_WIDTH (DATA_WIDTH),
                        .ADDR_WIDTH (ADDR_WIDTH),
                        .ID_WIDTH   (ID_WIDTH),
                        .WSTRB_WIDTH(WSTRB_WIDTH),
                        .BLEN_WIDTH (BLEN_WIDTH),
                        .XTOR_DEBUG_WIDTH(XTOR_DEBUG_WIDTH))

                                   master_node_i (

                                  .ACLK(clk_local),
                                  .ARESETn(reset),
                                                        .WDATA(AXI_WDATA),
                                                        .WID(AXI_WID),
                                                        .WSTRB(AXI_WSTRB),
                                                        .WLAST(AXI_WLAST),
                                                        .WVALID(AXI_WVALID),
                                                        .WREADY(AXI_WREADY),
                                                        .AWADDR(AXI_AWADDR),
                                                        .AWID(AXI_AWID),
                                                        .AWLEN(AXI_AWLEN),
                                                        .AWSIZE(AXI_AWSIZE),
                                                        .AWPROT(AXI_AWPROT),
                                                        .AWBURST(AXI_AWBURST),
                                                        .AWLOCK(AXI_AWLOCK),
                                                        .AWCACHE(AXI_AWCACHE),
                                                        .AWVALID(AXI_AWVALID),
                                                        .AWREADY(AXI_AWREADY),
                                                        .BID(AXI_BID),
                                                        .BRESP(AXI_BRESP),
                                                        .BVALID(AXI_BVALID),
                                                        .BREADY(AXI_BREADY),
                                                        .ARADDR(AXI_ARADDR),
                                                        .ARID(AXI_ARID),
                                                        .ARLEN(AXI_ARLEN),
                                                        .ARSIZE(AXI_ARSIZE),
                                                        .ARPROT(AXI_ARPROT),
                                                        .ARBURST(AXI_ARBURST),
                                                        .ARLOCK(AXI_ARLOCK),
                                                        .ARCACHE(AXI_ARCACHE),
                                                        .ARVALID(AXI_ARVALID),
                                                        .ARREADY(AXI_ARREADY),
                                                        .RDATA(AXI_RDATA),
                                                        .RID(AXI_RID),
                                                        .RRESP(AXI_RRESP),
                                  .RLAST(AXI_RLAST),
                                  .RVALID(AXI_RVALID),
                                  .RREADY(AXI_RREADY),
                                  .xtor_info_port(xtor_info)
                              );

AXI_SLAVE_DUT                #(
                                  .DATA_WRITE_WIDTH(DATA_WIDTH),
                                  .DATA_READ_WIDTH(DATA_WIDTH),
                                  .ID_WIDTH(ID_WIDTH),
                                  .WSTRB_WIDTH(WSTRB_WIDTH)
                              )

