# AXI5 Assertion IP User Guide

Version 2023.03, March 2023

SYNOPSYS®

# Contents

# Figures

# Tables

# Preface

This guide contains installation, setup, and usage instructions for the AMBA AXI5 SystemVerilog AIP. It is intended for use by the design or verification engineers who want to verify RTL designs with an AMBA AXI5 interface. Readers are assumed to be familiar with the AMBA AXI5 Protocol, SystemVerilog Assertions and the Verilog language.

This chapter includes the following sections:

- Guide Organization
- Web Resources
- Customer Support

## Guide Organization

The chapters of this guide are organized as follows:

Chapter 1,“Introduction”, introduces the Synopsys AXI5 AIP and its features.

Chapter 2, “Installation and Setup”, describes system requirements and provides instructions on how to install, configure, and begin using the Synopsys AXI5 AIP.

Chapter 3,“Using AXI5 AIP in a Formal Environment”, introduces the VC Formal tool and the AXI5 AIP usage in a formal environment.

Chapter 4, “The AXI5 AIP Configuration”, presents an insight into the functionality of the AXI5 AIP.

Chapter 5, “The AXI5 AIP Use Cases”, presents an example of how to install and use the Synopsys AXI5 AIP.

## Web Resources

The AXI5 AIP is compliant with the following specifications:

- AMBA specification ARM IHI 0022F.b (ID122117)
- AMBA specification ARM IHI 0022H.c (ID12621)
- AMBA FAQ document

  http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.set.amba/index.html

## Customer Support

To obtain support for your product, choose one of the following:

- Open a Case Through SolvNet:

  Go to http://onlinecase.synopsys.com/Support/OpenCase.aspx

  - Provide the requested information, including:
    - **Product L1**: VC Formal
    - **Product L2**: AIP
    - **Product L2**: AXI5 AIP
    - **Product Version**: 2020.03
    - Fill in the remaining fields according to your environment and issue.
- Send an e-mail message to support_center@synopsys.com
    - Include product name, sub-product name, and product version (as noted above) in your e-mail, so it can be routed correctly.
- Telephone your local support center:
    - North America:

      Call 1-800-245-8005 from 7 AM to 5:30 PM Pacific time, Monday through Friday
    - All other countries:

      http://www.synopsys.com/Support/GlobalSupportCenters

# 1 Introduction

This document describes the AXI5 protocol checkers available in the AXI5 Assertion IP (AIP). It also describes how to configure the AIP, including all parameters related to the configuration, how to instantiate the AIP, and so on.

Assertion IP is significant in reducing the verification effort and improving the design quality. Its value comes from the fact that the assertions can passively monitor the design behavior by simply attaching them to the target design, without modifying the RTL. Pre-built assertions from assertion IP provide a powerful quality criteria for sign-off.

This chapter consists of the following sections:

- Prerequisites
- References
- Product Overview
- Language and Methodology Support
- Feature Support
- Features Not Supported

**Note**
Based on the AMBA Progressive Terminology updates, you must interpret the term Master as Manager and Slave as Subordinate in the AIP documentation and messages.

## 1.1 Prerequisites

Familiarity with the AXI5 protocol, SystemVerilog Assertions and the SystemVerilog language.

## 1.2 References

AXI5 AIP is compliant with the following specifications:

- AMBA specification ARM IHI 0022F.b (ID122117)
- AMBA FAQ document

  http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.set.amba/index.html

## 1.3 Product Overview

The AXI5 AIP is a suite of SystemVerilog Assertions and related SystemVerilog code that are meant to be used with RTL designs. This AIP is used to verify the RTL with the VC Formal tool.

The AXI5 AIP consists of the following:

- Assertions properties
- Assume properties
- Cover properties
- Synthesizable modeling SystemVerilog for properties

## 1.4 Language and Methodology Support

In the current release, the AXI5 AIP suite supports the following languages and methodology:

- SystemVerilog Assertion
- Verilog

## 1.5 Feature Support

### 1.5.1 Protocol Features

The AXI5 AIP currently supports the following protocol functions:

- All data widths
- All address widths
- All transfer types
- All burst types and burst lengths
- All protection types
- All slave response types
- Exclusive transactions

### 1.5.2 Verification Features

The AXI5 AIP currently supports the following verification features:

- Ability to configure as Master
- Ability to configure as Slave
- Ability to configure as Monitor
- Ability to configure as Constraint Provider
- Protocol Checks
- Selective inclusion or exclusion of multiple features like CONFIG_X_CHECK, WR_OUT_OF_ORDER, RD_INTERLEAVE, ENABLE_ASSERT, ENABLE_ASSUME and ENABLE_COVER. Refer Tables 4-1 for more details.

## 1.6 Features Not Supported

- Atomic Transactions

- Atomic Transactions support is limited and not fully supported.

# 2 Installation and Setup

This chapter guides you through installing and setting up the AXI5 AIP. When you complete the checklist mentioned below, the provided example gets operational and you can use the AXI5 AIP.

The checklist consists of the following major steps:

- "Verifying Hardware Requirements" on page 15
- "Verifying Software Requirements" on page 15
- "Preparing for Installation" on page 17

## 2.1 Verifying Hardware Requirements

The AXI5 AIP suite requires a Linux workstation configured as follows:

- 400 MB available disk space for installation
- 1 GB available swap space
- 1 GB RAM (physical memory) recommended
- FTP anonymous access to ftp.synopsys.com (optional)

## 2.2 Verifying Software Requirements

This section lists software that the AXI5 AIP requires.

- VCS version P-2019.06-SP1 (Simulator)
- Verdi version P-2019.06-SP1 (Debugger)
- VCF version P-2019.06-SP1 (Formal)
- VC version P-2019.06-SP1 (Verification Compiler Platform)

### 2.2.1 Platform/OS and Simulator Software

- VC Formal tool is required

### 2.2.2 Synopsys Common Licensing (SCL) Software

The AXI5 AIP requires the following license feature:

- AIP-AXI5-SVA

The following topics describe the required environment variables and path settings for the AXI5 AIP:

#### 2.2.2.1 Running the AXI5 AIP on the VC Formal Tool

To run the AXI5 AIP on the VC Formal tool, set the following environment variable:

`SNPSLMD_LICENSE_FILE`: The absolute path to file(s) that contains the license keys for Synopsys software (AIP and/or other Synopsys Software tools) or the port@host reference to this file.

Example:

```
setenv SNPSLMD_LICENSE_FILE <port>@<server>:${SNPSLMD_LICENSE_FILE}
```

or

```
setenv SNPSLMD_LICENSE_FILE <full path to the license file>:${SNPSLMD_LICENSE_FILE}
```

#### 2.2.2.2 Running the AXI5 AIP on VCS

To run the AXI5 AIP on VCS, set the following two environment variables:

- `SNPSLMD_LICENSE_FILE`
- `DW_LICENSE_FILE`: The absolute path to file that contains the license keys for the AIP product software or the `port@host reference` to this file.

Example:

```
setenv SNPSLMD_LICENSE_FILE <port>@<server>:${SNPSLMD_LICENSE_FILE}
setenv DW_LICENSE_FILE <port>@<server>:${DW_LICENSE_FILE}
```

or

```
setenv SNPSLMD_LICENSE_FILE <full path to the license file>:${SNPSLMD_LICENSE_FILE}
setenv DW_LICENSE_FILE <full path to the license file>:${DW_LICENSE_FILE}
```

**Table 2-1 AIP Licensing Key Features**

| Package Name | Feature Keys | Included Titles |
|---|---|---|
| VC Formal AIP AMBA APB | AIP-APB-SVA | APB4, APB3, APB2 and APB |
| VC Formal AIP AMBA AHB | AIP-AHB-SVA | AHB5, AHB and AHB-Lite |
| VC Formal AIP AMBA AXI | AIP-AXI-SVA | AXI5, AXI5-Lite and AXI3 |
| VC Formal AIP AMBA ACE | AIP-ACE-SVA | ACE, ACE-Lite, AXI5, AXI5-Lite and AXI3 |
| VC Formal AIP AMBA5 AXI | AIP-AXI5-SVA | AXI5 and AXI5-Lite |
| VC Formal AIP AMBA5 CHI | AIP-CHI-SVA | CHI B, C, D, and E |

### 2.2.3 Other Third-Party Software

Adobe Acrobat: The AXI5 AIP documentation is available in the Acrobat PDF files. You can get the Adobe Acrobat Reader for free from http://www.adobe.com.

HTML browser: The AXI5 AIP coverage reports can be viewed in HTML. The following browser/platform combinations are supported:

- Microsoft Internet Explorer 6.0 or later (Windows)

- Firefox 1.0 or later (Windows and Linux)
- Netscape 7.x (Windows and Linux)

## 2.3 Preparing for Installation

Ensure that your environment and PATH variables are set correctly. For information on the environment variables and path settings required for the AXI5 AIP, see "Synopsys Common Licensing (SCL) Software" on page 15.

# 3 Using AXI5 AIP in a Formal Environment

This chapter describes the simulation environment for the AXI5 AIP, usage of the VC Formal tool, and the AXI5 AIP usage in a formal verification environment. This chapter discusses the following topics:

## 3.1 Introduction to the VC Formal Tool

The VC Formal tool is used to verify assertion properties by examining all sequences of possible value combinations. These valid inputs are constrained by the assume properties. The VC Formal tool provides several pieces of information as follows:

- Number of proven properties
- Number of falsified properties
- Number of vacuous properties
- Number of covered properties
- Number of witness properties

The VC Formal tool is useful for debugging failing properties by means of its GUI interface. For running the properties in the VC Formal tool, the following information must be provided through the tool's command line interface or through a Tcl script:

- The path of the AXI5 AIP source code
- The path of the RTL source code (if validating the RTL)
- Clock information
- Reset information
- Timeout details

## 3.2 The AXI5 AIP in a Formal Environment

The AXI5 AIP has AXI5 Master properties and AXI5 Slave properties. These properties are connected to either AXI5 Master or Slave module (for example, AXI5 Slave DUT to Master Properties) interface signals, then the formal verification tool, VC Formal, is used to verify the functional correctness of the module.

To use the AXI5 AIP in a formal verification environment, perform the following steps in a sequence:

- Instantiating the AXI5 AIP using the `bind` statement
- Creating a Tcl file
- Reading and running a Tcl file
- Analyzing results

## 3.2.1 Instantiating the AXI5 AIP Using bind Statement

Create the bind file to instantiate the AXI5 AIP and bind the AXI5 AIP to the design. Map the module and port names in the design to those of the AIP in the bind statement. Pass valid values to the configuration parameters of the AIP. The next step in the process is compiling the files.

**Note**
If a signal corresponding to the AXI5 AIP port does not exist in the DUT, set inactive value or the value expected by the DUT for the port. For example, if the DUT does not have the `awlock` signal, set with `1’b0`.

## 3.2.2 Creating a Tcl File

To compile the DUT and the attached AXI5 AIP (including RTL files, AIP files, and the bind file), create a Tcl file, where path to RTL directory (AXI5 AIP directory) and the bind file are set. The DUT clock and reset are initialized with the `create_clock` and `create_reset` commands, as shown in Table 3-1. VC Formal can report assertion status (proven, falsified, vacuous or inclusive using the `report_fv` command). For more command options, see the VC Formal User Guide. In case RTL is AXI5-Lite protocol, analyze ‘snps_axi5_lite_aip.sv’ file. In case RTL is AXI5 protocol, analyze ‘snps_axi5_aip.sv’ file.

**Table 3-1 Tcl File Example**

```
# variables setting
set AIP_HOME
$::env(VC_STATIC_HOME)/packages/aip
set AIP_SRC $AIP_HOME/AXI5_AIP/src
set TBH_DIR ../tb
set TCL_DIR ../tcl

set design axi5_dut

# vcs option
set vcs " \
  +incdir+${AIP_SRC} \
  ${AIP_SRC}/snps_axi_aip_pkg.sv \
  ${AIP_SRC}/snps_axi5_aip.sv \
  ${TBH_DIR}/axi5_dut.v \
  ${TBH_DIR}/bind_axi5_aip.sv \
  -assert svaext "
```

```
# Enable all Formal Debug Modes
set_fml_appmode FPV

# analyze and elaborate design and AIP
read_file -sva -top $design -format sverilog -
vcs "$vcs"

# clock setting
create_clock aclk -period 100

# reset setting
create_reset aresetn -low

sim_run -stable
sim_save_reset

# execute proof
check_fv -run_finish {
  report_fv
}
```

### 3.2.3 Reading and Running a Tcl File

To read and run a Tcl file, use either of the following two modes:

- "GUI Mode" on page 21
- "Reading and Running Tcl File in the Batch Mode" on page 22

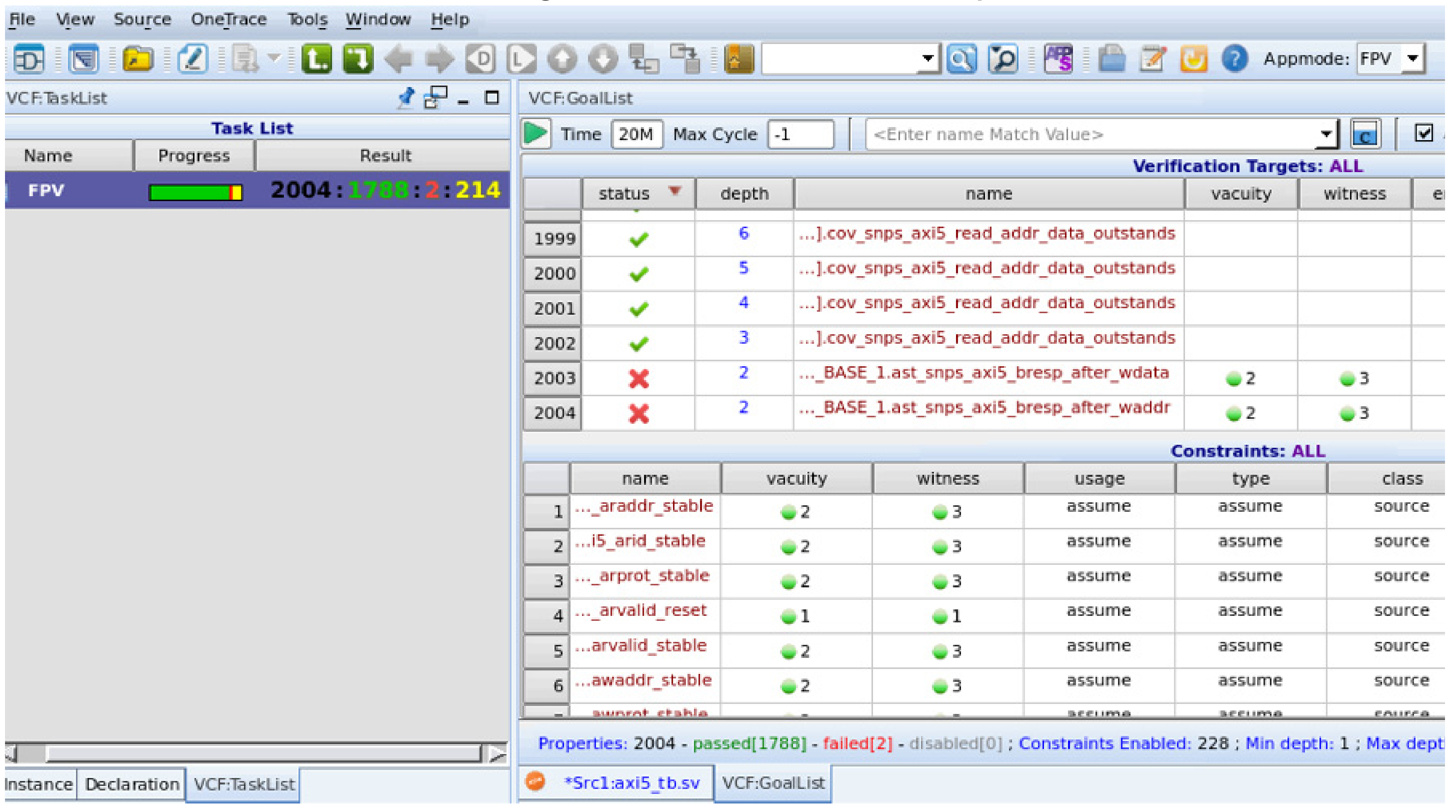
#### 3.2.3.1 GUI Mode

To read a VC formal Tcl file, invoke the VC Formal tool in the GUI mode and perform the following steps:

1. Navigate to the folder containing the Tcl file.
2. Open VC Formal in the GUI mode using the `vcf -gui -f <tcl file>` command.

After all the properties are executed, Verdi tool displays the list of properties as shown in Figure 3-1, where the red cross Indicates a falsified property and the green tick marks Indicate proven properties.

**Figure 3-1 Falsified and Proven Properties**


File View Source OneTrace Tools Window Help
Appmode: FPV
VCF:TaskList
VCF:GoalList
Time 20M Max Cycle -1
<Enter name Match Value>
Instance Declaration VCF:TaskList
*Src1:axi5_tb.sv VCF:GoalList

Task List

| Name | Progress | Result |
|---|---|---|
| FPV | 🟩 | 2004:1788:2:214 |

Verification Targets: ALL

| | status | depth | name | vacuity | witness |
|---|---|---|---|---|---|
| 1999 | ✓ | 6 | ...].cov_snps_axi5_read_addr_data_outstands | | |
| 2000 | ✓ | 5 | ...].cov_snps_axi5_read_addr_data_outstands | | |
| 2001 | ✓ | 4 | ...].cov_snps_axi5_read_addr_data_outstands | | |
| 2002 | ✓ | 3 | ...].cov_snps_axi5_read_addr_data_outstands | | |
| 2003 | ✗ | 2 | ..._BASE_1.ast_snps_axi5_bresp_after_wdata | ● 2 | ● 3 |
| 2004 | ✗ | 2 | ..._BASE_1.ast_snps_axi5_bresp_after_waddr | ● 2 | ● 3 |

Constraints: ALL

| | name | vacuity | witness | usage | type | class |
|---|---|---|---|---|---|---|
| 1 | ..._araddr_stable | ● 2 | ● 3 | assume | assume | source |
| 2 | ...i5_arid_stable | ● 2 | ● 3 | assume | assume | source |
| 3 | ..._arprot_stable | ● 2 | ● 3 | assume | assume | source |
| 4 | ..._arvalid_reset | ● 1 | ● 1 | assume | assume | source |
| 5 | ...arvalid_stable | ● 2 | ● 3 | assume | assume | source |
| 6 | ...awaddr_stable | ● 2 | ● 3 | assume | assume | source |

Properties: 2004 - passed[1788] - failed[2] - disabled[0] ; Constraints Enabled: 228 ; Min depth: 1 ; Max dept

##### 3.2.3.1.1 Analyzing Results for the GUI Mode Run

After running a session, its results are dumped into the `vcf.log` file. This log file gives the number of proven, falsified, vacuous, inconclusive, and covered properties. When there are no falsifications, the design is verified with regard to the parameters configured in a Tcl file.

