**AXI Protocol Verification Environment Specification**

Table of Contents

[Document history 2](#_Toc109746558)

[1.Introduction 4](#_Toc109746559)

[1.1 Scope 4](#_Toc109746560)

[1.2 Abbreviations 4](#_Toc109746561)

[1.3 References 4](#_Toc109746562)

[2. AXI Protocol Overview 5](#_Toc109746563)

[2.1. AXI Protocol Features 5](#_Toc109746564)

[2.2. AXI Protocol Interfaces 5](#_Toc109746565)

[3. Verification Environment 5](#_Toc109746566)

[3.1 Environment Overview 5](#_Toc109746567)

[3.2 Components description 6](#_Toc109746568)

[3.4 Checkers 6](#_Toc109746569)

[3.5 Coverage 6](#_Toc109746570)

[3.6 Tests 6](#_Toc109746571)

[4. Simulation flow 6](#_Toc109746572)

[5. How to run a test 7](#_Toc109746573)

[6. Limitations and assumptions 7](#_Toc109746574)

# Document history

| Version | Date | Approved by | Created by | Description |
| --- | --- | --- | --- | --- |
| 0.1 | 14.02.2023 | Liviu Ababei | Ioana Ailenei Andreea Haldan | Document created |
| 1.0 | {Date} | {Trainer} | {Trainee} | Revised by {Name} |

# 1.Introduction

## 1.1 Scope

This document describes the verification environment for the AXI Protocol.

## 1.2 Abbreviations

**DUT =>** Device Under Test

**AXI**  **=>** Advanced eXtensible Interface

## 1.3 References

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Name** | **Revision** | **Description** |
| 1. | AXI Metric plan.xlsx | 1.0 | Coverage, Checkers and Test plans |
| 2. | 1. IHI0022E\_amba\_axi\_and\_ace\_protocol\_spec.pdf | x.x | AXI Protocol specification |

# 2. AXI Overview

# 

1. Figure 1. The MASTER/SLAVE AXI input/output interfaces & architecture diagram

## 2.1. AXI Features

The AMBA AXI protocol supports high-performance, high-frequency system designs.

The AXI protocol:

* is suitable for high-bandwidth and low-latency designs
* provides high-frequency operation without using complex bridges
* meets the interface requirements of a wide range of components
* is suitable for memory controllers with high initial access latency
* provides flexibility in the implementation of interconnect architectures
* is backward-compatible with existing AHB and APB interfaces.

The key features of the AXI protocol are:

* separate address/control and data phases
* support for unaligned data transfers, using byte strobes
* uses burst-based transactions with only the start address issued
* separate read and write data channels, that can provide low-cost Direct Memory Access (DMA)
* support for issuing multiple outstanding addresses
* support for out-of-order transaction completion
* permits easy addition of register stages to provide timing closure.