         dut(
                                .AXI_WDATA(AXI_WDATA),
                                .AXI_WSTRB(AXI_WSTRB),
                                .AXI_WLAST(AXI_WLAST),
                                .AXI_WVALID(AXI_WVALID),
                                .AXI_WREADY(AXI_WREADY),
                                .AXI_AWADDR(AXI_AWADDR),
                                .AXI_AWID(AXI_AWID),
                                .AXI_AWLEN(AXI_AWLEN),
                                .AXI_AWSIZE(AXI_AWSIZE),
                                .AXI_AWPROT(AXI_AWPROT),
                                .AXI_AWBURST(AXI_AWBURST),
                                .AXI_AWLOCK(AXI_AWLOCK),
                                .AXI_AWCACHE(AXI_AWCACHE),
                                .AXI_AWVALID(AXI_AWVALID),
                                .AXI_AWREADY(AXI_AWREADY),
                                .AXI_BID(AXI_BID),
                                .AXI_BRESP(AXI_BRESP),
                                .AXI_BVALID(AXI_BVALID),
                                .AXI_BREADY(AXI_BREADY),
                                    .AXI_ARADDR(AXI_ARADDR),
                                    .AXI_ARID(AXI_ARID),
                                    .AXI_ARLEN(AXI_ARLEN),
                                    .AXI_ARSIZE(AXI_ARSIZE),
                                    .AXI_ARPROT(AXI_ARPROT),
                                    .AXI_ARBURST(AXI_ARBURST),
                                    .AXI_ARLOCK(AXI_ARLOCK),
                                    .AXI_ARCACHE(AXI_ARCACHE),
                                    .AXI_ARVALID(AXI_ARVALID),
                                    .AXI_ARREADY(AXI_ARREADY),
                                    .AXI_RDATA(AXI_RDATA),
                                    .AXI_RID(AXI_RID),
                                    .AXI_RRESP(AXI_RRESP),
                                    .AXI_RLAST(AXI_RLAST),
                                    .AXI_RVALID(AXI_RVALID),
                                    .AXI_RREADY(AXI_RREADY),
                                    .ARESETn(reset),
                                    .ACLK(clk_local));
//  ENABLE_MONITOR
monitor_amba_svs
#(.DATA_WIDTH(DATA_WIDTH),.ADDR_WIDTH(ADDR_WIDTH),.ID_WIDTH(4))
axi_perf_monitor  (
                                    .AXI_WDATA(AXI_WDATA),
                                    .AXI_WID(AXI_WID),
                                    .AXI_WSTRB(AXI_WSTRB),
                                    .AXI_WLAST(AXI_WLAST),
                                    .AXI_WVALID(AXI_WVALID),
                                    .AXI_WREADY(AXI_WREADY),
                                    .AXI_AWADDR(AXI_AWADDR),
                                    .AXI_AWID(AXI_AWID),
                                    .AXI_AWLEN({4'd0,AXI_AWLEN}),
                                    .AXI_AWSIZE(AXI_AWSIZE),
                                    .AXI_AWPROT(AXI_AWPROT),
                                        .AXI_AWBURST(AXI_AWBURST),
                                        .AXI_AWLOCK(AXI_AWLOCK),
                                        .AXI_AWCACHE(AXI_AWCACHE),
                                        .AXI_AWVALID(AXI_AWVALID),
                                        .AXI_AWREADY(AXI_AWREADY),
                                        .AXI_BID(AXI_BID),
                                        .AXI_BRESP(AXI_BRESP),
                                        .AXI_BVALID(AXI_BVALID),
                                        .AXI_BREADY(AXI_BREADY),
                                        .AXI_ARADDR(AXI_ARADDR),
                                        .AXI_ARID(AXI_ARID),
                                        .AXI_ARLEN({4'd0,AXI_ARLEN}),
                                        .AXI_ARSIZE(AXI_ARSIZE),
                                        .AXI_ARPROT(AXI_ARPROT),
                                        .AXI_ARBURST(AXI_ARBURST),
                                        .AXI_ARLOCK(AXI_ARLOCK),
                                        .AXI_ARCACHE(AXI_ARCACHE),
                                        .AXI_ARVALID(AXI_ARVALID),
                                        .AXI_ARREADY(AXI_ARREADY),
                                        .AXI_RDATA(AXI_RDATA),
                                        .AXI_RID(AXI_RID),
                                        .AXI_RRESP(AXI_RRESP),
                                        .AXI_RLAST(AXI_RLAST),
                                        .AXI_RVALID(AXI_RVALID),
                                        .AXI_RREADY(AXI_RREADY),
                                        // AXI4 signals
                                        .AXI_AWQOS(4'd0),
                                        .AXI_ARQOS(4'd0),
                                        .AXI_AWREGION(4'd0),
                                        .AXI_ARREGION(4'd0),
                                        .AXI_AWUSER(32'd0),
                                        .AXI_ARUSER(32'd0),
                                        .AXI_WUSER(32'd0),
                                .AXI_RUSER(32'd0),
                                .AXI_BUSER(32'd0),
                                // ACELite signals
                                .AXI_AWSNOOP(3'd0),
                                .AXI_ARSNOOP(4'd0),
                                .AXI_AWDOMAIN(2'd0),
                                .AXI_ARDOMAIN(2'd0),
                                .AXI_AWBAR(2'd0),
                                .AXI_ARBAR(2'd0),
                                .ARESETn(reset),
                                .ACLK(clk_local)

                            );

// END


`ifndef COEMU
  `ifdef ENABLE_PA
    initial $dumpvars();
  `endif
`endif

`ifndef ENABLE_PA
   `ifndef COEMU
      initial
      begin
          $fsdbDumpvars;
          $fsdbDumpon;
       end
   `endif

   `ifdef SEM_XTOR
```

```
       initial
       begin
           $fsdbDumpvars;
           $fsdbDumpon;
       end
   `endif
`endif

`ifdef COEMU
    ////for FWC generation
    //initial begin: dump_hw_top
    //     (* fwc *) $dumpvars  (0, hw_top);//Dump all ports of an instance
    //end
`endif

endmodule
```

Software testbench:

```
#include <string.h>
#include "xtor_amba_master_svs.hh"
#include <queue>
#include "stdio.h"
#include "tb_common.hh"

#ifndef XTOR_SCOPE
#define XTOR_SCOPE "hw_top.axi_perf_monitor"
#endif // XTOR_SCOPE

using namespace ZEBU_IP;
using namespace MONITOR_AMBA_SVS;
```

```
extern char* __progname;

class trivial_test
{
    public:
    int main_phase()
    {
          svt_c_runtime_cfg* runtime =
svt_c_runtime_cfg::get_default();
          int _argc = 1;
          char** _argV = (char**)malloc(_argc* sizeof(char*));
          _argV[0]= __progname;