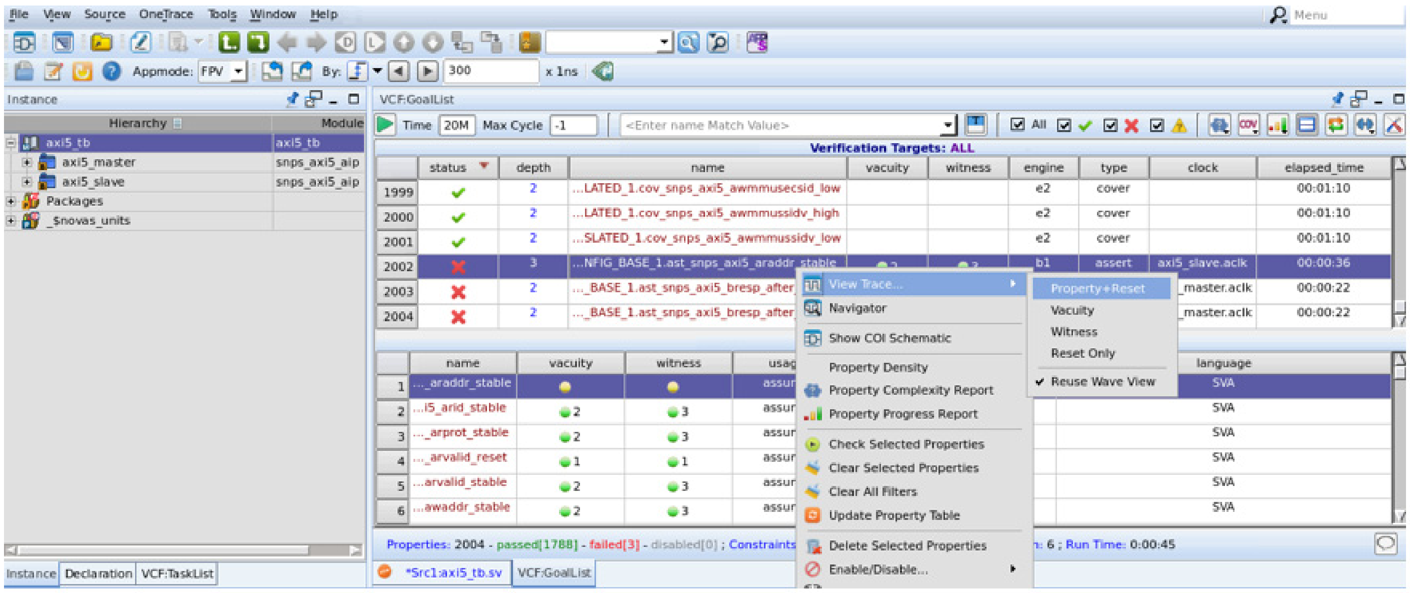
If properties are falsified, you should debug them to find the root cause. Perform the following steps to debug the falsification:

1. Right-click on any of the failures which you need to be debugged to view the options, such as View Trace, Explore the Property, Report, and so on.

2. When you select the View Trace option, waveforms are opened, providing signal details and falsification depth. You can dump other signals required for debugging into the waveform as well.

You can also explore the options in the VC Formal tool and debug the failure. See the VC Formal User Guide for more information on options. Figure 3-2 and Figure 3-3 shows options to debug properties and signal behavior in the waveforms respectively.

**Figure 3-2 Options to Debug Properties**



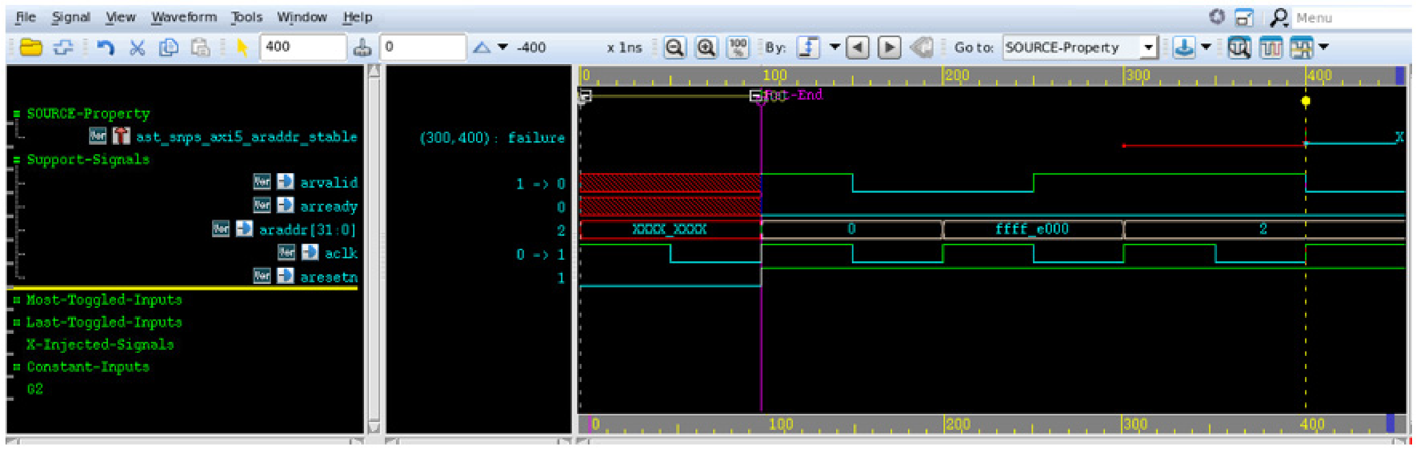
| Hierarchy | Module |
|---|---|
| axi5_tb | axi5_tb |
| axi5_master | snps_axi5_aip |
| axi5_slave | snps_axi5_aip |
| Packages | |
| _$novas_units | |

| | status | depth | name | vacuity | witness | engine | type | clock | elapsed_time |
|---|---|---|---|---|---|---|---|---|---|
| 1999 | ✓ | 2 | ...LATED_1.cov_snps_axi5_awmmusecsid_low | | | e2 | cover | | 00:01:10 |
| 2000 | ✓ | 2 | ...LATED_1.cov_snps_axi5_awmmussidv_high | | | e2 | cover | | 00:01:10 |
| 2001 | ✓ | 2 | ...SLATED_1.cov_snps_axi5_awmmussidv_low | | | e2 | cover | | 00:01:10 |
| 2002 | ✗ | 3 | ...NFIG_BASE_1.ast_snps_axi5_araddr_stable | ● | ● | b1 | assert | axi5_slave.aclk | 00:00:36 |
| 2003 | ✗ | 2 | ..._BASE_1.ast_snps_axi5_bresp_after | | | | | _master.aclk | 00:00:22 |
| 2004 | ✗ | 2 | ..._BASE_1.ast_snps_axi5_bresp_after | | | | | _master.aclk | 00:00:22 |

| | name | vacuity | witness | usag | language |
|---|---|---|---|---|---|
| 1 | ..._araddr_stable | ● | ● | assur | SVA |
| 2 | ...i5_arid_stable | ● 2 | ● 3 | assur | SVA |
| 3 | ..._arprot_stable | ● 2 | ● 3 | assur | SVA |
| 4 | ..._arvalid_reset | ● 1 | ● 1 | assur | SVA |
| 5 | ...arvalid_stable | ● 2 | ● 3 | assur | SVA |
| 6 | ...awaddr_stable | ● 2 | ● 3 | assur | SVA |

Properties: 2004 - passed[1788] - failed[3] - disabled[0] ; Constraints ... : 6 ; Run Time: 0:00:45

**Figure 3-3 Signal Behavior in Waves**



### 3.2.3.2 Reading and Running Tcl File in the Batch Mode

To read a VC Formal Tcl file using the command line, perform the following steps:

1. Check whether VC Static is installed and exists in PATH. For this, use the following command:

   ```
   % which vcf
   ```

   If the command gives the 'Command not found' error, install the VC Static tool.

2. Run a Tcl file using the following command:

```
% vcf  -f  <tcl file name>  -full64
```

For more information on the VC formal command line options, see the VC Formal User Guide.

Once the Tcl file is read, the tool runs a formal session and give results, such as proven or falsified for various properties.

#### 3.2.3.2.1 Analyzing Results for the Batch Mode Run

After running a session, results are dumped into a log file. This log file gives the number of proven, falsified, vacuous, inconclusive, and covered properties. If there are no falsifications, a design is qualified with regard to the parameters configured in a Tcl file.

If properties are falsified, you should debug them to find the root cause. Perform the following steps to debug the falsification

On the VC Formal window, execute the following commands:

1. `get_props -status falsified`

   To display the number of falsified properties.

2. `view_trace -property <property name with path of property shown in get_props command>`

   To open the Verdi tool and to display the waves of a falsified property which you want to debug.

### 3.2.4 Commonly Used AXI5 AIP Configurations

**Table 3-2 Common Usage Models**

| | | |
|---|---|---|
| 1 | Master | Instantiates the AXI5 AIP as a master to check the output behavior of a DUT slave and constraint a slave input. |
| | | AGENT_TYPE=MASTER |
| | | By default, ENABLE_ASSERT=1 and ENABLE_ASSUME=1 |
| 2 | Slave | Instantiates the AXI5 AIP as a slave to check the output behavior of a DUT master and constraint a master input. |
| | | AGENT_TYPE=SLAVE |
| | | By default, ENABLE_ASSERT=1 and ENABLE_ASSUME=1 |
| 3 | Monitor | Instantiates the AXI5 AIP as a monitor to check the behavior of a DUT master and slave. |
| | | AGENT_TYPE=MONITOR |
| | | By default, ENABLE_ASSERT=1 and ENABLE_ASSUME=0 |
| 4 | Constraint | Instantiates the AXI5 AIP to constraint the inputs of a DUT master and slave. |
| | | AGENT_TYPE=CONSTRAINT |
| | | By default, ENABLE_ASSERT=0 and ENABLE_ASSUME=1 |

### 3.2.5 The AXI5 AIP As a Master

To verify an AXI5 slave DUT, set the `AGENT_TYPE` parameter to `MASTER` during an AIP instantiation.

When the AXI5 AIP parameter is set as `MASTER`, all the AXI5 AIP properties that are related to the master inputs are declared as assertions, and all the AXI5 AIP properties that are related to the master outputs are declared as assumptions. This is required to make the AXI5 AIP AIP behave as a master.

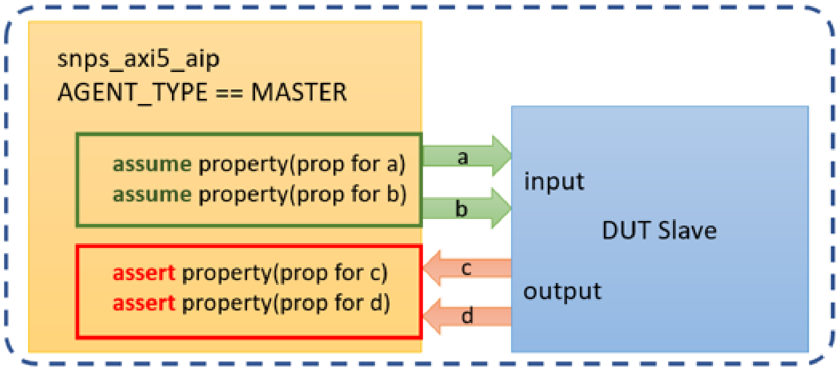
To disable all assert, assume, or cover properties, explicitly set the following parameters to value 0:

- `ENABLE_ASSERT`
- `ENABLE_ASSUME`
- `ENABLE_COVER`

For example, if you want to enable AXI5 slave checks (assert) only and do not want to apply constraints on AXI5 slave inputs (assume), set `ENABLE_ASSERT=1` and `ENABLE_ASSUME=0`. This enables all assert properties related to an AXI5 slave DUT, but disables all assume properties.

Figure 3-4 checks the behavior of an AXI5 slave DUT output and constraints the inputs of an AXI5 slave DUT to valid values.

**Figure 3-4 AXI5 AIP As a Master**


snps_axi5_aip
AGENT_TYPE == MASTER
assume property(prop for a)
assume property(prop for b)
assert property(prop for c)
assert property(prop for d)
a
b
c
d
input
DUT Slave
output

### 3.2.6 The AXI5 AIP As a Slave

To verify an AXI5 master DUT, set the `AGENT_TYPE` parameter to `SLAVE` during the AXI5 AIP instantiation. When this parameter is set as `SLAVE`, all the AXI5 AIP properties that are related to the slave inputs are declared as assertions, and all the AXI5 AIP properties that are related to the slave outputs are declared as assumptions. This is required to make the AXI5 AIP behave as a slave.

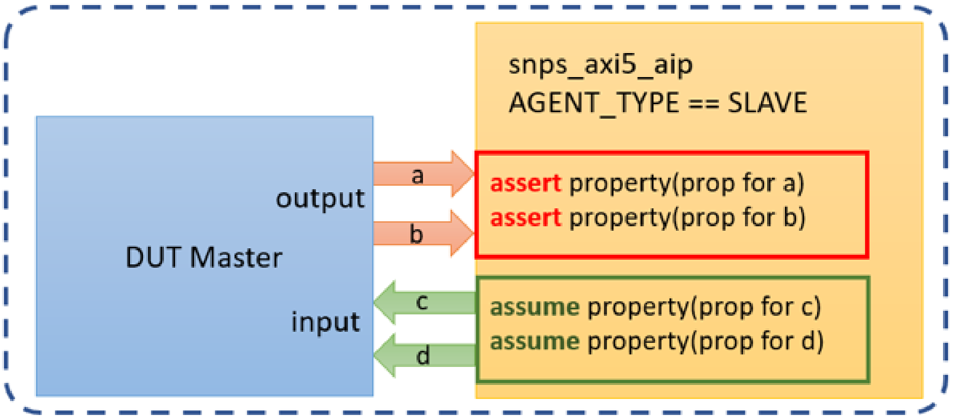
To disable all assert, assume, or cover properties, explicitly set the following parameters to value 0:

- `ENABLE_ASSERT`
- `ENABLE_ASSUME`
- `ENABLE_COVER`

For example, if you want to enable the AXI5 slave checks (assert) only and do not want to apply constraints on the AXI5 master inputs (assume), set `ENABLE_ASSERT=1` and `ENABLE_ASSUME=0`. This enables all assert properties related to an AXI5 master DUT, but disables all assume properties.

Figure 3-5 checks the behavior of an AXI5 master DUT output and constraints the inputs of the AXI5 Master DUT to valid values.

**Figure 3-5 AXI5 AIP As a Slave**


snps_axi5_aip
AGENT_TYPE == SLAVE
output
DUT Master
input
a
b
c
d
assert property(prop for a)
assert property(prop for b)
assume property(prop for c)
assume property(prop for d)

### 3.2.7 The AXI5 AIP As a Monitor

To verify the behavior of an AXI5 master and slave DUT, set the `AGENT_TYPE` parameter to `MONITOR` during the AIP instantiation. When this parameter is set to `MONITOR`, AXI5 AIP is instantiated as a monitor to check the behavior of both the inputs and outputs of the DUT slave and DUT master.

By default, `ENABLE_ASSERT=1` and `ENABLE_ASSUME=0`, this will disable all the assume properties.

**AXI5 AIP Use Case As A Monitor**

In an RTL verification environment, the AXI5 AIP can be instantiated as a monitor on each AXI5 interface to check for protocol correctness.

### 3.2.8 The AXI5 AIP As a Constraint Model

If the `AGENT_TYPE` parameter is set to `CONSTRAINT`, the AXI5 AIP is instantiated to constraint the DUT slave input and DUT master input.

By default, `ENABLE_ASSERT=0` and `ENABLE_ASSUME=1`.

**AXI5 AIP Use Case As A Constraint**

To verify the RTL in a formal verification environment, the AXI5 AIP can be used in the constraint mode to generate stimulus to the RTL.

## 3.3 Clock and Reset Functionality

- To run the AXI5 AIP and a design in the VC Formal tool, create clock using the following command:

  ```
  create_clock <design_clk> -period <time_period>
  ```

  This command specifies clock period.
- To run the AXI5 AIP and a design in the VC Formal tool, create reset using the following command:

  ```
  create_reset<design_reset> -low/high
  ```

  The reset can be active low or active high depending on the type of reset in a design.

  The formal analysis of properties starts after the reset state.

# 4 The AXI5 AIP Configuration

This chapter describes about configuration of the AXI5 AIP in the following Sections:

- "The AXI5 AIP Configuration Parameters" on page 27
- "The AXI5 AIP Interface Ports" on page 32
- "The AXI5 AIP Properties" on page 36
- "The AXI5 AIP Cover Properties" on page 78
- "Behavior of Properties" on page 85
- "The CONFIG_MAXOUTS Parameter Setting Change" on page 86
- "Supported Version" on page 89

## 4.1 The AXI5 AIP Configuration Parameters

**Table 4-1 The AXI5 AIP Configuration Parameters**

| Parameter Name | Description | Default value |
|---|---|---|
| AGENT_TYPE | Agent Type: either one of MASTER, SLAVE, MONITOR or CONSTRAINT | MASTER |
| READ_WRITE | Check type: BOTHRW (Read and Write) or RDONLY (Read Only) or WRONLY (Write Only) | BOTHRW |
| ENABLE_ASSERT | 1: Enable 0: Disable Assertions | 1 |
| ENABLE_ASSUME | 1: Enable 0: Disable Assumptions | 1 |
| ENABLE_COVER | 1: Enable 0: Disable Cover Properties | 1 |
| CHECK_FORMAL | 1: Use Formal 0: Use Simulation/Emulation | 1 |
| CHECK_PARAMETERS | Indicate check if parameter setting or not | 0 |
| CONFIG_USER | Indicate if use AWUSER/WUSER/BUSER/ARUSER/RUSER signals or not | 1 |

**Table 4-1 The AXI5 AIP Configuration Parameters**

| Parameter Name | Description | Default value |
|---|---|---|
| CONFIG_QOS | Indicate if use AWQOS and ARQOS signals or not (axi5 or later) | 1 |
| CONFIG_REGION | Indicate if use AWREGION and ARREGION signals or not (axi5 or later) | 1 |
| CONFIG_ATOMIC | Indicate if use AWATOP signal or not (AXI5 or later) | 1 |
| CONFIG_LOOP | Indicate if use AWLOOP, BLOOP, ARLOOP and RLOOP signals or not (AXI5 or later) | 1 |
| CONFIG_CHECK_TYPE | Indicate if use WDATACHK and RDATACHK signals or not (AXI5 or later) | 1 |
| CONFIG_POISON | Indicate if use WPOISON and RPOISON signals or not (AXI5 or later) | 1 |
| CONFIG_QOSACCEPT | Indicate if use VAWQOSACCEPT and VARQOSACCEPT signals or not (AXI5 or later) | 1 |
| CONFIG_TRACE | Indicate if use AWTRACE, BTRACE, ARTRACE and RTRACE signals or not (AXI5 or later) | 1 |
| CONFIG_UNTRANSLATED | Indicate if use AxMMUSECSID, AxMMUSID, AxMMUSSIDV, AxMMUSSID and AxMMUATST signals or not (AXI5 or later) | 1 |
| CONFIG_NSACCESS | Indicate if use AWNSAID and ARNSAID signals or not (AXI5 or later) | 1 |
| CONFIG_WAKEUP | Indicate if use AWAKEUP signal or not (AXI5 or later) | 1 |
| CONFIG_LOWPOWER | Indicate check if Low Power properties or not | 1 |
| | (1: check 0: no check) | |
| CONFIG_X_CHECK | Indicate check if X signal properties or not | 1 |
| | (1: check 0: no check) | |
| CONFIG_RECOMMEND | Indicate check if ARM recommendation properties or not | 1 |
| | (1: check 0: no check) | |
| CONFIG_WSTRB | Indicate check if Write Strobe related properties | 1 |
| | (1: check 0: no check) | |
| WDATA_STROBE | Indicate check if Write Data (1: Only valid byte lanes 0: All bits [better performance in Formal]) | 1 |
| RDATA_STROBE | Indicate check if Read Data (1: Only valid byte lanes 0: All bits [better performance in Formal]) | 1 |
| CONFIG_MAXOUTS | Indicate to check number of maximum outstanding transactions (0: Disable 1: Enable when need to check design implementation 2: Enable when need to constrain). For more information, refer to Section 4.6. | 1 |