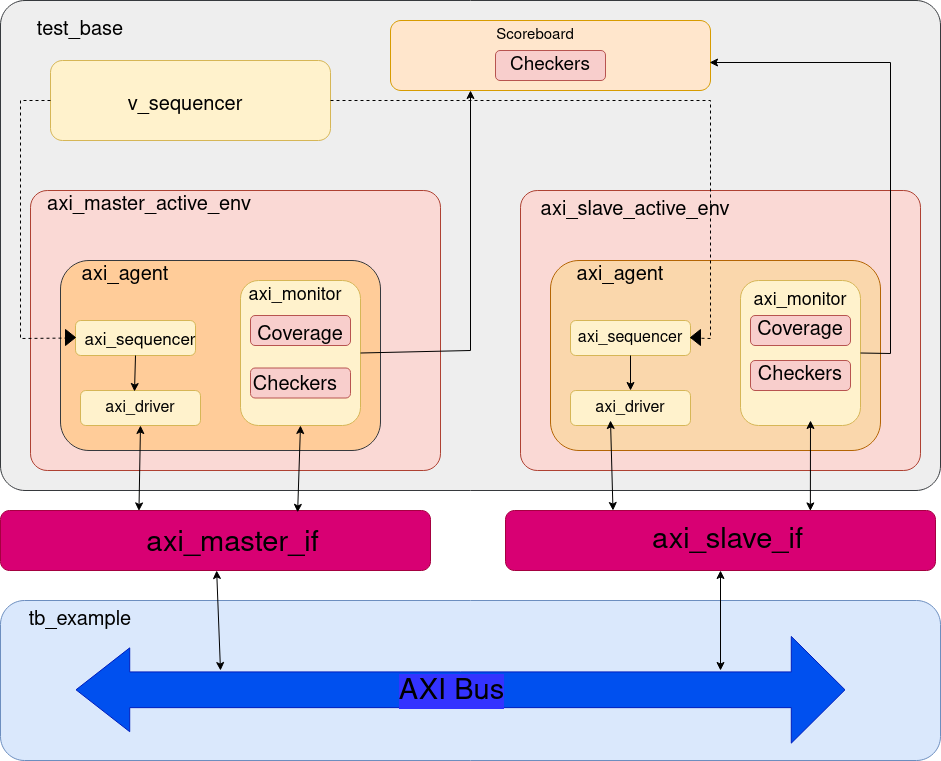
## 2.2. AXI Interfaces

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 1. Signal name | 1. Direction | 1. Width | 1. Description |
| 1. global | 1. aclk | 1. I | 1. 1 | 1. Clock |
| 1. aresetn | 1. I | 1. 1 | 1. HW reset, active low |
| 1. write address channel | 1. awvalid | 1. O | 1. 1 | 1. Write request |
| 1. awid | 1. O | 1. 4 | 1. Write ID. Between 0 and 15 |
| 1. awaddr | 1. O | 1. 32 | 1. Write address. Access is: Byte ([1:0] can take any value) , Halfword ([1:0] can be 0 or 2) or Word alligned ([2:0] is always 0) |
| 1. awlen | 1. O | 1. 4 | 1. Burst length (no of cycles in burst - 1) |
| 1. awsize | 1. O | 1. 3 | 1. Byte, Halfword, Word. Byte and Halfword is supported only if burst length is 1. |
| 1. awburst | 1. O | 1. 2 | 1. Burst type: FIXED, INCR, WRAP |
| 1. awready | 1. I | 1. 1 | 1. Write request acknowledge |
| 1. write data channel | 1. wvalid | 1. O | 1. 1 | 1. Write data valid |
| 1. wdata | 1. O | 1. 32 | 1. Write data |
| 1. wstrb | 1. O | 1. 4 | 1. Write strobe. For bursts (more than 1 cycle) is all ones. For bursts of 1 cycle, it can be between 1 and 15. If size is not Word, make sure the bits set to 1 are matching the selected Bytes. |
| 1. wlast | 1. O | 1. 1 | 1. Write data last |
| 1. wready | 1. I | 1. 1 | 1. Write data ready |
| 1. write  response channel | 1. bvalid | 1. I | 1. 1 | 1. Write response valid |
| 1. bid | 1. I | 1. 4 | 1. Write response ID. |
| 1. bready | 1. O | 1. 1 | 1. Write response ready - stuck to 1 |
| 1. read address channel | 1. arvalid | 1. O | 1. 1 | 1. Read request |
| 1. arid | 1. O | 1. 4 | 1. Read ID. Between 0 and 15 |
| 1. araddr | 1. O | 1. 32 | 1. Read address. Access is: Byte ([1:0] can take any value) , Halfword ([1:0] can be 0 or 2) or Word alligned ([2:0] is always 0) |
| 1. arlen | 1. O | 1. 4 | 1. Burst length (no of cycles in burst - 1) |
| 1. arsize | 1. O | 1. 3 | 1. Byte, Halfword, Word. Byte and Halfword is supported only if burst length is 1. |
| 1. arburst | 1. O | 1. 2 | 1. Burst type: FIXED, INCR, WRAP |
| 1. arready | 1. I | 1. 1 | 1. Read request acknowledge |
| 1. read data  channel | 1. rvalid | 1. I | 1. 1 | 1. Read data valid |
| 1. rdata | 1. I | 1. 32 | 1. Read data |
| 1. rlast | 1. I | 1. 1 | 1. Read data last |
| 1. rid | 1. I | 1. 4 | 1. Read response ID. |
| 1. rresp | 1. I | 1. 2 | 1. Read response status: all posibilities supported |
| 1. rready | 1. O | 1. 1 | 1. Read data ready. |

# Verification Environment

## 3.1 Environment Overview

* 1. The AXI Protocol verification environment diagram is presented in Figure 2.

1. Figure 2. The AXI Protocol Verification Environment Diagram

## 3.2 Components description

**Sequence**: series of transactions

**Sequencer**: It generates sequences of transaction for the driver with different constraints based on the test sequences.

**Driver**: Gets a transaction from Sequencer via a TLM port and puts it on AXI interface following the AXI standard protocol.

**Monitor**: It monitors the AXI interface, collects data from the bus and encapsulates in an item to be sent to the higher level via an analysis port for further checking. Also, it collects coverage information.

**Agent**: The AXI agent instantiates the driver, monitor and sequencer into a single entity and connects the components via TLM interfaces. AXI agent is configured as master and slave agents. The number of AXI agents in the system are two..

## 3.4 Checkers

The checkers list can be found in AXI Protocol metric driven plan.xlsx, the “Checkers” sheet.

## 3.5 Coverage

The functional coverage description can be found in AXI Protocol metric driven plan.xlsx, the “Coverage” sheet.

## 3.6 Tests

The tests’ description can be found in AXI Protocol metric driven plan.xlsx, the “Tests” sheet.

# 4. Simulation flow

1. Reset the AXI Bus. Generate reset and clock in top, then drive the signals in idle state to interface

2. Start a Master Sequence and a Slave Sequence in a fork.

3. Master Sequence Send Packet: Generates the random transfer packet, send the packet to driver, then drive the signals to the interface accordingly with the protocol specifications. Monitors the interface forever and forms packets to send to the higher level, checks and collect coverage.

4. Slave Sequence Send Packet: If Active: Generates the random transfer packet, send the packet to driver, then drive the signals to the interface accordingly with the protocol specifications. Monitors the interface forever and forms packets to send to the higher level, checks and collect coverage. If Reactive: looks on interface forever and put random data on interface when is needed to. Also monitors.

5. Done?: If we have more sequences to run then step 2, 3.active will repeat, if not, we finish the test. 3.reactive will stop when the master will stop.

# 5. How to run a test

[Script to use]

1. ../example/run.ius

[From where to run]

1. ../example

[Example of command line parameters]

1. run\_ius test\_example\_1

# 6. Limitations and assumptions

[What doen’t work]

reset on fly

[What assumptions you took on DUT functionality]

[What is still to be verified]

[What is not supported in the VE from the DUT functionality]