// ENABLE_MONITOR
          monitor_amba_svs* m1=new
monitor_amba_svs(5,XTOR_SCOPE,AXI3,FSDB_LOG,runtime);
          monitor_amba_svs::init(_argc, _argV,"pa_axi.fsdb");
          m1->enable(1);
// END



        // Create instance of AMBA master
        xtor_amba_master_svs *objAmba =
xtor_amba_master_svs::getInstance("hw_top.master_node_i",
runtime);

        // Start Test
        if(objAmba != NULL)
        {
            //register the callbacks
            objAmba->registerBrespCB(&cb_wresp);
            objAmba->registerRrespCB(&cb_rresp);

            objAmba->enableTxnDump(true);
            objAmba->setDebugLevel(0);
            objAmba->runUntilReset();
            objAmba->runClk(10);

            AXIResp   wr_resp, rd_resp;
            enAXIResp *wrresp, *rdresp;
            AXIData   *axi_rd_data = new AXIData;

            uint32_t data_width_in_bits  = 256;
            int max_num_txn              = 25 ;

            for(int j = 0; j < max_num_txn ; j++)
            {
               printf("TEST :: Starting transaction number %d \n",
j);
              AXIChanInfo axi_addr;
              enAXIResp   axi_resp;
              AXIData     *axi_wr_data;

              axi_addr.id       = rand()%16;
              axi_addr.len      = rand()%16;
              axi_addr.size     = 0x5 - (rand()%6);
              axi_addr.addr     = 0x0 + ((1 <<
axi_addr.size)*(axi_addr.len + 1))*j;
              axi_addr.prot     = 0x0;
              axi_addr.lock     = 0x0;
              axi_addr.cache    = 0x0;
              axi_addr.burst    = AXI_BURST_INCR;
```

```
                      // out of 4kb boundary check
                     uint64_t num_bytes    = ( 1 << axi_addr.size);
                      uint64_t aligned_addr = (axi_addr.addr/
num_bytes) * num_bytes;
                    uint64_t boundary     = (aligned_addr & ~4095)
+ 4096;
                      uint64_t total_bytes  = (axi_addr.len +1
)*num_bytes;
                      if((aligned_addr + total_bytes) > boundary)
                      {
                          std::cout<<"\n A burst must not cross a
4KB address boundary. \n"; fflush(stdout);
                          continue;
                      }

                uint32_t total_num_bytes     = (axi_addr.len + 1)*
(1 << axi_addr.size);
                axi_wr_data                  = new AXIData();
                axiBE_t byte_en              = 0xFF;

                for(int i = 0; i < total_num_bytes; i++)
                {
                    uint8_t data       = i + j*total_num_bytes;
                    axiBE_t byte_en    = 0xFF;
                  axi_wr_data->FillByteArray( &data, 1, &byte_en);
                }

                objAmba->wrTxn(axi_addr, axi_wr_data, &wr_resp);
                wrresp = wr_resp.GetResp();
                if(wrresp!=NULL)
```

```
                   std::cout << "TEST :: OKAY: wresp received resp
= " <<wrresp[0] << " for id = " << axi_addr.id << "\n";
                else
                  std::cout << "TEST :: ERROR: wresp received resp
= " << wrresp[0] << " for id = " << axi_addr.id << "\n";


                objAmba->runClk(100);
                objAmba->rdTxn(axi_addr, axi_rd_data, &rd_resp);
                rdresp = rd_resp.GetResp();
                if(rdresp!=NULL)
                   std::cout << "TEST :: OKAY: wresp received resp
= " << rdresp[0] << " for id = " << axi_addr.id << "\n";
                else
                  std::cout << "TEST :: ERROR: wresp received resp
= " << rdresp[0] << " for id = " << axi_addr.id << "\n";


                // Compare Data sent and received
                if( *axi_rd_data != *axi_wr_data)
                  std::cout << "TEST :: ERROR: rresp received with
read data not matching with the write data for id = "<< axi_addr.id
<< "\n";
                else
                   std::cout << "TEST :: OKAY: rresp received with
read data matching with the write data for id = "<<axi_addr.id <<
"\n";


               printf("TEST :: Sent transaction number %d \n", j);
            }
            objAmba->runClk(20);
// ENABLE_MONITOR
            m1->enable(0);
            sleep(3);
            m1->close();
            sleep(3);
            TEST_END = true;
            objAmba->runClk(100);
// END
            return 0;
        }
        else
        {
            printf("TEST :: ERROR: Could not create valid Amba 
Master instance \n");
            return -1;
        }
    }
};
```