**Table 4-1 The AXI5 AIP Configuration Parameters**

| Parameter Name | Description | Default value |
|---|---|---|
| RD_MAX_BURSTS | Maximum number of outstanding Read burst | 4 |
| WR_MAX_BURSTS | Maximum number of outstanding Write burst | 4 |
| MAXBURSTLENGTH | Maximum configured burst length | 256 |
| EXCL_DEPTH | Indicate Exclusive monitoring depth in slave | 4 |
|  | (>=1: Exclusive depth 0: No Exclusive support) |  |
| WR_OUT_OF_ORDER | Indicate if support Write Out Of Order Response or not (1: supported 0: not allow out of order) | 1 |
| RD_INTERLEAVE | Indicate if support Read Data Interleave or not | 1 |
|  | (1: allow interleaving 0: not allow interleaving) |  |
| RD_OUT_OF_ORDER | Indicate if support Read Out Of Order Data or not | 1 |
|  | (1: supported 0: not allow out of order) |  |
| WDATA_ADVANCE | Indicate if Write Data can be issued earlier than Write Address or not (1: possible 0: not allow) | 1 |
| WR_ALLOW_DECERR | Indicate if DECERR is allowed as write response | 1 |
|  | (1: possible 0: not allow) |  |
| WR_ALLOW_SLVERR | Indicate if SLVERR is allowed as write response | 1 |
|  | (1: possible 0: not allow) |  |
| RD_ALLOW_DECERR | Indicate if DECERR is allowed as read response | 1 |
|  | (1: possible 0: not allow) |  |
| RD_ALLOW_SLVERR | Indicate if SLVERR is allowed as read response | 1 |
|  | (1: possible 0: not allow) |  |
| CONFIG_WAITS | Indicate check if Valid/Ready wait cycles properties or not | 1 |
|  | (1: check 0: no check) |  |
| AW_MAX_WAITS | Maximum number of wait cycle from AWVALID to AWREADY, if set 0, liveness assertion or fairness constraint will be generated. | 16 |
| W_MAX_WAITS | Maximum number of wait cycle from WVALID to WREADY, if set 0, liveness assertion or fairness constraint will be generated. | 16 |
| B_MAX_WAITS | Maximum number of wait cycle from BVALID to BREADY, if set 0, liveness assertion or fairness constraint will be generated. | 16 |
| AR_MAX_WAITS | Maximum number of wait cycle from ARVALID to ARREADY, if set 0, liveness assertion or fairness constraint will be generated. | 16 |

Table 4-1 The AXI5 AIP Configuration Parameters

| Parameter Name | Description | Default value |
|---|---|---|
| R_MAX_WAITS | Maximum number of wait cycle from RVALID to RREADY, if set 0, liveness assertion or fairness constraint will be generated. | 16 |
| INTERCHANNEL_LATENCY | Indicate check if Inter-channel latency properties or not | 0 |
| | (1: check 0: no check) | |
| AW_W_MAX_LATENCY | Maximum latency from AWVALID to WVALID, if set 0, liveness assertion or fairness constraint will be generated. | 16 |
| W_AW_MAX_LATENCY | Maximum latency from WVALID to AWVALID, if set 0, liveness assertion or fairness constraint will be generated. | 16 |
| AWW_B_MAX_LATENCY | Maximum latency from address/data channels complete to BVALID, if set 0, liveness assertion or fairness constraint will be generated. | 16 |
| WLAST_MAX_LATENCY | Maximum latency from WVALID to WLAST, if set 0, liveness assertion or fairness constraint will be generated. | 16 |
| AR_R_MAX_LATENCY | Maximum latency from ARVALID to RVALID, if set 0, liveness assertion or fairness constraint will be generated. | 16 |
| RLAST_MAX_LATENCY | Maximum latency from RVALID to RLAST, if set 0, liveness assertion or fairness constraint will be generated. | 16 |
| CONFIG_BASE | Indicate check if basic properties or not.<br>(1: enable all basic properties 0: disable basic properties) | 1 |
| CONFIG_MPAM | Setting for Memory Partitioning and Monitoring (MPAM).<br>0: Disable MPAM support<br>1: Enable MPAM support | 0 |
| CONFIG_UNQID | Setting for unique ID indicator.<br>0: Interface does not have unique ID indicator signals ARIDUNQ/RIDUNQ/AWIDUNQ/BIDUNQ<br>1: Interface has unique ID indicator signals ARIDUNQ/RIDUNQ/AWIDUNQ/BIDUNQ | 0 |
| CONFIG_MTE | Setting for Memory Tagging Extension (MTE).<br>0: Disable MTE support<br>1: Enable MTE support | 0 |
| CONFIG_BCOMP | Setting for BCOMP signal.<br>0: Interface does not have BCOMP signal<br>1: Interface has BCOMP signal | 0 |

**Table 4-1 The AXI5 AIP Configuration Parameters**

| Parameter Name | Description | Default value |
|---|---|---|
| RD_DATA_CHUNKING | Setting for read data chunking.<br>0: Disable support for Read data chunking. Interface does not have the ARCHUNKEN/RCHUNKV/RCHUNKNUM/RCHUNKSTRB signals.<br>1: Enable support for Read data chunking. Interface contains the ARCHUNKEN/RCHUNKV/RCHUNKNUM/RCHUNKSTRB signals. | 0 |
| CONSISTENT_DECERR | Setting for DECERR in RRESP.<br>0: DECERR may be signaled on any number of read data beats.<br>1: DECERR is signaled for every beat of read data, or no beats of read data within each cache line of data. | 0 |
| REGULAR_TXN_ONLY | Setting for Regular Transactions.<br>0: All legal combinations of AxBURST, AxSIZE, and AxLEN are supported.<br>1: Only regular transactions are supported. | 0 |
| MAX_TXN_BYTES | Setting for Maximum transaction size and boundary.<br>0: The maximum size of a transaction is 4KB and transactions are not permitted to cross a 4KB boundary.<br>1: Set maximum size of transaction smaller than 4KB. | 0 |
| WDATA_WIDTH | Write Data bus bit width | 128 |
| RDATA_WIDTH | Read Data bus bit width | 128 |
| ADDR_WIDTH | Address bus bit width | 64 |
| ID_WIDTH | ID bit width | 8 |
| LEN_WIDTH | Length bit width | 8 |
| SIZE_WIDTH | Size bit width | 3 |
| AWUSER_WIDTH | AWUSER user signal bit width | 32 |
| WUSER_WIDTH | WUSER user signal bit width | 32 |
| BUSER_WIDTH | BUSER user signal bit width | 32 |
| ARUSER_WIDTH | ARUSER user signal bit width | 32 |
| RUSER_WIDTH | RUSER user signal bit width | 32 |
| LOOP_WIDTH | AWLOOP, BLOOP, ARLOOP and RLOOP signals bit width | 2 |
| MMUSID_WIDTH | AWMMUSID and ARMMUSID signals bit width | 8 |
| MMUSSID_WIDTH | AWMMUSSID and ARMMUSSID signals bit width | 4 |
| POISON_WIDTH | WPOISON and RPOISON signals bit width | 2 |
| FLOW_WIDTH | AWMMUFLOW and ARMMUFLOW signals bit width | 2 |

Table 4-1 **The AXI5 AIP Configuration Parameters**

| Parameter Name | Description | Default value |
|---|---|---|
| MPAM_WIDTH | AWMPAM and ARMPAM signals bit width | 11 |
| TAGOP_WIDTH | AWTAGOP and ARTAGOP signals bit width | 2 |
| TAGMATCH_WIDTH | BTAGMATCH signal bit width | 2 |
| ADDR_RANGE | Number of address ranges | 1 |
| MIN_ADDR | Minimum address for each address range. For example: " {32'h0, 32'h8000} in case of ADDR_RANGE=2" | {32'b0} |
| MAX_ADDR | Maximum address for each address range: for example {32'h3fff, 32'heffffff} in case of ADDR_RANGE=2 | {32'hffffffff} |
| COV_MAX_ID_WIDTH | Generate cover properties for xID per ID value (ID_WIDTH<=COV_MAX_ID_WIDTH) or each bit (ID_WIDTH > COV_MAX_ID_WIDTH) | 6 |

## 4.2 The AXI5 AIP Interface Ports

This Section describes interface signals of the AXI5 AIP.

### 4.2.1 AXI5 Interface Ports

Table 4-2 **AXI5 related Interface Ports**

| Signal Name | Signal Width | Description | |
|---|---|---|---|
| | | **Source** | **Destination** |
| aclk | 1 | Input clock from the system | |
| aresetn | 1 | Reset input from the system | |
| awid | ID_WIDTH | Master | Slave |
| awaddr | ADDR_WIDTH | Master | Slave |
| awlen | LEN_WIDTH | Master | Slave |
| awsize | 3 | Master | Slave |
| awburst | 2 | Master | Slave |
| awcache | 4 | Master | Slave |
| awprot | 3 | Master | Slave |
| awlock | 1 | Master | Slave |
| awqos | 4 | Master | Slave |
| awregion | 4 | Master | Slave |

**Table 4-2 AXI5 related Interface Ports**

| Signal Name | Signal Width | Description | |
|---|---|---|---|
| awuser | AWUSER_WIDTH | Master | Slave |
| awvalid | 1 | Master | Slave |
| awready | 1 | Master | Slave |
| wdata | WDATA_WIDTH | Master | Slave |
| wstrb | STRB_WIDTH | Master | Slave |
| wuser | WUSER_WIDTH | Master | Slave |
| wlast | 1 | Master | Slave |
| wvalid | 1 | Master | Slave |
| wready | 1 | Slave | Master |
| bid | ID_WIDTH | Slave | Master |
| bresp | 2 | Slave | Master |
| buser | BUSER_WIDTH | Slave | Master |
| bvalid | 1 | Slave | Master |
| bready | 1 | Master | Slave |
| arid | ID_WIDTH | Master | Slave |
| araddr | ADDR_WIDTH | Master | Slave |
| arlen | LEN_WIDTH | Master | Slave |
| arsize | 3 | Master | Slave |
| arburst | 2 | Master | Slave |
| arcache | 4 | Master | Slave |
| arprot | 3 | Master | Slave |
| arlock | 1 | Master | Slave |
| arqos | 4 | Master | Slave |
| arregion | 4 | Master | Slave |
| aruser | ARUSER_WIDTH | Master | Slave |
| arvalid | 1 | Master | Slave |
| arready | 1 | Slave | Master |
| rid | ID_WIDTH | Slave | Master |
| rdata | RDATA_WIDTH | Slave | Master |

**Table 4-2 AXI5 related Interface Ports**

| Signal Name | Signal Width | Description | |
|---|---|---|---|
| rresp | 4 | Slave | Master |
| ruser | RUSER_WIDTH | Slave | Master |
| rlast | 1 | Slave | Master |
| rvalid | 1 | Slave | Master |
| rready | 1 | Master | Slave |
| awatop | 6 | Master | Slave |
| awtrace | 1 | Master | Slave |
| awloop | LOOP_WIDTH | Master | Slave |
| awmmusecsid | 1 | Master | Slave |
| awmmusid | MMUSID_WIDTH | Master | Slave |
| awmmussidv | 1 | Master | Slave |
| awmmussid | MMUSSID_WIDTH | Master | Slave |
| awmmuatst | 1 | Master | Slave |
| awnsaid | 4 | Master | Slave |
| wdatachk | WDATACHK_WIDTH | Master | Slave |
| wpoison | POISON_WIDTH | Master | Slave |
| wtrace | 1 | Master | Slave |
| btrace | 1 | Slave | Master |
| bloop | LOOP_WIDTH | Slave | Master |
| artrace | 1 | Master | Slave |
| arloop | LOOP_WIDTH | Master | Slave |
| armmusecsid | 1 | Master | Slave |
| armmusid | MMUSID_WIDTH | Master | Slave |
| armmussidv | 1 | Master | Slave |
| armmussid | MMUSSID_WIDTH | Master | Slave |
| armmuatst | 1 | Master | Slave |
| arnsaid | 4 | Master | Slave |
| rdatachk | RDATACHK_WIDTH | Slave | Master |
| rpoison | POISON_WIDTH | Slave | Master |

**Table 4-2 AXI5 related Interface Ports**

| Signal Name | Signal Width | Description | |
|---|---|---|---|
| rtrace | 1 | Slave | Master |
| rloop | LOOP_WIDTH | Slave | Master |
| vawqosaccept | 4 | Slave | Master |
| varqosaccept | 4 | Slave | Master |
| awakeup | 1 | Clock controller | Master/Slave |
| awmmuflow | 2 | Master | Slave |
| awmpam | 11 | Master | Slave |
| awidunq | 1 | Master | Slave |
| awtagop | 2 | Master | Slave |
| wtag | Depends on data bit width | Master | Slave |
| wtagupdate | Depends on data bit width | Master | Slave |
| bidunq | 1 | Slave | Master |
| btagmatch | 2 | Slave | Master |
| bcomp | 1 | Slave | Master |
| armmuflow | 2 | Master | Slave |
| armpam | 11 | Master | Slave |
| aridunq | 1 | Master | Slave |
| archunken | 1 | Master | Slave |
| artagop | 2 | Master | Slave |
| ridunq | 1 | Slave | Master |
| rchunkv | 1 | Slave | Master |
| rchunknum | Depends on data bit width | Slave | Master |
| rchunkstrb | Depends on data bit width | Slave | Master |
| rtag | 2 | Slave | Master |

## 4.3 The AXI5 AIP Properties

Table 4-3 AXI5 Properties

| Property Name | Master/ Slave | Error Kind | Spec Reference | Property Description |
|---|---|---|---|---|
| ast_snps_axi5_awaddr_boundary | MASTER | ERROR | [ARM IHI 0022F.b] section A3.4.1 on page A3-48. | A write burst cannot cross a 4kbyte boundary. |
| ast_snps_axi5_awaddr_wrap_align | MASTER | ERROR | [ARM IHI 0022F.b] section A3.4.1 on page A3-50. | For wrapping bursts, the start address must be aligned to the size of each transfer. |
| ast_snps_axi5_awburst_no_reserved | MASTER | ERROR | [ARM IHI 0022F.b] section A3.4.1 table A3-3 on page A3-50. | When AWVALID is high, a value of 2'b11 on AWBURST is reserved. |
| ast_snps_axi5_awcache_legal | MASTER | ERROR | [ARM IHI 0022F.b] section A4.4 table A4-5 on page A4-69. | When AWVALID is HIGH and AWCACHE[1] is LOW then AWCACHE[3:2] are also LOW |
| ast_snps_axi5_awlen_wrap | MASTER | ERROR | [ARM IHI 0022F.b] section A3.4.1 on page A3-48. | For wrapping bursts, the burst length must be 2, 4, 8, or 16. |
| ast_snps_axi5_awlen_fixed | MASTER | ERROR | [ARM IHI 0022F.b] section A3.4.1 on page A3-48. | FIXED burst type supports burst length 1 to 16. |
| ast_snps_axi5_awlen_lock | MASTER | ERROR | [ARM IHI 0022F.b] section A7.2.4 on page A7-99. | The burst length for an exclusive access must not exceed 16 transfers. |
| ast_snps_axi5_awsize_max | MASTER | ERROR | [ARM IHI 0022F.b] section A3.4.1 on page A3-49. | The size of any transfer must not exceed the data bus width of either agent in the transaction. |
| ast_snps_axi5_awvalid_reset | MASTER | ERROR | [ARM IHI 0022F.b] section A3.1.2 Figure A3-1 on page A3-40. | During reset, a master interface must drive AWVALID LOW. The earliest point after reset that a master is permitted to begin driving AWVALID HIGH is at a rising ACLK edge after ARESETn is HIGH. |

**Table 4-3 AXI5 Properties**

| Property Name | Master/ Slave | Error Kind | Spec Reference | Property Description |
|---|---|---|---|---|
| ast_snps_axi5_awaddr_stable | MASTER | ERROR | [ARM IHI 0022F.b] section A3.2.1, and figure A3-2, on page A3-41. | Once AWVALID is asserted AWADDR must remain asserted until the handshake occurs, at a rising clock edge at which AWVALID and AWREADY are both asserted. |
| ast_snps_axi5_awburst_stable | MASTER | ERROR | [ARM IHI 0022F.b] section A3.2.1, and figure A3-2, on page A3-41. | Once AWVALID is asserted AWBURST must remain asserted until the handshake occurs, at a rising clock edge at which AWVALID and AWREADY are both asserted. |
| ast_snps_axi5_awid_stable | MASTER | ERROR | [ARM IHI 0022F.b] section A3.2.1, and figure A3-2, on page A3-41. | Once AWVALID is asserted AWID must remain asserted until the handshake occurs, at a rising clock edge at which AWVALID and AWREADY are both asserted. |
| ast_snps_axi5_awlen_stable | MASTER | ERROR | [ARM IHI 0022F.b] section A3.2.1, and figure A3-2, on page A3-41. | Once AWVALID is asserted AWLEN must remain asserted until the handshake occurs, at a rising clock edge at which AWVALID and AWREADY are both asserted. |
| ast_snps_axi5_awsize_stable | MASTER | ERROR | [ARM IHI 0022F.b] section A3.2.1, and figure A3-2, on page A3-41. | Once AWVALID is asserted AWSIZE must remain asserted until the handshake occurs, at a rising clock edge at which AWVALID and AWREADY are both asserted. |
| ast_snps_axi5_awlock_stable | MASTER | ERROR | [ARM IHI 0022F.b] section A3.2.1, and figure A3-2, on page A3-41. | Once AWVALID is asserted AWLOCK must remain asserted until the handshake occurs, at a rising clock edge at which AWVALID and AWREADY are both asserted. |

**Table 4-3 AXI5 Properties**

| Property Name | Master/ Slave | Error Kind | Spec Reference | Property Description |
|---|---|---|---|---|
| ast_snps_axi5_awcache_stable | MASTER | ERROR | [ARM IHI 0022F.b] section A3.2.1, and figure A3-2, on page A3-41. | Once AWVALID is asserted AWCACHE must remain asserted until the handshake occurs, at a rising clock edge at which AWVALID and AWREADY are both asserted. |
| ast_snps_axi5_awqos_stable | MASTER | ERROR | [ARM IHI 0022F.b] section A3.2.1, and figure A3-2, on page A3-41. | Once AWVALID is asserted AWQOS must remain asserted until the handshake occurs, at a rising clock edge at which AWVALID and AWREADY are both asserted. |
| ast_snps_axi5_awregion_stable | MASTER | ERROR | [ARM IHI 0022F.b] section A3.2.1, and figure A3-2, on page A3-41. | Once AWVALID is asserted AWREGION must remain asserted until the handshake occurs, at a rising clock edge at which AWVALID and AWREADY are both asserted. |
| ast_snps_axi5_awprot_stable | MASTER | ERROR | [ARM IHI 0022F.b] section A3.2.1, and figure A3-2, on page A3-41. | Once AWVALID is asserted AWPROT must remain asserted until the handshake occurs, at a rising clock edge at which AWVALID and AWREADY are both asserted. |
| ast_snps_axi5_awvalid_stable | MASTER | ERROR | [ARM IHI 0022F.b] section A3.2.2 on page A3-42. | When asserted, AWVALID must remain asserted until the rising clock edge after the master asserts AWREADY. |
| ast_snps_axi5_wvalid_reset | MASTER | ERROR | [ARM IHI 0022F.b] section A3.1.2 Figure A3-1 on page A3-40. | During reset, a master interface must drive WVALID LOW. The earliest point after reset that a master is permitted to begin driving WVALID HIGH is at a rising ACLK edge after ARESETn is HIGH. |

Table 4-3 **AXI5 Properties**

| Property Name | Master/ Slave | Error Kind | Spec Reference | Property Description |
|---|---|---|---|---|
| ast_snps_axi5_wdata_stable | MASTER | ERROR | [ARM IHI 0022F.b] section A3.2.1, and figure A3-2, on page A3-41. | Once WVALID is asserted WDATA must remain asserted until the handshake occurs, at a rising clock edge at which WVALID and WREADY are both asserted. |
| ast_snps_axi5_wdata_stable_allbits | MASTER | ERROR | [ARM IHI 0022F.b] section A3.2.1, and figure A3-2, on page A3-41. | Once WVALID is asserted WDATA must remain asserted until the handshake occurs, at a rising clock edge at which WVALID and WREADY are both asserted. |
| ast_snps_axi5_wlast_stable | MASTER | ERROR | [ARM IHI 0022F.b] section A3.2.1, and figure A3-2, on page A3-41. | Once WVALID is asserted WLAST must remain asserted until the handshake occurs, at a rising clock edge at which WVALID and WREADY are both asserted. |
| ast_snps_axi5_wstrb_stable | MASTER | ERROR | [ARM IHI 0022F.b] section A3.2.1, and figure A3-2, on page A3-41. | Once WVALID is asserted WSTRB must remain asserted until the handshake occurs, at a rising clock edge at which WVALID and WREADY are both asserted. |
| ast_snps_axi5_wvalid_stable | MASTER | ERROR | [ARM IHI 0022F.b] section A3.2.2 on page A3-43. | When asserted, WVALID must remain asserted until the rising clock edge after the master asserts WREADY. |
| ast_snps_axi5_araddr_boundary | MASTER | ERROR | [ARM IHI 0022F.b] section A3.4.1 on page A3-48. | A burst must not cross a 4KB address boundary. |
| ast_snps_axi5_araddr_wrap_align | MASTER | ERROR | [ARM IHI 0022F.b] section A3.4.1 on page A3-50. | For wrapping bursts, the start address must be aligned to the size of each transfer. |
| ast_snps_axi5_arburst_no_reserved | MASTER | ERROR | [ARM IHI 0022F.b] section A3.4.1 table A3-3 on page A3-50. | When ARVALID is high, a value of 2'b11 on ARBURST is reserved. |

**Table 4-3 AXI5 Properties**

| Property Name | Master/ Slave | Error Kind | Spec Reference | Property Description |
|---|---|---|---|---|
| ast_snps_axi5_arcache_legal | MASTER | ERROR | [ARM IHI 0022F.b] section A4.4 table A4-5 on page A4-69. | When ARVALID is HIGH and ARCACHE[1] is LOW then ARCACHE[3:2] are also LOW |
| ast_snps_axi5_arlen_wrap | MASTER | ERROR | [ARM IHI 0022F.b] section A3.4.1 on page A3-48. | For wrapping bursts, the burst length must be 2, 4, 8, or 16. |
| ast_snps_axi5_arlen_fixed | MASTER | ERROR | [ARM IHI 0022F.b] section A3.4.1 on page A3-48. | FIXED burst type supports burst length 1 to 16. |
| ast_snps_axi5_arsize_max | MASTER | ERROR | [ARM IHI 0022F.b] section A3.4.1 on page A3-49. | The size of any transfer must not exceed the data bus width of either agent in the transaction. |
| ast_snps_axi5_arvalid_reset | MASTER | ERROR | [ARM IHI 0022F.b] section A3.1.2 Figure A3-1 on page A3-40. | During reset, a master interface must drive ARVALID LOW. The earliest point after reset that a master is permitted to begin driving ARVALID HIGH is at a rising ACLK edge after ARESETn is HIGH. |
| ast_snps_axi5_araddr_stable | MASTER | ERROR | [ARM IHI 0022F.b] section A3.2.1, and figure A3-2, on page A3-41. | Once ARVALID is asserted ARADDR must remain asserted until the handshake occurs, at a rising clock edge at which ARVALID and ARREADY are both asserted. |
| ast_snps_axi5_arburst_stable | MASTER | ERROR | [ARM IHI 0022F.b] section A3.2.1, and figure A3-2, on page A3-41. | Once ARVALID is asserted ARBURST must remain asserted until the handshake occurs, at a rising clock edge at which ARVALID and ARREADY are both asserted. |
| ast_snps_axi5_arid_stable | MASTER | ERROR | [ARM IHI 0022F.b] section A3.2.1, and figure A3-2, on page A3-41. | Once ARVALID is asserted ARID must remain asserted until the handshake occurs, at a rising clock edge at which ARVALID and ARREADY are both asserted. |

**Table 4-3 AXI5 Properties**

| Property Name | Master/ Slave | Error Kind | Spec Reference | Property Description |
|---|---|---|---|---|
| ast_snps_axi5_arlen_stable | MASTER | ERROR | [ARM IHI 0022F.b] section A3.2.1, and figure A3-2, on page A3-41. | Once ARVALID is asserted ARLEN must remain asserted until the handshake occurs, at a rising clock edge at which ARVALID and ARREADY are both asserted. |
| ast_snps_axi5_arsize_stable | MASTER | ERROR | [ARM IHI 0022F.b] section A3.2.1, and figure A3-2, on page A3-41. | Once ARVALID is asserted ARSIZE must remain asserted until the handshake occurs, at a rising clock edge at which ARVALID and ARREADY are both asserted. |
| ast_snps_axi5_arlock_stable | MASTER | ERROR | [ARM IHI 0022F.b] section A3.2.1, and figure A3-2, on page A3-41. | Once ARVALID is asserted ARLOCK must remain asserted until the handshake occurs, at a rising clock edge at which ARVALID and ARREADY are both asserted. |
| ast_snps_axi5_arcache_stable | MASTER | ERROR | [ARM IHI 0022F.b] section A3.2.1, and figure A3-2, on page A3-41. | Once ARVALID is asserted ARCACHE must remain asserted until the handshake occurs, at a rising clock edge at which ARVALID and ARREADY are both asserted. |
| ast_snps_axi5_arqos_stable | MASTER | ERROR | [ARM IHI 0022F.b] section A3.2.1, and figure A3-2, on page A3-41. | Once ARVALID is asserted ARQOS must remain asserted until the handshake occurs, at a rising clock edge at which ARVALID and ARREADY are both asserted. |
| ast_snps_axi5_arregion_stable | MASTER | ERROR | [ARM IHI 0022F.b] section A3.2.1, and figure A3-2, on page A3-41. | Once ARVALID is asserted ARREGION must remain asserted until the handshake occurs, at a rising clock edge at which ARVALID and ARREADY are both asserted. |

Table 4-3 **AXI5 Properties**

| Property Name | Master/ Slave | Error Kind | Spec Reference | Property Description |
|---|---|---|---|---|
| ast_snps_axi5_arprot_stable | MASTER | ERROR | [ARM IHI 0022F.b] section A3.2.1, and figure A3-2, on page A3-41. | Once ARVALID is asserted ARPROT must remain asserted until the handshake occurs, at a rising clock edge at which ARVALID and ARREADY are both asserted. |
| ast_snps_axi5_arvalid_stable | MASTER | ERROR | [ARM IHI 0022F.b] section A3.2.2 on page A3-43. | When asserted, ARVALID must remain asserted until the rising clock edge after the master asserts ARREADY. |
| ast_snps_axi5_awaddr_x | MASTER | ERROR | Signal Integrity | When AWVALID is high, a value of X/Z on AWADDR is not permitted. |
| ast_snps_axi5_awburst_x | MASTER | ERROR | Signal Integrity | When AWVALID is high, a value of X/Z on AWBURST is not permitted. |
| ast_snps_axi5_awid_x | MASTER | ERROR | Signal Integrity | When AWVALID is high, a value of X/Z on AWID is not permitted. |
| ast_snps_axi5_awlen_x | MASTER | ERROR | Signal Integrity | When AWVALID is high, a value of X/Z on AWLEN is not permitted. |
| ast_snps_axi5_awsize_x | MASTER | ERROR | Signal Integrity | When AWVALID is high, a value of X/Z on AWSIZE is not permitted. |
| ast_snps_axi5_awlock_x | MASTER | ERROR | Signal Integrity | When AWVALID is high, a value of X/Z on AWLOCK is not permitted. |
| ast_snps_axi5_awcache_x | MASTER | ERROR | Signal Integrity | When AWVALID is high, a value of X/Z on AWCACHE is not permitted. |
| ast_snps_axi5_awqos_x | MASTER | ERROR | Signal Integrity | When AWVALID is high, a value of X/Z on AWQOS is not permitted. |
| ast_snps_axi5_awregion_x | MASTER | ERROR | Signal Integrity | When AWVALID is high, a value of X/Z on AWREGION is not permitted. |

**Table 4-3 AXI5 Properties**

| Property Name | Master/ Slave | Error Kind | Spec Reference | Property Description |
|---|---|---|---|---|
| ast_snps_axi5_awprot_x | MASTER | ERROR | Signal Integrity | When AWVALID is high, a value of X/Z on AWPROT is not permitted. |
| ast_snps_axi5_awvalid_x | MASTER | ERROR | Signal Integrity | When not in reset, a value of X/Z on AWVALID is not permitted. |
| ast_snps_axi5_wdata_x | MASTER | ERROR | Signal Integrity | When WVALID is high, a value of X/Z on WDATA is not permitted. |
| ast_snps_axi5_wlast_x | MASTER | ERROR | Signal Integrity | When WVALID is high, a value of X/Z on WLAST is not permitted. |
| ast_snps_axi5_wstrb_x | MASTER | ERROR | Signal Integrity | When WVALID is high, a value of X/Z on WSTRB is not permitted. |
| ast_snps_axi5_wvalid_x | MASTER | ERROR | Signal Integrity | When not in reset, a value of X/Z on WVALID is not permitted. |
| ast_snps_axi5_bready_x | MASTER | ERROR | Signal Integrity | When not in reset, a value of X/Z on BREADY is not permitted. |
| ast_snps_axi5_araddr_x | MASTER | ERROR | Signal Integrity | When ARVALID is high, a value of X/Z on ARADDR is not permitted. |
| ast_snps_axi5_arburst_x | MASTER | ERROR | Signal Integrity | When ARVALID is high, a value of X/Z on ARBURST is not permitted. |
| ast_snps_axi5_arid_x | MASTER | ERROR | Signal Integrity | When ARVALID is high, a value of X/Z on ARID is not permitted. |
| ast_snps_axi5_arlen_x | MASTER | ERROR | Signal Integrity | When ARVALID is high, a value of X/Z on ARLEN is not permitted. |
| ast_snps_axi5_arsize_x | MASTER | ERROR | Signal Integrity | When ARVALID is high, a value of X/Z on ARSIZE is not permitted. |
| ast_snps_axi5_arlock_x | MASTER | ERROR | Signal Integrity | When ARVALID is high, a value of X/Z on ARLOCK is not permitted. |

**Table 4-3 AXI5 Properties**

| Property Name | Master/ Slave | Error Kind | Spec Reference | Property Description |
|---|---|---|---|---|
| ast_snps_axi5_arcache_x | MASTER | ERROR | Signal Integrity | When ARVALID is high, a value of X/Z on ARCACHE is not permitted. |
| ast_snps_axi5_arqos_x | MASTER | ERROR | Signal Integrity | When ARVALID is high, a value of X/Z on ARQOS is not permitted. |
| ast_snps_axi5_arregion_x | MASTER | ERROR | Signal Integrity | When ARVALID is high, a value of X/Z on ARREGION is not permitted. |
| ast_snps_axi5_arprot_x | MASTER | ERROR | Signal Integrity | When ARVALID is high, a value of X/Z on ARPROT is not permitted. |
| ast_snps_axi5_arvalid_x | MASTER | ERROR | Signal Integrity | When not in reset, a value of X/Z on ARVALID is not permitted. |
| ast_snps_axi5_rready_x | MASTER | ERROR | Signal Integrity | When not in reset, a value of X/Z on RREADY is not permitted. |
| ast_snps_axi5_wlast_beat | MASTER | ERROR | [ARM IHI 0022F.b] section A3.2.2 on page A3-43. | The master must assert the WLAST signal while it is driving the final write transfer in the burst. |
| ast_snps_axi5_wstrb_align | MASTER | ERROR | [ARM IHI 0022F.b] section A3.4.3 on page A3-54. | A master must encure that the write strobes are HIGH only for byte lanes that contains valid data. |
| ast_snps_axi5_wstrb_fixed | MASTER | ERROR | [ARM IHI 0022F.b] section A3.4.1 on page A3-49. | The byte lanes that are valid are constant for all beats in the burst. However, within those byte lanes, the actual bytes that have WSTRB asserted can differe for each beat in the burst. |
| ast_snps_axi5_bready_max_waits | MASTER | WARNING | Timeout check | BREADY should be asserted within B_MAX_WAITS cycles of BVALID being asserted. |
| ast_snps_axi5_bready_eventually | MASTER | WARNING | Timeout check | BREADY should be asserted eventually when BVALID being asserted. |

**Table 4-3 AXI5 Properties**

| Property Name | Master/ Slave | Error Kind | Spec Reference | Property Description |
|---|---|---|---|---|
| ast_snps_axi5_rready_max_waits | MASTER | WARNING | Timeout check | RREADY should be asserted within R_MAX_WAITS cycles of RVALID being asserted. |
| ast_snps_axi5_rready_eventually | MASTER | WARNING | Timeout check | RREADY should be asserted eventually when RVALID being asserted. |
| ast_snps_axi5_awvalid_maxouts | MASTER | WARNING | Maximum number of outstanding transactions. | Master issued Write Requests for more than WR_MAX_BURSTS. |
| ast_snps_axi5_wvalid_maxouts | MASTER | WARNING | Maximum number of outstanding transactions. | Master issued Write Data for more than WR_MAX_BURSTS. |
| ast_snps_axi5_arvalid_maxouts | MASTER | WARNING | Maximum number of outstanding transactions. | Master issued Read Requests for more than RD_MAX_BURSTS. |
| ast_snps_axi5_no_wdata_advanced | MASTER | WARNING | The relation between Write Address and Write Data. | Some design may not accept Write Data before Write Address. |
| ast_snps_axi5_excl_awcache | MASTER | ERROR | [ARM IHI 0022F.b] section A7.2.4 on page A7-99. | The value of the AxCACHE signals must guarantee that the slave that is monitoring the exclusive access sees the transaction. For example, an exclusive access must not have an AxCACHE value that indicates that the transaction is Cacheable. |
| ast_snps_axi5_excl_arcache | MASTER | ERROR | [ARM IHI 0022F.b] section A7.2.4 on page A7-99. | The value of the AxCACHE signals must guarantee that the slave that is monitoring the exclusive access sees the transaction. For example, an exclusive access must not have an AxCACHE value that indicates that the transaction is Cacheable. |

**Table 4-3 AXI5 Properties**

| Property Name | Master/ Slave | Error Kind | Spec Reference | Property Description |
|---|---|---|---|---|
| ast_snps_axi5_excl_addr_align | MASTER | ERROR | [ARM IHI 0022F.b] section A7.2.4 on page A7-99. | The address of an exclusive access must be aligned to the total number of bytes in the transaction, that is, the product of the burst size and burst length. |
| ast_snps_axi5_excl_length | MASTER | ERROR | [ARM IHI 0022F.b] section A7.2.4 on page A7-99. | In axi5, the burst length for an exclusive access must not exceed 16 transfers. |
| ast_snps_axi5_excl_transfer_size | MASTER | ERROR | [ARM IHI 0022F.b] section A7.2.4 on page A7-99. | The number of bytes to be transferred in an exclusive access burst must be a power of 2, that is, 1, 2, 4, 8, 16, 32, 64, or 128 bytes. |
| ast_snps_axi5_excl_max_bytes | MASTER | ERROR | [ARM IHI 0022F.b] section A7.2.4 on page A7-99. | The maximum number of bytes that can be transferred in an exclusive burst is 128. |
| ast_snps_axi5_excl_max_depth | MASTER | ERROR | EXCL_DEPTH overflow | Outstanding Exclusive Reads issued more than EXCL_DEPTH. |
| ast_snps_axi5_excl_no_conc_rw | MASTER | ERROR | [ARM IHI 0022F.b] section A7.2.2 on page A7-98. | It is Recommended that exclusive reads and writes with the same ID not be issued at the same time |
| ast_snps_axi5_excl_addr_match | MASTER | ERROR | [ARM IHI 0022F.b] section A7.2.4 on page A7-99. | The address for the exclusive read and the exclusive write must be identical. |
| ast_snps_axi5_excl_size_match | MASTER | ERROR | [ARM IHI 0022F.b] section A7.2.4 on page A7-99. | The size for the exclusive read and exclusive write transactions must be identical. |
| ast_snps_axi5_excl_length_match | MASTER | ERROR | [ARM IHI 0022F.b] section A7.2.4 on page A7-99. | The length for the exclusive read and exclusive write transactions must be identical. |
| ast_snps_axi5_excl_burst_match | MASTER | ERROR | [ARM IHI 0022F.b] section A7.2.4 on page A7-99. | The BURST field for the exclusive read and exclusive write transactions must be identical. |

**Table 4-3 AXI5 Properties**

| Property Name | Master/ Slave | Error Kind | Spec Reference | Property Description |
|---|---|---|---|---|
| ast_snps_axi5_excl_cache_match | MASTER | ERROR | [ARM IHI 0022F.b] section A7.2.4 on page A7-99. | The CACHE field for the exclusive read and exclusive write transactions must be identical. |
| ast_snps_axi5_excl_region_match | MASTER | ERROR | [ARM IHI 0022F.b] section A7.2.4 on page A7-99. | The REGION field for the exclusive read and exclusive write transactions must be identical. |
| ast_snps_axi5_excl_prot_match | MASTER | ERROR | [ARM IHI 0022F.b] section A7.2.4 on page A7-99. | The PROT field for the exclusive read and exclusive write transactions must be identical. |
| ast_snps_axi5_excl_user_match | MASTER | ERROR | [ARM IHI 0022F.b] section A7.2.4 on page A7-99. | The USER field for the exclusive read and exclusive write transactions must be identical. |
| ast_snps_axi5_excl_read_done | MASTER | ERROR | [ARM IHI 0022F.b] section A7.2.2 on page A7-98. | A master must not start the write part of an exclusive access sequence until the read part is complete. |
| ast_snps_axi5_awuser_stable | MASTER | ERROR | [ARM IHI 0022F.b] section A3.2.1, and figure A3-2, on page A3-41. | Once AWVALID is asserted AWUSER must remain asserted until the handshake occurs, at a rising clock edge at which AWVALID and AWREADY are both asserted. |
| ast_snps_axi5_wuser_stable | MASTER | ERROR | [ARM IHI 0022F.b] section A3.2.1, and figure A3-2, on page A3-41. | Once WVALID is asserted WUSER must remain asserted until the handshake occurs, at a rising clock edge at which WVALID and WREADY are both asserted. |
| ast_snps_axi5_aruser_stable | MASTER | ERROR | [ARM IHI 0022F.b] section A3.2.1, and figure A3-2, on page A3-41. | Once ARVALID is asserted ARUSER must remain asserted until the handshake occurs, at a rising clock edge at which ARVALID and ARREADY are both asserted. |

**Table 4-3 AXI5 Properties**

| Property Name | Master/ Slave | Error Kind | Spec Reference | Property Description |
|---|---|---|---|---|
| ast_snps_axi5_awuser_x | MASTER | ERROR | Signal Integrity | When AWVALID is high, a value of X/Z on AWUSER is not permitted. |
| ast_snps_axi5_wuser_x | MASTER | ERROR | Signal Integrity | When WVALID is high, a value of X/Z on WUSER is not permitted. |
| ast_snps_axi5_aruser_x | MASTER | ERROR | Signal Integrity | When ARVALID is high, a value of X/Z on ARUSER is not permitted. |
| ast_snps_axi5_awready_x | SLAVE | ERROR | Signal Integrity | When not in reset, a value of X/Z on AWREADY is not permitted. |
| ast_snps_axi5_wready_x | SLAVE | ERROR | Signal Integrity | When not in reset, a value of X/Z on WREADY is not permitted. |
| ast_snps_axi5_bresp_after_waddr | SLAVE | ERROR | [ARM IHI 0022F.b] section A3.3.1 on page A3-46. | The slave must wait for AWVALID, AWREADY, WVALID, and WREADY to be asserted before asserting BVALID. |
| ast_snps_axi5_bresp_after_wdata | SLAVE | ERROR | [ARM IHI 0022F.b] section A3.3.1 on page A3-46. | The slave must also wait for WLAST to be asserted before asserting BVALID because the write response, BRESP must be signaled only after the last data transfer of a write transaction. |
| ast_snps_axi5_no_extra_bid | SLAVE | ERROR | [ARM IHI 0022F.b] section A3.2.2 on page A3-43 and A5.1 on page A5-80. | The slave can assert the BVALID signal only when it drives a valid write response. Slaves are required to reflect on the appropriate BID response an AXI ID received from a master. |
| ast_snps_axi5_bvalid_reset | SLAVE | ERROR | [ARM IHI 0022F.b] section A3.1.2 on page A3-40. | During reset, a slave interface must drive BVALID LOW. |

**Table 4-3 AXI5 Properties**

| Property Name | Master/ Slave | Error Kind | Spec Reference | Property Description |
|---|---|---|---|---|
| ast_snps_axi5_bid_stable | SLAVE | ERROR | [ARM IHI 0022F.b] section A3.2.1, and figure A3-2, on page A3-41. | Once BVALID is asserted BID must remain asserted until the handshake occurs, at a rising clock edge at which BVALID and BREADY are both asserted. |
| ast_snps_axi5_bresp_stable | SLAVE | ERROR | [ARM IHI 0022F.b] section A3.2.1, and figure A3-2, on page A3-41. | Once BVALID is asserted BRESP must remain asserted until the handshake occurs, at a rising clock edge at which BVALID and BREADY are both asserted. |
| ast_snps_axi5_bvalid_stable | SLAVE | ERROR | [ARM IHI 0022F.b] section A3.2.2 on page A3-43. | When asserted, BVALID must remain asserted until the rising clock edge after the master asserts BREADY. |
| ast_snps_axi5_bresp_as_decerr | SLAVE | ERROR | [ARM IHI 0022F.b] section A3.4.4, and Table A3-4, on page A3-59. | If the interconnect cannot successfully decode a slave access, it must return the DECERR response. |
| ast_snps_axi5_bresp_no_decerr | SLAVE | WARNING | Design specific. | This design expects not to respond with DECERR. |
| ast_snps_axi5_bresp_no_slverr | SLAVE | WARNING | Design specific. | This design expects not to respond with SLVERR. |
| ast_snps_axi5_bid_x | SLAVE | ERROR | Signal Integrity | When BVALID is High, a value of X/Z on BID is not permitted. |
| ast_snps_axi5_bresp_x | SLAVE | ERROR | Signal Integrity | When BVALID is High, a value of X/Z on BRESP is not permitted. |
| ast_snps_axi5_bvalid_x | SLAVE | ERROR | Signal Integrity | When not in reset, a value of X/Z on BVALID is not permitted. |
| ast_snps_axi5_arready_x | SLAVE | ERROR | Signal Integrity | When not in reset, a value of X/Z on ARREADY is not permitted. |

**Table 4-3 AXI5 Properties**

| Property Name | Master/ Slave | Error Kind | Spec Reference | Property Description |
|---|---|---|---|---|
| ast_snps_axi5_no_extra_rid | SLAVE | ERROR | [ARM IHI 0022F.b] section A3.2.2 on page A3-43 and A5.1 on page A5-80. | The slave can assert the RVALID signal only when it drives a valid read response. Slaves are required to reflect on the appropriate RID response an AXI ID received from a master. |
| ast_snps_axi5_rvalid_reset | SLAVE | ERROR | [ARM IHI 0022F.b] section A3.1.2 on page A3-40. | During reset, a slave interface must drive RVALID LOW. |
| ast_snps_axi5_rdata_stable | SLAVE | ERROR | [ARM IHI 0022F.b] section A3.2.1, and figure A3-2, on page A3-41. | Once RVALID is asserted RDATA must remain asserted until the handshake occurs, at a rising clock edge at which RVALID and RREADY are both asserted. |
| ast_snps_axi5_rdata_stable_allbits | SLAVE | ERROR | [ARM IHI 0022F.b] section A3.2.1, and figure A3-2, on page A3-41. | Once RVALID is asserted RDATA must remain asserted until the handshake occurs, at a rising clock edge at which RVALID and RREADY are both asserted. |
| ast_snps_axi5_rid_stable | SLAVE | ERROR | [ARM IHI 0022F.b] section A3.2.1, and figure A3-2, on page A3-41. | Once RVALID is asserted RID must remain asserted until the handshake occurs, at a rising clock edge at which RVALID and RREADY are both asserted. |
| ast_snps_axi5_rlast_stable | SLAVE | ERROR | [ARM IHI 0022F.b] section A3.2.1, and figure A3-2, on page A3-41. | Once RVALID is asserted RLAST must remain asserted until the handshake occurs, at a rising clock edge at which RVALID and RREADY are both asserted. |

**Table 4-3 AXI5 Properties**

| Property Name | Master/ Slave | Error Kind | Spec Reference | Property Description |
|---|---|---|---|---|
| ast_snps_axi5_rresp_stable | SLAVE | ERROR | [ARM IHI 0022F.b] section A3.2.1, and figure A3-2, on page A3-41. | Once RVALID is asserted RRESP must remain asserted until the handshake occurs, at a rising clock edge at which RVALID and RREADY are both asserted. |
| ast_snps_axi5_rvalid_stable | SLAVE | ERROR | [ARM IHI 0022F.b] section A3.2.2 on page A3-43. | When asserted, RVALID must remain asserted until the rising clock edge after the master asserts RREADY. |
| ast_snps_axi5_rresp_as_decerr | SLAVE | ERROR | [ARM IHI 0022F.b] section A3.4.4, and Table A3-4, on page A3-59. | If the interconnect cannot successfully decode a slave access, it must return the DECERR response. |
| ast_snps_axi5_rresp_no_decerr | SLAVE | WARNING | Design specific. | This design expects not to respond with DECERR. |
| ast_snps_axi5_rresp_no_slverr | SLAVE | WARNING | Design specific. | This design expects not to respond with SLVERR. |
| ast_snps_axi5_rdata_x | SLAVE | ERROR | Signal Integrity | When RVALID is High, a value of X/Z on RDATA is not permitted. |
| ast_snps_axi5_rid_x | SLAVE | ERROR | Signal Integrity | When RVALID is High, a value of X/Z on RID is not permitted. |
| ast_snps_axi5_rlast_x | SLAVE | ERROR | Signal Integrity | When RVALID is High, a value of X/Z on RLAST is not permitted. |
| ast_snps_axi5_rresp_x | SLAVE | ERROR | Signal Integrity | When RVALID is High, a value of X/Z on RRESP is not permitted. |
| ast_snps_axi5_rvalid_x | SLAVE | ERROR | Signal Integrity | When not in reset, a value of X/Z on RVALID is not permitted. |
| ast_snps_axi5_rlast_beat | SLAVE | ERROR | [ARM IHI 0022F.b] section A3.2.2 on page A3-43. | The slave must assert the RLAST signal while it is driving the final write transfer in the burst. |

**Table 4-3 AXI5 Properties**

| Property Name | Master/ Slave | Error Kind | Spec Reference | Property Description |
|---|---|---|---|---|
| ast_snps_axi5_bresp_exokay_allow | SLAVE | ERROR | [ARM IHI 0022F.b] section A7.2.3 on page A7-99. | An EXOKAY write response can only be given to an exclusive write access. |
| ast_snps_axi5_bresp_exokay_not_allow | SLAVE | ERROR | [ARM IHI 0022F.b] section A7.2.3 on page A7-99. | A slave that does not support exclusive accesses can ignore the AxLOCK signals. It must provide an OKAY response for both normal and exclusive accesses. |
| ast_snps_axi5_rresp_exokay_allow | SLAVE | ERROR | [ARM IHI 0022F.b] section A7.2.3 on page A7-99. | An EXOKAY read response can only be given to an exclusive read access. |
| ast_snps_axi5_rresp_exokay_not_allow | SLAVE | ERROR | [ARM IHI 0022F.b] section A7.2.3 on page A7-99. | A slave that does not support exclusive accesses can ignore the AxLOCK signals. It must provide an OKAY response for both normal and exclusive accesses. |
| ast_snps_axi5_awready_max_waits | SLAVE | WARNING | Timeout check | AWREADY should be asserted within AW_MAX_WAITS cycles of AWVALID being asserted. |
| ast_snps_axi5_awready_eventually | SLAVE | WARNING | Timeout check | AWREADY should be asserted eventually when AWVALID being asserted. |
| ast_snps_axi5_wready_max_waits | SLAVE | WARNING | Timeout check | WREADY should be asserted within W_MAX_WAITS cycles of WVALID being asserted. |
| ast_snps_axi5_wready_eventually | SLAVE | WARNING | Timeout check | WREADY should be asserted eventually when WVALID being asserted. |
| ast_snps_axi5_arready_max_waits | SLAVE | WARNING | Timeout check | ARREADY should be asserted within AR_MAX_WAITS cycles of ARVALID being asserted. |
| ast_snps_axi5_arready_eventually | SLAVE | WARNING | Timeout check | ARREADY should be asserted eventually when ARVALID being asserted. |

**Table 4-3 AXI5 Properties**

| Property Name | Master/ Slave | Error Kind | Spec Reference | Property Description |
|---|---|---|---|---|
| ast_snps_axi5_buser_stable | SLAVE | ERROR | [ARM IHI 0022F.b] section A3.2.1, and figure A3-2, on page A3-41. | Once BVALID is asserted BUSER must remain asserted until the handshake occurs, at a rising clock edge at which BVALID and BREADY are both asserted. |
| ast_snps_axi5_ruser_stable | SLAVE | ERROR | [ARM IHI 0022F.b] section A3.2.1, and figure A3-2, on page A3-41. | Once RVALID is asserted RUSER must remain asserted until the handshake occurs, at a rising clock edge at which RVALID and RREADY are both asserted. |
| ast_snps_axi5_buser_x | SLAVE | ERROR | Signal Integrity | When BVALID is high, a value of X/Z on BUSER is not permitted. |
| ast_snps_axi5_ruser_x | SLAVE | ERROR | Signal Integrity | When RVALID is high, a value of X/Z on RUSER is not permitted. |
| ast_snps_axi5_awready_maxouts | SLAVE | WARNING | Maximum number of outstanding transactions. | Slave should not accept Write Requests for more than WR_MAX_BURSTS. |
| ast_snps_axi5_wready_maxouts | SLAVE | WARNING | Maximum number of outstanding transactions. | Slave should not accept Write Data for more than WR_MAX_BURSTS. |
| ast_snps_axi5_arready_maxouts | SLAVE | WARNING | Maximum number of outstanding transactions. | Slave should not accept Read Requests for more than RD_MAX_BURSTS. |
| ast_snps_axi5_aw_maxbursts | MASTER | ERROR | Maximum Burst Length. | Master cannot issue AWLEN greater than the configured maximum burst length. |
| ast_snps_axi5_ar_maxbursts | MASTER | ERROR | Maximum Burst Length. | Master cannot issue ARLEN greater than the configured maximum burst length. |
| ast_snps_axi5_param_wdata_width |  | ERROR | Parameter Setting | WDATA_WIDTH should be either one of 32, 64, 128, 256, 512 or 1024. |

**Table 4-3 AXI5 Properties**

| Property Name | Master/ Slave | Error Kind | Spec Reference | Property Description |
|---|---|---|---|---|
| ast_snps_axi5_param_rdata_width | | ERROR | Parameter Setting | "RDATA_WIDTH should be either one of 32, 64, 128, 256, 512 or 1024. |
| ast_snps_axi5_param_maxwbursts | | ERROR | Parameter Setting | WR_MAX_BURSTS should be greater than or equal to 1. |
| ast_snps_axi5_param_maxrbursts | | ERROR | Parameter Setting | RD_MAX_BURSTS should be greater than or equal to 1. |
| ast_snps_axi5_param_maxburstlength | | ERROR | Parameter Setting | MAXBURSTLENGTH should be within 1 and 256. |
| ast_snps_axi5_wlast_max_latency | MASTER | WARNING | Recommendation to avoid deadlock. | If WVALID is asserted without WLAST, corresponding WLAST should be asserted within WLAST_MAX_LATENCY cycles. |
| ast_snps_axi5_aw_w_max_latency | MASTER | WARNING | Recommendation to avoid deadlock. | If AWVALID is asserted, corresponding WVALID should be asserted within AW_W_MAX_LATENCY cycles. |
| ast_snps_axi5_w_aw_max_latency | MASTER | WARNING | Recommendation to avoid deadlock. | If WVALID is asserted, corresponding AWVALID should be asserted within W_AW_MAX_LATENCY cycles. |
| ast_snps_axi5_aww_b_max_latency | SLAVE | WARNING | Recommendation to avoid deadlock. | If both write address and write data completed, corresponding BVALID should be asserted within AWW_B_MAX_LATENCY cycles. |
| ast_snps_axi5_ar_r_max_latency | SLAVE | WARNING | Recommendation to avoid deadlock. | If ARVALID is asserted, corresponding RVALID should be asserted within AR_R_MAX_LATENCY cycles. |

**Table 4-3 AXI5 Properties**

| Property Name | Master/ Slave | Error Kind | Spec Reference | Property Description |
|---|---|---|---|---|
| ast_snps_axi5_rlast_max_latency | SLAVE | WARNING | Recommendation to avoid deadlock. | If RVALID is asserted without RLAST, corresponding RLAST should be asserted within RLAST_MAX_LATENCY cycles. |
| ast_snps_axi5_wlast_eventually | MASTER | WARNING | Recommendation to avoid deadlock. | If WVALID is asserted without WLAST, corresponding WLAST should be eventually asserted. |
| ast_snps_axi5_aw_w_eventually | MASTER | WARNING | Recommendation to avoid deadlock. | If AWVALID is asserted, corresponding WVALID should be eventually asserted. |
| ast_snps_axi5_w_aw_eventually | MASTER | WARNING | Recommendation to avoid deadlock. | If WVALID is asserted, corresponding AWVALID should be eventually asserted. |
| ast_snps_axi5_aww_b_eventually | SLAVE | WARNING | Recommendation to avoid deadlock. | If both write address and write data completed, corresponding BVALID should be eventually asserted. |
| ast_snps_axi5_ar_r_eventually | SLAVE | WARNING | Recommendation to avoid deadlock. | If ARVALID is asserted, corresponding RVALID should be eventually asserted. |
| ast_snps_axi5_rlast_eventually | SLAVE | WARNING | Recommendation to avoid deadlock. | If RVALID is asserted without RLAST, corresponding RLAST should be eventually asserted. |
| ast_snps_axi5_awatop_stable | MASTER | ERROR | [ARM IHI 0022F.b] section A3.2.1, and figure A3-2, on page A3-41. | Once AWVALID is asserted AWATOP must remain asserted until the handshake occurs, at a rising clock edge at which AWVALID and AWREADY are both asserted. |

**Table 4-3 AXI5 Properties**

| Property Name | Master/ Slave | Error Kind | Spec Reference | Property Description |
|---|---|---|---|---|
| ast_snps_axi5_awtrace_stable | MASTER | ERROR | [ARM IHI 0022F.b] section A3.2.1, and figure A3-2, on page A3-41. | Once AWVALID is asserted AWTRACE must remain asserted until the handshake occurs, at a rising clock edge at which AWVALID and AWREADY are both asserted. |
| ast_snps_axi5_awloop_stable | MASTER | ERROR | [ARM IHI 0022F.b] section A3.2.1, and figure A3-2, on page A3-41. | Once AWVALID is asserted AWLOOP must remain asserted until the handshake occurs, at a rising clock edge at which AWVALID and AWREADY are both asserted. |
| ast_snps_axi5_awmmusecsid_stable | MASTER | ERROR | [ARM IHI 0022F.b] section A3.2.1, and figure A3-2, on page A3-41. | Once AWVALID is asserted AWMMUSECSID must remain asserted until the handshake occurs, at a rising clock edge at which AWVALID and AWREADY are both asserted. |
| ast_snps_axi5_awmmusid_stable | MASTER | ERROR | [ARM IHI 0022F.b] section A3.2.1, and figure A3-2, on page A3-41. | Once AWVALID is asserted AWMMUSID must remain asserted until the handshake occurs, at a rising clock edge at which AWVALID and AWREADY are both asserted. |
| ast_snps_axi5_awmmussidv_stable | MASTER | ERROR | [ARM IHI 0022F.b] section A3.2.1, and figure A3-2, on page A3-41. | Once AWVALID is asserted AWMMUSSIDV must remain asserted until the handshake occurs, at a rising clock edge at which AWVALID and AWREADY are both asserted. |
| ast_snps_axi5_awmmussid_stable | MASTER | ERROR | [ARM IHI 0022F.b] section A3.2.1, and figure A3-2, on page A3-41. | Once AWVALID is asserted AWMMUSSID must remain asserted until the handshake occurs, at a rising clock edge at which AWVALID and AWREADY are both asserted. |

**Table 4-3 AXI5 Properties**

| Property Name | Master/ Slave | Error Kind | Spec Reference | Property Description |
|---|---|---|---|---|
| ast_snps_axi5_awmmuatst_stable | MASTER | ERROR | [ARM IHI 0022F.b] section A3.2.1, and figure A3-2, on page A3-41. | Once AWVALID is asserted AWMMUATST must remain asserted until the handshake occurs, at a rising clock edge at which AWVALID and AWREADY are both asserted. |
| ast_snps_axi5_awnsaid_stable | MASTER | ERROR | [ARM IHI 0022F.b] section A3.2.1, and figure A3-2, on page A3-41. | Once AWVALID is asserted AWNSAID must remain asserted until the handshake occurs, at a rising clock edge at which AWVALID and AWREADY are both asserted. |
| ast_snps_axi5_awmmussid_inactive | MASTER | ERROR | [ARM IHI 0022F.b] section E2.12.1 on page E2-371. | For transactions that do not specify a substream ID, as indicated by AWMMUSSIDV deasserted, AWMMUSSID must be driven to all zeros. |
| ast_snps_axi5_awmmusecsid_inactiv e | MASTER | ERROR | [ARM IHI 0022F.b] section E2.12.1 on page E2-371. | For transactions that have already undergone a translation, as indicated by AWMMUATST asserted, AWMMUSECSID must be Low. Secure translated transactions are not supported. |
| ast_snps_axi5_awmmussidv_inactive | MASTER | ERROR | [ARM IHI 0022F.b] section E2.12.1 on page E2-371. | For transactions that have already undergone a translation, as indicated by AWMMUATST asserted, AWMMUSSIDV must be Low. Secure translated transactions are not supported. |
| ast_snps_axi5_awmmusecsid_awpro t | MASTER | ERROR | [ARM IHI 0022F.b] section E2.12.1 on page E2-371. | For transactions that are in a Non-secure stream, as indicated by AWMMUSECSID deasserted, AWPROT[1] must be HIGH. Indicates a Non-secure transactions. |

**Table 4-3 AXI5 Properties**

| Property Name | Master/ Slave | Error Kind | Spec Reference | Property Description |
|---|---|---|---|---|
| ast_snps_axi5_awatop_x | MASTER | ERROR | Signal Integrity | When AWVALID is high, a value of X/Z on AWATOP is not permitted. |
| ast_snps_axi5_awtrace_x | MASTER | ERROR | Signal Integrity | When AWVALID is high, a value of X/Z on AWTRACE is not permitted. |
| ast_snps_axi5_awloop_x | MASTER | ERROR | Signal Integrity | When AWVALID is high, a value of X/Z on AWLOOP is not permitted. |
| ast_snps_axi5_awmmusecsid_x | MASTER | ERROR | Signal Integrity | When AWVALID is high, a value of X/Z on AWMMUSECSID is not permitted. |
| ast_snps_axi5_awmmusid_x | MASTER | ERROR | Signal Integrity | When AWVALID is high, a value of X/Z on AWMMUSID is not permitted. |
| ast_snps_axi5_awmmussidv_x | MASTER | ERROR | Signal Integrity | When AWVALID is high, a value of X/Z on AWMMUSSIDV is not permitted. |
| ast_snps_axi5_awmmussid_x | MASTER | ERROR | Signal Integrity | When AWVALID is high, a value of X/Z on AWMMUSSID is not permitted. |
| ast_snps_axi5_awmmuatst_x | MASTER | ERROR | Signal Integrity | When AWVALID is high, a value of X/Z on AWMMUATST is not permitted. |
| ast_snps_axi5_awnsaid_x | MASTER | ERROR | Signal Integrity | When AWVALID is high, a value of X/Z on AWNSAID is not permitted. |
| ast_snps_axi5_artrace_stable | MASTER | ERROR | [ARM IHI 0022F.b] section A3.2.1, and figure A3-2, on page A3-41. | Once ARVALID is asserted ARTRACE must remain asserted until the handshake occurs, at a rising clock edge at which ARVALID and ARREADY are both asserted. |

**Table 4-3 AXI5 Properties**

| Property Name | Master/ Slave | Error Kind | Spec Reference | Property Description |
|---|---|---|---|---|
| ast_snps_axi5_arloop_stable | MASTER | ERROR | [ARM IHI 0022F.b] section A3.2.1, and figure A3-2, on page A3-41. | Once ARVALID is asserted ARLOOP must remain asserted until the handshake occurs, at a rising clock edge at which ARVALID and ARREADY are both asserted. |
| ast_snps_axi5_armmusecsid_stable | MASTER | ERROR | [ARM IHI 0022F.b] section A3.2.1, and figure A3-2, on page A3-41. | Once ARVALID is asserted ARMMUSECSID must remain asserted until the handshake occurs, at a rising clock edge at which ARVALID and ARREADY are both asserted. |
| ast_snps_axi5_armmusid_stable | MASTER | ERROR | [ARM IHI 0022F.b] section A3.2.1, and figure A3-2, on page A3-41. | Once ARVALID is asserted ARMMUSID must remain asserted until the handshake occurs, at a rising clock edge at which ARVALID and ARREADY are both asserted. |
| ast_snps_axi5_armmussidv_stable | MASTER | ERROR | [ARM IHI 0022F.b] section A3.2.1, and figure A3-2, on page A3-41. | Once ARVALID is asserted ARMMUSSIDV must remain asserted until the handshake occurs, at a rising clock edge at which ARVALID and ARREADY are both asserted. |
| ast_snps_axi5_armmussid_stable | MASTER | ERROR | [ARM IHI 0022F.b] section A3.2.1, and figure A3-2, on page A3-41. | Once ARVALID is asserted ARMMUSSID must remain asserted until the handshake occurs, at a rising clock edge at which ARVALID and ARREADY are both asserted" |
| ast_snps_axi5_armmuatst_stable | MASTER | ERROR | [ARM IHI 0022F.b] section A3.2.1, and figure A3-2, on page A3-41. | Once ARVALID is asserted ARMMUATST must remain asserted until the handshake occurs, at a rising clock edge at which ARVALID and ARREADY are both asserted. |

Table 4-3 AXI5 Properties

| Property Name | Master/ Slave | Error Kind | Spec Reference | Property Description |
|---|---|---|---|---|
| ast_snps_axi5_arnsaid_stable | MASTER | ERROR | [ARM IHI 0022F.b] section A3.2.1, and figure A3-2, on page A3-41. | Once ARVALID is asserted ARNSAID must remain asserted until the handshake occurs, at a rising clock edge at which ARVALID and ARREADY are both asserted. |
| ast_snps_axi5_armmussid_inactive | MASTER | ERROR | [ARM IHI 0022F.b] section E2.12.1 on page E2-371. | For transactions that do not specify a substream ID, as indicated by ARMMUSSIDV deasserted, ARMMUSSID must be driven to all zeros. |
| ast_snps_axi5_armmusecsid_inactive | MASTER | ERROR | [ARM IHI 0022F.b] section E2.12.1 on page E2-371. | For transactions that have already undergone a translation, as indicated by ARMMUATST asserted, ARMMUSECSID must be Low. Secure translated transactions are not supported. |
| ast_snps_axi5_armmussidv_inactive | MASTER | ERROR | [ARM IHI 0022F.b] section E2.12.1 on page E2-371. | For transactions that have already undergone a translation, as indicated by ARMMUATST asserted, ARMMUSSIDV must be Low. Secure translated transactions are not supported. |
| ast_snps_axi5_armmusecsid_arprot | MASTER | ERROR | [ARM IHI 0022F.b] section E2.12.1 on page E2-371. | For transactions that are in a Non-secure stream, as indicated by ARMMUSECSID deasserted, ARPROT[1] must be HIGH. Indicates a Non-secure transactions. |
| ast_snps_axi5_artrace_x | MASTER | ERROR | Signal Integrity | When ARVALID is high, a value of X/Z on ARTRACE is not permitted. |
| ast_snps_axi5_arloop_x | MASTER | ERROR | Signal Integrity | When ARVALID is high, a value of X/Z on ARLOOP is not permitted. |

**Table 4-3 AXI5 Properties**

| Property Name | Master/ Slave | Error Kind | Spec Reference | Property Description |
|---|---|---|---|---|
| ast_snps_axi5_armmusecsid_x | MASTER | ERROR | Signal Integrity | When ARVALID is high, a value of X/Z on ARMMUSECSID is not permitted. |
| ast_snps_axi5_armmusid_x | MASTER | ERROR | Signal Integrity | When ARVALID is high, a value of X/Z on ARMMUSID is not permitted. |
| ast_snps_axi5_armmussidv_x | MASTER | ERROR | Signal Integrity | When ARVALID is high, a value of X/Z on ARMMUSSIDV is not permitted. |
| ast_snps_axi5_armmussid_x | MASTER | ERROR | Signal Integrity | When ARVALID is high, a value of X/Z on ARMMUSSID is not permitted. |
| ast_snps_axi5_armmuatst_x | MASTER | ERROR | Signal Integrity | When ARVALID is high, a value of X/Z on ARMMUATST is not permitted. |
| ast_snps_axi5_arnsaid_x | MASTER | ERROR | Signal Integrity | When ARVALID is high, a value of X/Z on ARNSAID is not permitted. |
| ast_snps_axi5_wtrace_stable | MASTER | ERROR | [ARM IHI 0022F.b] section A3.2.1, and figure A3-2, on page A3-41. | Once WVALID is asserted WTRACE must remain asserted until the handshake occurs, at a rising clock edge at which WVALID and WREADY are both asserted. |
| ast_snps_axi5_wdatachk_stable | MASTER | ERROR | [ARM IHI 0022F.b] section A3.2.1, and figure A3-2, on page A3-41. | Once WVALID is asserted WDATACHK must remain asserted until the handshake occurs, at a rising clock edge at which WVALID and WREADY are both asserted. |
| ast_snps_axi5_wpoison_stable | MASTER | ERROR | [ARM IHI 0022F.b] section A3.2.1, and figure A3-2, on page A3-41. | Once WVALID is asserted WPOISON must remain asserted until the handshake occurs, at a rising clock edge at which WVALID and WREADY are both asserted. |

**Table 4-3 AXI5 Properties**

| Property Name | Master/ Slave | Error Kind | Spec Reference | Property Description |
|---|---|---|---|---|
| ast_snps_axi5_btrace_stable | SLAVE | ERROR | [ARM IHI 0022F.b] section A3.2.1, and figure A3-2, on page A3-41. | Once BVALID is asserted BTRACE must remain asserted until the handshake occurs, at a rising clock edge at which BVALID and BREADY are both asserted. |
| ast_snps_axi5_bloop_stable | SLAVE | ERROR | [ARM IHI 0022F.b] section A3.2.1, and figure A3-2, on page A3-41. | Once BVALID is asserted BLOOP must remain asserted until the handshake occurs, at a rising clock edge at which BVALID and BREADY are both asserted. |
| ast_snps_axi5_rtrace_stable | SLAVE | ERROR | [ARM IHI 0022F.b] section A3.2.1, and figure A3-2, on page A3-41. | Once RVALID is asserted RTRACE must remain asserted until the handshake occurs, at a rising clock edge at which RVALID and RREADY are both asserted. |
| ast_snps_axi5_rloop_stable | SLAVE | ERROR | [ARM IHI 0022F.b] section A3.2.1, and figure A3-2, on page A3-41. | Once RVALID is asserted RLOOP must remain asserted until the handshake occurs, at a rising clock edge at which RVALID and RREADY are both asserted. |
| ast_snps_axi5_rdatachk_stable | SLAVE | ERROR | [ARM IHI 0022F.b] section A3.2.1, and figure A3-2, on page A3-41. | Once RVALID is asserted RDATACHK must remain asserted until the handshake occurs, at a rising clock edge at which RVALID and RREADY are both asserted. |
| ast_snps_axi5_rpoison_stable | SLAVE | ERROR | [ARM IHI 0022F.b] section A3.2.1, and figure A3-2, on page A3-41. | Once RVALID is asserted RPOISON must remain asserted until the handshake occurs, at a rising clock edge at which RVALID and RREADY are both asserted. |

**Table 4-3 AXI5 Properties**

| Property Name | Master/ Slave | Error Kind | Spec Reference | Property Description |
|---|---|---|---|---|
| ast_snps_axi5_wtrace_x | MASTER | ERROR | Signal Integrity | When WVALID is high, a value of X/Z on WTRACE is not permitted. |
| ast_snps_axi5_wdatachk_x | MASTER | ERROR | Signal Integrity | When WVALID is high, a value of X/Z on WDATACHK is not permitted. |
| ast_snps_axi5_wpoison_x | MASTER | ERROR | Signal Integrity | When WVALID is high, a value of X/Z on WPOISON is not permitted. |
| ast_snps_axi5_wdatachk_odd_parity | MASTER | ERROR | [ARM IHI 0022F.b] section E2.5.1 on page E2-352. | This specification supports data checking using odd parity at a byte granularity. |
| ast_snps_axi5_wdata_invalid_lanes | MASTER | ERROR | [ARM IHI 0022F.b] section E2.5.3 on page E2-353. | If data checking signaling is supported, it is required that all invalid data lanes are driven to zero. |
| ast_snps_axi5_btrace_x | SLAVE | ERROR | Signal Integrity | When BVALID is high, a value of X/Z on BTRACE is not permitted. |
| ast_snps_axi5_bloop_x | SLAVE | ERROR | Signal Integrity | When BVALID is high, a value of X/Z on BLOOP is not permitted. |
| ast_snps_axi5_rtrace_x | SLAVE | ERROR | Signal Integrity | When RVALID is high, a value of X/Z on RTRACE is not permitted. |
| ast_snps_axi5_rloop_x | SLAVE | ERROR | Signal Integrity | When RVALID is high, a value of X/Z on RLOOP is not permitted. |
| ast_snps_axi5_rdatachk_x | SLAVE | ERROR | Signal Integrity | When RVALID is high, a value of X/Z on RDATACHK is not permitted. |
| ast_snps_axi5_rpoison_x | SLAVE | ERROR | Signal Integrity | When RVALID is high, a value of X/Z on RPOISON is not permitted. |
| ast_snps_axi5_btrace_value | SLAVE | ERROR | [ARM IHI 0022F.b] section E2.6 on page E2-356. | A slave that receives a write request with AWTRACE asserted should assert the BTRACE signal alongside the write response. |

**Table 4-3 AXI5 Properties**

| Property Name | Master/ Slave | Error Kind | Spec Reference | Property Description |
|---|---|---|---|---|
| ast_snps_axi5_bloop_value | SLAVE | ERROR | [ARM IHI 0022F.b] section E2.7 on page E2-357. | The value of BLOOP must be identical to the value that was presented on the AWLOOP signal. |
| ast_snps_axi5_rdatachk_odd_parity | SLAVE | ERROR | [ARM IHI 0022F.b] section E2.5.1 on page E2-352. | This specification supports data checking using odd parity at a byte granularity. |
| ast_snps_axi5_rdata_invalid_lanes | SLAVE | ERROR | [ARM IHI 0022F.b] section E2.5.3 on page E2-353. | If data checking signaling is supported, it is required that all invalid data lanes are driven to zero. |
| ast_snps_axi5_rtrace_value | SLAVE | ERROR | [ARM IHI 0022F.b] section E2.6 on page E2-356. | A slave that receives a read request with ARTRACE signal asserted should assert the RTRACE signal alongside every beat of the read response. |
| ast_snps_axi5_rloop_value | SLAVE | ERROR | [ARM IHI 0022F.b] section E2.7 on page E2-357. | The value of RLOOP must be identical to the value that was presented on the ARLOOP signal. |
| ast_snps_axi5_awakeup_fall | MASTER | ERROR | [ARM IHI 0022F.b] section E2.9.1 on page E2-360. | To ensure progress of the transaction, AWAKEUP must remain asserted until the associated ARVALID, ARREADY handshake, or the AWVALID, AWREADY handshake. |
| ast_snps_axi5_awakeup_rise | MASTER | WARNING | [ARM IHI 0022F.b] section E2.9.1 on page E2-360. | The specification recommends that AWAKEUP is asserted at least one cycle before the assertion of ARVALID, AWVALID, or WVALID to prevent the acceptance of a new transaction being delayed. |
| ast_snps_axi5_awakeup_x | MASTER | ERROR | Signal Integrity | When not in reset, a value of X/Z on AWAKEUP is not permitted. |

**Table 4-3 AXI5 Properties**

| Property Name | Master/ Slave | Error Kind | Spec Reference | Property Description |
|---|---|---|---|---|
| p_snps_axi_rresp_consistent_decerr | SLAVE | ERROR | [ARM IHI 0022H.c] section E1.18.4 on page E1-401. | DECERR is signaled for every beat of read data, or no beats of read data within each cache line of data. |
| p_snps_axi_atomic_awatop_legal | MASTER | ERROR | [ARM IHI 0022H.c] section E1.1.6 Table E1-2 on page E1-345. | Legal value for AWATOP is listed on Table E1-2. |
| p_snps_axi_atomic_awsize_multi_beats | MASTER | ERROR | [ARM IHI 0022H.c] section E1.1.3 on page E1-343. | If AWLEN indicates a burst length greater than one, AWSIZE is required to be the full data bus width. |
| p_snps_axi_atomic_wstrb_exact | MASTER | ERROR | [ARM IHI 0022H.c] section E1.1.3 on page E1-343. | Write strobes that are not within the data window, as specified by AWADDR and AWSIZE, must be deasserted. Write Strobes within the data window must be asserted. |
| p_snps_axi_atomic_awid_unique | MASTER | ERROR | [ARM IHI 0022H.c] section E1.1.4 on page E1-344 and E1-345. | Atomic transactions must not use AXI ID values that are used by Non-atomic transactions that are outstanding at the same time. |
| p_snps_axi_awid_non_atomic | MASTER | ERROR | [ARM IHI 0022H.c] section E1.1.4 on page E1-344 and E1-345. | Atomic transactions must not use AXI ID values that are used by Non-atomic transactions that are outstanding at the same time. |
| p_snps_axi_atomic_awlock_normal | MASTER | ERROR | [ARM IHI 0022H.c] section E1.1.5 on page E1-345. | For Atomic transactions, AWLOCK must be 0b0, Normal access. |
| p_snps_axi_atomic_stldsw_write_bytes | MASTER | ERROR | [ARM IHI 0022H.c] section E1.1.3 on page E1-343. | For AtomicStore, AtomicLoad, and AtomicSwap: The write data is 1, 2, 4, or 8 bytes. |
| p_snps_axi_atomic_stldsw_awaddr_align | MASTER | ERROR | [ARM IHI 0022H.c] section E1.1.3 on page E1-343. | For AtomicStore, AtomicLoad, and AtomicSwap: AWADDR must be aligned to the data size. |

**Table 4-3 AXI5 Properties**

| Property Name | Master/ Slave | Error Kind | Spec Reference | Property Description |
|---|---|---|---|---|
| p_snps_axi_atomic_stldsw_awburst_incr | MASTER | ERROR | [ARM IHI 0022H.c] section E1.1.3 on page E1-343. | For AtomicStore, AtomicLoad, and AtomicSwap: AWADDR must be INCR. |
| p_snps_axi_atomic_cmp_write_bytes | MASTER | ERROR | [ARM IHI 0022H.c] section E1.1.3 on page E1-344. | For AtomicCompare: The write data is 2, 4, 8, 16, or 32 bytes. |
| p_snps_axi_atomic_cmp_awaddr_half_align | MASTER | ERROR | [ARM IHI 0022H.c] section E1.1.3 on page E1-344. | For AtomicCompare: AWADDR must be aligned to a single write data value, half the total write data size. |
| p_snps_axi_atomic_cmp_awaddr_lower | MASTER | ERROR | [ARM IHI 0022H.c] section E1.1.3 on page E1-344. | For AtomicCompare: If AWADDR points to the lower half of the transaction: AWBURST must be INCR. |
| p_snps_axi_atomic_cmp_awaddr_upper | MASTER | ERROR | [ARM IHI 0022H.c] section E1.1.3 on page E1-344. | For AtomicCompare: If AWADDR points to the upper half of the transaction: AWBURST must be WRAP. |
| p_snps_axi_arid_non_atomic | MASTER | ERROR | [ARM IHI 0022H.c] section E1.1.4 on page E1-344 and E1-345. | Atomic transactions must not use AXI ID values that are used by Non-atomic transactions that are outstanding at the same time. |
| p_snps_axi_rchunk_arsize_multi_beats | MASTER | ERROR | [ARM IHI 0022H.c] section E1.11.1 on page E1-378. | When ARCHUNKEN is asserted, ARSIZE is equal to the data bus with or ARLEN is one beat. |
| p_snps_axi_rchunk_arsize_legal | MASTER | ERROR | [ARM IHI 0022H.c] section E1.11.1 on page E1-378. | When ARCHUNKEN is asserted, ARSIZE is 128 bits or larger. |
| p_snps_axi_rchunk_araddr_aligned | MASTER | ERROR | [ARM IHI 0022H.c] section E1.11.1 on page E1-378. | When ARCHUNKEN is asserted, ARADDR is aligned to 16 bytes. |
| p_snps_axi_rchunk_arburst_legal | MASTER | ERROR | [ARM IHI 0022H.c] section E1.11.1 on page E1-379. | When ARCHUNKEN is asserted, ARBURST is INCR or WRAP. |

**Table 4-3 AXI5 Properties**

| Property Name | Master/ Slave | Error Kind | Spec Reference | Property Description |
|---|---|---|---|---|
| p_snps_axi_rchunk_arid_unique | MASTER | ERROR | [ARM IHI 0022H.c] section E1.11.1 on page E1-379. | ARCHUNKEN can only be asserted if there are no outstanding read transactions using the same ARID value. |
| p_snps_axi_rchunk_aridunq | MASTER | ERROR | [ARM IHI 0022H.c] section E1.11.1 on page E1-379. | When ARCHUNKEN is asserted, if present on the interface, ARIDUNQ must be asserted. |
| p_snps_axi_rchunk_rchunkv_allowed | SLAVE | ERROR | [ARM IHI 0022H.c] section E1.11.1 on page E1-379. | If ARCHUNKEN is deasserted, RCHUNKV must be deasserted for all response beats of the transaction. If ARCHUNKEN is asserted, RCHUNKV can be asserted for response beats of the transaction. |
| p_snps_axi_rchunk_rchunkv_consistent | SLAVE | ERROR | [ARM IHI 0022H.c] section E1.11.1 on page E1-379. | RCHUNKV must be the same for every response beat of a transaction. |
| p_snps_axi_rchunk_rchunknum_legal | SLAVE | ERROR | [ARM IHI 0022H.c] section E1.11.1 on page E1-379. | When RVALID and RCHUNKV are asserted, RCHUNKNUM must be between zero and ARLEN. |
| p_snps_axi_rchunk_rchunkstrb_legal | SLAVE | ERROR | [ARM IHI 0022H.c] section E1.11.1 on page E1-379. | When RVALID and RCHUNKV are asserted, RCHUNKSTRB must not be zero. |
| p_snps_axi_rchunk_rchunknum_rchunkstrb_complete | SLAVE | ERROR | [ARM IHI 0022H.c] section E1.11.1 on page E1-379. | The number of data chunks transferred must be consistent with ARLEN and ARSIZE, the number of bytes transferred in a burst is the same whether chunking is enabled or not. For unaligned transactions, chunks at addresses lower than ARADDR are not transferred. |

**Table 4-3 AXI5 Properties**

| Property Name | Master/ Slave | Error Kind | Spec Reference | Property Description |
|---|---|---|---|---|
| p_snps_axi_aridunq_valid | MASTER | ERROR | [ARM IHI 0022H.c] section E1.13 on page E1-383. | The unique ID indicator indicates when a request on the read address channel is using an AXI identifier that is unique for in-flight transactions. |
| p_snps_axi_ridunq_valid | SLAVE | ERROR | [ARM IHI 0022H.c] section E1.13 on page E1-383. | The unique ID on the read response channel indicates that a transaction is using a unique ID. |
| p_snps_axi_awidunq_valid | MASTER | ERROR | [ARM IHI 0022H.c] section E1.13 on page E1-383. | The unique ID indicator indicates when a request on the write address channel is using an AXI identifier that is unique for in-flight transactions. |
| p_snps_axi_bidunq_valid | SLAVE | ERROR | [ARM IHI 0022H.c] section E1.13 on page E1-383. | The unique ID on the write response channel indicates that a transaction is using a unique ID. |
| p_snps_axi_read_max_transaction_bytes | MASTER | ERROR | [ARM IHI 0022H.c] section E1.18.3 on page E1-400. | The property Max_Transaction_Bytes defines the maximum size of a transaction in bytes. |
| p_snps_axi_write_max_transaction_bytes | MASTER | ERROR | [ARM IHI 0022H.c] section E1.18.3 on page E1-400. | The property Max_Transaction_Bytes defines the maximum size of a transaction in bytes. |
| p_snps_axi_arlen_regular_txn | MASTER | ERROR | [ARM IHI 0022H.c] section A3.4.3 on page A3-53. | The Regular attribute is defined, to identify transactions which meet the following criteria: ARLEN is 1, 2, 4, 8, or 16. |
| p_snps_axi_arsize_regular_txn | MASTER | ERROR | [ARM IHI 0022H.c] section A3.4.3 on page A3-53. | The Regular attribute is defined, to identify transactions which meet the following criteria: ARSIZE is the same as the data bus width, if ARLEN is greater than 1. |

**Table 4-3 AXI5 Properties**

| Property Name | Master/ Slave | Error Kind | Spec Reference | Property Description |
|---|---|---|---|---|
| p_snps_axi_arburst_regular_txn | MASTER | ERROR | [ARM IHI 0022H.c] section A3.4.3 on page A3-53. | The Regular attribute is defined, to identify transactions which meet the following criteria: ARBURST is INCR or WRAP, not FIXED. |
| p_snps_axi_araddr_regular_txn_incr | MASTER | ERROR | [ARM IHI 0022H.c] section A3.4.3 on page A3-53. | The Regular attribute is defined, to identify transactions which meet the following criteria: ARADDR is aligned to the transaction container for INCR transactions. |
| p_snps_axi_awlen_regular_txn | MASTER | ERROR | [ARM IHI 0022H.c] section A3.4.3 on page A3-53. | The Regular attribute is defined, to identify transactions which meet the following criteria: AWLEN is 1, 2, 4, 8, or 16. |
| p_snps_axi_awsize_regular_txn | MASTER | ERROR | [ARM IHI 0022H.c] section A3.4.3 on page A3-53. | The Regular attribute is defined, to identify transactions which meet the following criteria: AWSIZE is the same as the data bus width, if AWLEN is greater than 1. |
| p_snps_axi_awburst_regular_txn | MASTER | ERROR | [ARM IHI 0022H.c] section A3.4.3 on page A3-53. | The Regular attribute is defined, to identify transactions which meet the following criteria: AWBURST is INCR or WRAP, not FIXED. |
| p_snps_axi_awaddr_regular_txn_incr | MASTER | ERROR | [ARM IHI 0022H.c] section A3.4.3 on page A3-53. | The Regular attribute is defined, to identify transactions which meet the following criteria: AWADDR is aligned to the transaction container for INCR transactions. |

Table 4-3 **AXI5 Properties**

| **Property Name** | **Master/ Slave** | **Error Kind** | **Spec Reference** | **Property Description** |
|---|---|---|---|---|
| p_snps_axi_archunken_stable | MASTER | ERROR | [ARM IHI 0022H.c] section A3.2.1, and figure A3-2, on page A3-41. | Once ARVALID is asserted ARCHUNKEN must remain asserted until the handshake occurs, at a rising clock edge at which ARVALID and ARREADY are both asserted. |
| p_snps_axi_rchunkv_stable | SLAVE | ERROR | [ARM IHI 0022H.c] section A3.2.1, and figure A3-2, on page A3-41. | Once RVALID is asserted RCHUNKV must remain asserted until the handshake occurs, at a rising clock edge at which RVALID and RREADY are both asserted. |
| p_snps_axi_rchunknum_stable | SLAVE | ERROR | [ARM IHI 0022H.c] section A3.2.1, and figure A3-2, on page A3-41. | Once RVALID is asserted RCHUNNUM must remain asserted until the handshake occurs, at a rising clock edge at which RVALID and RREADY are both asserted. |
| p_snps_axi_rchunkstrb_stable | SLAVE | ERROR | [ARM IHI 0022H.c] section A3.2.1, and figure A3-2, on page A3-41. | Once RVALID is asserted RCHUNKSTRB must remain asserted until the handshake occurs, at a rising clock edge at which RVALID and RREADY are both asserted. |
| p_snps_axi_aridunq_stable | MASTER | ERROR | [ARM IHI 0022H.c] section A3.2.1, and figure A3-2, on page A3-41. | Once ARVALID is asserted ARIDUNQ must remain asserted until the handshake occurs, at a rising clock edge at which ARVALID and ARREADY are both asserted. |
| p_snps_axi_ridunq_stable | SLAVE | ERROR | [ARM IHI 0022H.c] section A3.2.1, and figure A3-2, on page A3-41. | Once RVALID is asserted RIDUNQ must remain asserted until the handshake occurs, at a rising clock edge at which RVALID and RREADY are both asserted. |

**Table 4-3 AXI5 Properties**

| Property Name | Master/ Slave | Error Kind | Spec Reference | Property Description |
|---|---|---|---|---|
| p_snps_axi_awidunq_stable | MASTER | ERROR | [ARM IHI 0022H.c] section A3.2.1, and figure A3-2, on page A3-41. | Once AWVALID is asserted AWIDUNQ must remain asserted until the handshake occurs, at a rising clock edge at which AWVALID and AWREADY are both asserted. |
| p_snps_axi_bidunq_stable | SLAVE | ERROR | [ARM IHI 0022H.c] section A3.2.1, and figure A3-2, on page A3-41. | Once BVALID is asserted BIDUNQ must remain asserted until the handshake occurs, at a rising clock edge at which BVALID and BREADY are both asserted. |
| p_snps_axi_armpam_stable | MASTER | ERROR | [ARM IHI 0022H.c] section A3.2.1, and figure A3-2, on page A3-41. | Once ARVALID is asserted ARIDUNQ must remain asserted until the handshake occurs, at a rising clock edge at which ARVALID and ARREADY are both asserted. |
| p_snps_axi_awmpam_stable | MASTER | ERROR | [ARM IHI 0022H.c] section A3.2.1, and figure A3-2, on page A3-41. | Once AWVALID is asserted AWMPAM must remain asserted until the handshake occurs, at a rising clock edge at which AWVALID and AWREADY are both asserted. |
| p_snps_axi_awmmuflow_stable | MASTER | ERROR | [ARM IHI 0022H.c] section A3.2.1, and figure A3-2, on page A3-41. | Once AWVALID is asserted AWMMUFLOW must remain asserted until the handshake occurs, at a rising clock edge at which AWVALID and AWREADY are both asserted. |
| p_snps_axi_awtagop_stable | MASTER | ERROR | [ARM IHI 0022H.c] section A3.2.1, and figure A3-2, on page A3-41. | Once AWVALID is asserted AWTAGOP must remain asserted until the handshake occurs, at a rising clock edge at which AWVALID and AWREADY are both asserted. |

Table 4-3 **AXI5 Properties**

| Property Name | Master/ Slave | Error Kind | Spec Reference | Property Description |
|---|---|---|---|---|
| p_snps_axi_wtag_stable | MASTER | ERROR | [ARM IHI 0022H.c] section A3.2.1, and figure A3-2, on page A3-41. | Once WVALID is asserted WTAG must remain asserted until the handshake occurs, at a rising clock edge at which WVALID and WREADY are both asserted. |
| p_snps_axi_wtagupdate_stable | MASTER | ERROR | [ARM IHI 0022H.c] section A3.2.1, and figure A3-2, on page A3-41. | Once WVALID is asserted WTAGUPDATE must remain asserted until the handshake occurs, at a rising clock edge at which WVALID and WREADY are both asserted. |
| p_snps_axi_btagmatch_stable | SLAVE | ERROR | [ARM IHI 0022H.c] section A3.2.1, and figure A3-2, on page A3-41. | Once BVALID is asserted BTAGMATCH must remain asserted until the handshake occurs, at a rising clock edge at which BVALID and BREADY are both asserted. |
| p_snps_axi_bcomp_stable | SLAVE | ERROR | [ARM IHI 0022H.c] section A3.2.1, and figure A3-2, on page A3-41. | Once BVALID is asserted BCOMP must remain asserted until the handshake occurs, at a rising clock edge at which BVALID and BREADY are both asserted. |
| p_snps_axi_armmuflow_stable | MASTER | ERROR | [ARM IHI 0022H.c] section A3.2.1, and figure A3-2, on page A3-41. | Once ARVALID is asserted ARMMUFLOW must remain asserted until the handshake occurs, at a rising clock edge at which ARVALID and ARREADY are both asserted. |
| p_snps_axi_artagop_stable | MASTER | ERROR | [ARM IHI 0022H.c] section A3.2.1, and figure A3-2, on page A3-41. | Once ARVALID is asserted ARTAGOP must remain asserted until the handshake occurs, at a rising clock edge at which ARVALID and ARREADY are both asserted. |

**Table 4-3 AXI5 Properties**

| Property Name | Master/ Slave | Error Kind | Spec Reference | Property Description |
|---|---|---|---|---|
| p_snps_axi_rtag_stable | MASTER | ERROR | [ARM IHI 0022H.c] section A3.2.1, and figure A3-2, on page A3-41. | Once RVALID is asserted RTAG must remain asserted until the handshake occurs, at a rising clock edge at which RVALID and RREADY are both asserted. |
| p_snps_axi_archunken_x | MASTER | ERROR | Signal Integrity | When ARVALID is high, a value of X/Z on ARCHUNKEN is not permitted. |
| p_snps_axi_rchunkv_x | SLAVE | ERROR | Signal Integrity | When RVALID is high, a value of X/Z on RCHUNKV is not permitted. |
| p_snps_axi_rchunknum_x | SLAVE | ERROR | Signal Integrity | When RVALID is high, a value of X/Z on RCHUNKNUM is not permitted. |
| p_snps_axi_rchunkstrb_x | SLAVE | ERROR | Signal Integrity | When RVALID is high, a value of X/Z on RCHUNKSTRB is not permitted. |
| p_snps_axi_awidunq_x | MASTER | ERROR | Signal Integrity | When AWVALID is high, a value of X/Z on AWIDUNQ is not permitted. |
| p_snps_axi_bidunq_x | SLAVE | ERROR | Signal Integrity | When BVALID is high, a value of X/Z on BIDUNQ is not permitted. |
| p_snps_axi_aridunq_x | MASTER | ERROR | Signal Integrity | When ARVALID is high, a value of X/Z on ARIDUNQ is not permitted. |
| p_snps_axi_ridunq_x | SLAVE | ERROR | Signal Integrity | When RVALID is high, a value of X/Z on RIDUNQ is not permitted. |
| p_snps_axi_armpam_x | MASTER | ERROR | Signal Integrity | When ARVALID is high, a value of X/Z on ARMPAM is not permitted. |
| p_snps_axi_awmpam_x | MASTER | ERROR | Signal Integrity | When AWVALID is high, a value of X/Z on AWMPAM is not permitted. |

**Table 4-3 AXI5 Properties**

| Property Name | Master/ Slave | Error Kind | Spec Reference | Property Description |
|---|---|---|---|---|
| p_snps_axi_awmmuflow_x | MASTER | ERROR | Signal Integrity | When AWVALID is high, a value of X/Z on AWMMUFLOW is not permitted. |
| p_snps_axi_awtagop_x | MASTER | ERROR | Signal Integrity | When AWVALID is high, a value of X/Z on AWTAGOP is not permitted. |
| p_snps_axi_wtag_x | MASTER | ERROR | Signal Integrity | When WVALID is high, a value of X/Z on WTAG is not permitted. |
| p_snps_axi_wtagupdate_x | MASTER | ERROR | Signal Integrity | When WVALID is high, a value of X/Z on WTAGUPDATE is not permitted. |
| p_snps_axi_btagmatch_x | SLAVE | ERROR | Signal Integrity | When BVALID is high, a value of X/Z on BTAGMATCH is not permitted. |
| p_snps_axi_bcomp_x | SLAVE | ERROR | Signal Integrity | When BVALID is high, a value of X/Z on BCOMP is not permitted. |
| p_snps_axi_armmuflow_x | MASTER | ERROR | Signal Integrity | When ARVALID is high, a value of X/Z on ARMMUFLOW is not permitted. |
| p_snps_axi_artagop_x | MASTER | ERROR | Signal Integrity | When ARVALID is high, a value of X/Z on ARTAGOP is not permitted. |
| p_snps_axi_rtag_x | SLAVE | ERROR | Signal Integrity | When RVALID is high, a value of X/Z on RTAG is not permitted. |
| p_snps_axi_mte_write_transaction_size | MASTER | ERROR | [ARM IHI 0022H.c] section E1.15.4 on page E1-389. | The transaction must be cache-line-sized or smaller. |
| p_snps_axi_mte_awburst_legal | MASTER | ERROR | [ARM IHI 0022H.c] section E1.15.4 on page E1-389. | AxBURST must be INCR or WRAP, not FIXED. |
| p_snps_axi_mte_awcache_legal | MASTER | ERROR | [ARM IHI 0022H.c] section E1.15.4 on page E1-390. | AxCACHE[3:2] is not 0b00, AxCACHE[1:0] is 0b11. |

**Table 4-3 AXI5 Properties**

| Property Name | Master/ Slave | Error Kind | Spec Reference | Property Description |
|---|---|---|---|---|
| p_snps_axi_mte_awid_unique | MASTER | ERROR | [ARM IHI 0022H.c] section E1.15.4 on page E1-390. | The ID value must be unique-in-flight. A write with tag Transfer, Update or Match can only be issued if there are no outstanding write transactions using the same AWID value. |
| p_snps_axi_non_mte_awid | MASTER | ERROR | [ARM IHI 0022H.c] section E1.15.4 on page E1-390. | The ID value must be unique-in-flight. A Manager must not issue a request on the write channel with the same AWID as an outstanding write with tag Transfer, Update or Match. |
| p_snps_axi_mte_awidunq | MASTER | ERROR | [ARM IHI 0022H.c] section E1.15.4 on page E1-390. | The ID value must be unique-in-flight. If present, AWIDUNQ must be asserted for a write with tag Transfer, Update or Match. |
| p_snps_axi_mte_basic_awtagop_transfer_match_not_allowed | MASTER | ERROR | [ARM IHI 0022H.c] section E1.15.6 Table E1-25 on page E1-392 and E1-393. | MTE_Support = Basic - Transfer and Match are not used |
| p_snps_axi_mte_transfer_update_no_atomic | MASTER | ERROR | [ARM IHI 0022H.c] section E1.15.7 on page E1-394. | Atomic transactions cannot be used with Transfer or Update operations. |
| p_snps_axi_mte_match_atomic_compare_size | MASTER | ERROR | [ARM IHI 0022H.c] section E1.15.7 on page E1-394. | AtomicCompare transactions with Match can be 16 bytes or 32 bytes. |
| p_snps_axi_mte_wtagupdate_legal | MASTER | ERROR | [ARM IHI 0022H.c] section E1.15.6 Table E1-24 on page E1-392. | When AWTAGOP is Invalid, Transfer or Match, no tag updating or checking is required. WTAGUPDATE must be deasserted. |
| p_snps_axi_mte_wtagupdate_valid | MASTER | ERROR | [ARM IHI 0022H.c] section E1.15.6 Table E1-24 on page E1-392. | AWTAGOP Update: Tags that are partially addressed in the transaction must have WTAGUPDATE deasserted. |
| p_snps_axi_mte_wtag_legal | MASTER | ERROR | [ARM IHI 0022H.c] section E1.15.6 Table E1-24 on page E1-392. | When AWTAGOP is Invalid, no tag updating or checking is required. WTAG must be zero. |

Table 4-3 **AXI5 Properties**

| Property Name | Master/ Slave | Error Kind | Spec Reference | Property Description |
|---|---|---|---|---|
| p_snps_axi_mte_completion_combined_bloop_value | SLAVE | ERROR | [ARM IHI 0022H.c] section E1.15.6 on page E1-393 and E1-394. | If loop back signaling is supported, BLOOP must have the same value as AWLOOP for Completion and Combined response. |
| p_snps_axi_mte_completion_combined_bresp_value | SLAVE | ERROR | [ARM IHI 0022H.c] section E1.15.6 on page E1-393 and E1-394. | BRESP can be OKAY, EXOKAY, SLVERR, or DECERR for Completion or Combined response. |
| p_snps_axi_mte_match_bresp_value | SLAVE | ERROR | [ARM IHI 0022H.c] section E1.15.6 on page E1-393 and E1-394. | BRESP can be OKAY, SLVERR, or DECERR for Match response. |
| p_snps_axi_mte_match_btagmatch_w_bcomp | SLAVE | ERROR | [ARM IHI 0022H.c] section E1.15.6 on page E1-393 and E1-394. | BTAGMATCH must be 0b01 with BCOMP asserted for Completion response or must be 0b11 or 0b10 with BCOMP asserted for Combined response. |
| p_snps_axi_mte_match_btagmatch_wo_bcomp | SLAVE | ERROR | [ARM IHI 0022H.c] section E1.15.6 on page E1-393 and E1-394. | BTAGMATCH must be 0b11 or 0b10 with BCOMP deasserted for Match response. |
| p_snps_axi_mte_artagop_reserved | MASTER | ERROR | [ARM IHI 0022H.c] section E1.15.2 Table E1-21 on page E1-388. | ARTAGOP - 0b10 is Reserved. |
| p_snps_axi_mte_arburst_legal | MASTER | ERROR | [ARM IHI 0022H.c] section E1.15.4 on page E1-389. | AxBURST must be INCR or WRAP, not FIXED. |
| p_snps_axi_mte_arcache_legal | MASTER | ERROR | [ARM IHI 0022H.c] section E1.15.4 on page E1-390. | AxCACHE[3:2] is not 0b00, AxCACHE[1:0] is 0b11. |
| p_snps_axi_mte_arid_unique | MASTER | ERROR | [ARM IHI 0022H.c] section E1.15.4 on page E1-390. | The ID value must be unique-in-flight. A read with tag Transfer or Fetch can only be issued if there are no outstanding read transactions using the same ARID value. |

**Table 4-3 AXI5 Properties**

| Property Name | Master/ Slave | Error Kind | Spec Reference | Property Description |
|---|---|---|---|---|
| p_snps_axi_non_mte_arid | MASTER | ERROR | [ARM IHI 0022H.c] section E1.15.4 on page E1-390. | The ID value must be unique-in-flight. A Manager must not issue a request on the read channel with the same ARID as an outstanding read with tag Transfer or Fetch. |
| p_snps_axi_mte_aridunq | MASTER | ERROR | [ARM IHI 0022H.c] section E1.15.4 on page E1-390. | The ID value must be unique-in-flight. If present, ARIDUNQ must be asserted for a read with tag Transfer or Fetch. |
| p_snps_axi_mte_basic_artagop_fetch_not_allowed | MASTER | ERROR | [ARM IHI 0022H.c] section E1.15.5 Table E1-23 on page E1-391. | MTE_Support = Basic - Fetch is not used. |
| p_snps_axi_mte_standard_artagop_fetch_allowed | MASTER | ERROR | [ARM IHI 0022H.c] section E1.15.5 Table E1-23 on page E1-391. | MTE_Support = Standard - Fetch can be used with ReadNoSnoop. |
| p_snps_axi_mte_rtag_for_invalid | SLAVE | ERROR | [ARM IHI 0022H.c] section E1.15.5 Table E1-22 on page E1-390. | In the response to ARTAGOP Invalid, RTAG is invalid and must be zero. |
| p_snps_axi_bcomp_one_beat_response | SLAVE | ERROR | [ARM IHI 0022H.c] section D7.7.5 on page D7-275. | If BCOMP is present on an interface, it must be asserted for one beat of the write response for all write transactions. |

## 4.4 The AXI5 AIP Cover Properties

**Table 4-4 AXI5 Cover Properties**

| Property Name | Property Description |
|---|---|
| cov_snps_axi5_awburst_fixed | Observed FIXED write burst |
| cov_snps_axi5_awburst_incr | Observed INCR write burst |
| cov_snps_axi5_awburst_wrap | Observed WRAP write burst |
| cov_snps_axi5_awlen_len | Observed write burst with AWLEN = N (N: 0 to MAXBURSTLENGTH-1) |
| cov_snps_axi5_awsize_8_bits | Observed write burst with AWSIZE = 8 bits |
| cov_snps_axi5_awsize_16_bits | Observed write burst with AWSIZE = 16 bits |
| cov_snps_axi5_awsize_32_bits | Observed write burst with AWSIZE = 32 bits |
| cov_snps_axi5_awsize_64_bits | Observed write burst with AWSIZE = 64 bits |
| cov_snps_axi5_awsize_128_bits | Observed write burst with AWSIZE = 128 bits |
| cov_snps_axi5_awsize_256_bits | Observed write burst with AWSIZE = 256 bits |
| cov_snps_axi5_awsize_512_bits | Observed write burst with AWSIZE = 512 bits |
| cov_snps_axi5_awsize_1024_bits | Observed write burst with AWSIZE = 1024 bits |
| cov_snps_axi5_awlock_normal | Observed write burst with AWLOCK = Normal access |
| cov_snps_axi5_awlock_excl | Observed write burst with AWLOCK = Exclusive access |
| cov_snps_axi5_awcache_noncacheable_nonbufferable | Observed write burst with AWCACHE = Noncacheable and nonbufferable |
| cov_snps_axi5_awcache_bufferable_only | Observed write burst with AWCACHE = Bufferable only |
| cov_snps_axi5_awcache_cacheable_not_allocate | Observed write burst with AWCACHE = Cacheable, but do not allocate" |
| cov_snps_axi5_awcache_cacheable_bufferable_not_allocate | Observed write burst with AWCACHE = Cacheable and bufferable, but do not allocate" |
| cov_snps_axi5_awcache_cacheable_writethrough_allocate_readonly | Observed write burst with AWCACHE = Cacheable write-through, allocate on reads only" |
| cov_snps_axi5_awcache_cacheable_writeback_allocate_readonly | Observed write burst with AWCACHE = Cacheable write-back, allocate on reads only" |
| cov_snps_axi5_awcache_cacheable_writethrough_allocate_writeonly | Observed write burst with AWCACHE = Cacheable write-through, allocate on writes only" |
| cov_snps_axi5_awcache_cacheable_writeback_allocate_writeonly | Observed write burst with AWCACHE = Cacheable write-back, allocate on writes only" |

**Table 4-4 AXI5 Cover Properties**

| Property Name | Property Description |
|---|---|
| cov_snps_axi5_awcache_cacheable_writethrough_allocate_read_write | Observed write burst with AWCACHE = Cacheable write-through, allocate on both reads and writes " |
| cov_snps_axi5_awcache_cacheable_writeback_allocate_read_write | Observed write burst with AWCACHE = Cacheable write-back, allocate on both reads and writes " |
| cov_snps_axi5_awprot_normal_access | Observed write burst with AWPROT = normal access |
| cov_snps_axi5_awprot_privileged_access | Observed write burst with AWPROT = privileged access |
| cov_snps_axi5_awprot_secure_access | Observed write burst with AWPROT = secure access |
| cov_snps_axi5_awprot_nonsecure_access | Observed write burst with AWPROT = nonsecure access |
| cov_snps_axi5_awprot_data_access | Observed write burst with AWPROT = data access |
| cov_snps_axi5_awprot_instruction_access | Observed write burst with AWPROT = instruction access |
| cov_snps_axi5_awid_value | Observed write burst with AWID = N (all possible IDs) |
| cov_snps_axi5_awid_bit_l | Observed write burst with AWID[n] = 0 |
| cov_snps_axi5_awid_bit_h | Observed write burst with AWID[n] = 1 |
| cov_snps_axi5_awvalid_wait_awready | Observed AWVALID = High and AWREADY = Low |
| cov_snps_axi5_awready_wait_awvalid | Observed AWVALID = Low and AWREADY = High |
| cov_snps_axi5_awvalid_awready_both | Observed AWVALID = High and AWREADT = High |
| cov_snps_axi5_awvalid_awready_idle | Observed AWVALID = Low and AWREADT = Low |
| cov_snps_axi5_wvalid_wait_wready | Observed WVALID = High and WREADT = Low |
| cov_snps_axi5_wready_wait_wvalid | Observed WVALID = Low and WREADT = High |
| cov_snps_axi5_wvalid_wready_both | Observed WVALID = High and WREADT = High |
| cov_snps_axi5_wvalid_wready_idle | Observed WVALID = Low and WREADT = Low |
| cov_snps_axi5_bresokay | Observed write response with BRESP = OKAY |
| cov_snps_axi5_bresp_exokay | Observed write response with BRESP = EXOKAY |
| cov_snps_axi5_bresp_decerr | Observed write response with BRESP = SLVERR |
| cov_snps_axi5_bresp_slverr | Observed write response with BRESP = DECERR |
| cov_snps_axi5_bid_value | Observed write response with BID = N (all possible IDs) |
| cov_snps_axi5_bid_bit_l | Observed write response with BID[n] = 0 |
| cov_snps_axi5_bid_bit_h | Observed write response with BID[n] = 1 |
| cov_snps_axi5_bvalid_wait_bready | Observed BVALID = High and BREADY = Low |

**Table 4-4 AXI5 Cover Properties**

| Property Name | Property Description |
|---|---|
| cov_snps_axi5_bready_wait_bvalid | Observed BVALID = Low and BREADY = High |
| cov_snps_axi5_bvalid_bready_both | Observed BVALID = High and BREADY = High |
| cov_snps_axi5_bvalid_bready_idle | Observed BVALID = Low and BREADY = Low |
| cov_snps_axi5_write_out_of_order | Observed out of order write response |
| cov_snps_axi5_write_addr_resp_outstands | Observed the maximum number of outstanding write transactions reach to WR_MAX_BURSTS (address channel) |
| cov_snps_axi5_write_data_resp_outstands | Observed the maximum number of outstanding write transactions reach to WR_MAX_BURSTS (data channel) |
| cov_snps_axi5_arburst_fixed | Observed FIXED read burst |
| cov_snps_axi5_arburst_incr | Observed INCR read burst |
| cov_snps_axi5_arburst_wrap | Observed WRAP read burst |
| cov_snps_axi5_arlen_len | Observed read burst with ARLEN = N (N: 0 to MAXBURSTLENGTH-1) |
| cov_snps_axi5_arsize_8_bits | Observed read burst with ARSIZE = 8 bits |
| cov_snps_axi5_arsize_16_bits | Observed read burst with ARSIZE = 16 bits |
| cov_snps_axi5_arsize_32_bits | Observed read burst with ARSIZE = 32 bits |
| cov_snps_axi5_arsize_64_bits | Observed read burst with ARSIZE = 64 bits |
| cov_snps_axi5_arsize_128_bits | Observed read burst with ARSIZE = 128 bits |
| cov_snps_axi5_arsize_256_bits | Observed read burst with ARSIZE = 256 bits |
| cov_snps_axi5_arsize_512_bits | Observed read burst with ARSIZE = 512 bits |
| cov_snps_axi5_arsize_1024_bits | Observed read burst with ARSIZE = 1024 bits |
| cov_snps_axi5_arlock_normal | Observed read burst with ARLOCK = Normal access |
| cov_snps_axi5_arlock_excl | Observed read burst with ARLOCK = Exclusive access |
| cov_snps_axi5_arcache_noncacheable_nonbufferable | Observed read burst with ARCACHE = Noncacheable and nonbufferable |
| cov_snps_axi5_arcache_bufferable_only | Observed read burst with ARCACHE = Bufferable only |
| cov_snps_axi5_arcache_cacheable_not_allocate | Observed read burst with ARCACHE = Cacheable, but do not allocate" |
| cov_snps_axi5_arcache_cacheable_bufferable_not_allocate | Observed read burst with ARCACHE = Cacheable and bufferable, but do not allocate" |
| cov_snps_axi5_arcache_cacheable_writethrough_allocate_readonly | Observed read burst with ARCACHE = Cacheable write-through, allocate on reads only" |

**Table 4-4 AXI5 Cover Properties**

| Property Name | Property Description |
|---|---|
| cov_snps_axi5_arcache_cacheable_writeback_allocate_readonly | Observed read burst with ARCACHE = Cacheable write-back, allocate on reads only" |
| cov_snps_axi5_arcache_cacheable_writethrough_allocate_writeonly | Observed read burst with ARCACHE = Cacheable write-through, allocate on writes only" |
| cov_snps_axi5_arcache_cacheable_writeback_allocate_writeonly | Observed read burst with ARCACHE = Cacheable write-back, allocate on writes only" |
| cov_snps_axi5_arcache_cacheable_writethrough_allocate_read_write | Observed read burst with ARCACHE = Cacheable write-through, allocate on both reads and writes " |
| cov_snps_axi5_arcache_cacheable_writeback_allocate_read_write | Observed read burst with ARCACHE = Cacheable write-back, allocate on both reads and writes " |
| cov_snps_axi5_arprot_normal_access | Observed read burst with ARPROT = normal access |
| cov_snps_axi5_arprot_privileged_access | Observed read burst with ARPROT = privileged access |
| cov_snps_axi5_arprot_secure_access | Observed read burst with ARPROT = secure access |
| cov_snps_axi5_arprot_nonsecure_access | Observed read burst with ARPROT = nonsecure access |
| cov_snps_axi5_arprot_data_access | Observed read burst with ARPROT = data access |
| cov_snps_axi5_arprot_instruction_access | Observed read burst with ARPROT = instruction access |
| cov_snps_axi5_arid_value | Observed read burst with ARID = N (all possible IDs) |
| cov_snps_axi5_arid_bit_l | Observed read burst with ARID[n] = 0 |
| cov_snps_axi5_arid_bit_h | Observed read burst with ARID[n] = 1 |
| cov_snps_axi5_arvalid_wait_arready | Observed ARVALID = High and ARREADY = Low |
| cov_snps_axi5_arready_wait_arvalid | Observed ARVALID = Low and ARREADY = High |
| cov_snps_axi5_arvalid_arready_both | Observed ARVALID = High and ARREADT = High |
| cov_snps_axi5_arvalid_arready_idle | Observed ARVALID = Low and ARREADT = Low |
| cov_snps_axi5_rresokay | Observed read response with RRESP = OKAY |
| cov_snps_axi5_rresp_exokay | Observed read response with RRESP = EXOKAY |
| cov_snps_axi5_rresp_decerr | Observed read response with RRESP = SLVERR |
| cov_snps_axi5_rresp_slverr | Observed read response with RRESP = DECERR |
| cov_snps_axi5_rid_value | Observed read data with RID = N (all possible IDs) |
| cov_snps_axi5_rid_bit_l | Observed read data with RID[n] = 0 |
| cov_snps_axi5_rid_bit_h | Observed read data with RID[n] = 1 |
| cov_snps_axi5_rvalid_wait_rready | Observed RVALID = High and RREADY = Low |

Table 4-4 **AXI5 Cover Properties**

| Property Name | Property Description |
|---|---|
| cov_snps_axi5_rready_wait_rvalid | Observed RVALID = Low and RREADY = High |
| cov_snps_axi5_rvalid_rready_both | Observed RVALID = High and RREADY = High |
| cov_snps_axi5_rvalid_rready_idle | Observed RVALID = Low and RREADY = Low |
| cov_snps_axi5_read_interleaved | Observed interleaved read bursts |
| cov_snps_axi5_read_out_of_order | Observed out of order read response |
| cov_snps_axi5_read_addr_data_outstands | Observed the maximum number of outstanding read transactions reach to RD_MAX_BURSTS |
| cov_snps_axi5_awqos_value | Observed specific value in AWQOS. |
| cov_snps_axi5_arqos_value | Observed specific value in ARQOS. |
| cov_snps_axi5_awregion_value | Observed specific value in AWREGION. |
| cov_snps_axi5_arregion_value | Observed specific value in ARREGION. |
| cov_snps_axi5_non_atomic | Observed Non Atomic transaction. |
| cov_snps_axi5_atomic_store_be_add | Observed Atomic Store big endian ADD transaction. |
| cov_snps_axi5_atomic_store_le_add | Observed Atomic Store little endian ADD transaction. |
| cov_snps_axi5_atomic_store_be_clr | Observed Atomic Store big endian CLR transaction. |
| cov_snps_axi5_atomic_store_le_clr | Observed Atomic Store little endian CLR transaction. |
| cov_snps_axi5_atomic_store_be_eor | Observed Atomic Store big endian EOR transaction. |
| cov_snps_axi5_atomic_store_le_eor | Observed Atomic Store little endian EOR transaction. |
| cov_snps_axi5_atomic_store_be_set | Observed Atomic Store big endian SET transaction. |
| cov_snps_axi5_atomic_store_le_set | Observed Atomic Store little endian SET transaction. |
| cov_snps_axi5_atomic_store_be_smax | Observed Atomic Store big endian SMAX transaction. |
| cov_snps_axi5_atomic_store_le_smax | Observed Atomic Store little endian SMAX transaction. |
| cov_snps_axi5_atomic_store_be_smin | Observed Atomic Store big endian SMIN transaction. |
| cov_snps_axi5_atomic_store_le_smin | Observed Atomic Store little endian SMIN transaction. |
| cov_snps_axi5_atomic_store_be_umax | Observed Atomic Store big endian UMAX transaction. |
| cov_snps_axi5_atomic_store_le_umax | Observed Atomic Store little endian UMAX transaction. |
| cov_snps_axi5_atomic_store_be_umin | Observed Atomic Store big endian UMIN transaction. |
| cov_snps_axi5_atomic_store_le_umin | Observed Atomic Store little endian UMIN transaction. |
| cov_snps_axi5_atomic_load_be_add | Observed Atomic Load big endian ADD transaction. |

**Table 4-4 AXI5 Cover Properties**

| Property Name | Property Description |
|---|---|
| cov_snps_axi5_atomic_load_le_add | Observed Atomic Load little endian ADD transaction. |
| cov_snps_axi5_atomic_load_be_clr | Observed Atomic Load big endian CLR transaction. |
| cov_snps_axi5_atomic_load_le_clr | Observed Atomic Load little endian CLR transaction. |
| cov_snps_axi5_atomic_load_be_eor | Observed Atomic Load big endian EOR transaction. |
| cov_snps_axi5_atomic_load_le_eor | Observed Atomic Load little endian EOR transaction. |
| cov_snps_axi5_atomic_load_be_set | Observed Atomic Load big endian SET transaction. |
| cov_snps_axi5_atomic_load_le_set | Observed Atomic Load little endian SET transaction. |
| cov_snps_axi5_atomic_load_be_smax | Observed Atomic Load big endian SMAX transaction. |
| cov_snps_axi5_atomic_load_le_smax | Observed Atomic Load little endian SMAX transaction. |
| cov_snps_axi5_atomic_load_be_smin | Observed Atomic Load big endian SMIN transaction. |
| cov_snps_axi5_atomic_load_le_smin | Observed Atomic Load little endian SMIN transaction. |
| cov_snps_axi5_atomic_load_be_umax | Observed Atomic Load big endian UMAX transaction. |
| cov_snps_axi5_atomic_load_le_umax | Observed Atomic Load little endian UMAX transaction. |
| cov_snps_axi5_atomic_load_be_umin | Observed Atomic Load big endian UMIN transaction. |
| cov_snps_axi5_atomic_load_le_umin | Observed Atomic Load little endian UMIN transaction. |
| cov_snps_axi5_atomic_swap | Observed Atomic Swap transaction. |
| cov_snps_axi5_atomic_comp | Observed Atomic Compare transaction. |
| cov_snps_axi5_awloop_value | Observed specific value in AWLOOP. |
| cov_snps_axi5_bloop_value | Observed specific value in BLOOP. |
| cov_snps_axi5_arloop_value | Observed specific value in ARLOOP. |
| cov_snps_axi5_rloop_value | Observed specific value in RLOOP. |
| cov_snps_axi5_wdatachk_high | Observed High in each bit of WDATACHK. |
| cov_snps_axi5_wdatachk_low | Observed Low in each bit of WDATACHK. |
| cov_snps_axi5_rdatachk_high | Observed High in each bit of RDATACHK. |
| cov_snps_axi5_rdatachk_low | Observed Low in each bit of RDATACHK. |
| cov_snps_axi5_wpoison_high | Observed High in each bit of WPOISON. |
| cov_snps_axi5_wpoison_low | Observed Low in each bit of WPOISON. |
| cov_snps_axi5_rpoison_high | Observed High in each bit of RPOISON. |
| cov_snps_axi5_rpoison_low | Observed Low in each bit of RPOISON. |

**Table 4-4 AXI5 Cover Properties**

| Property Name | Property Description |
|---|---|
| cov_snps_axi5_vawqosaccept_value | Observed specific value in VAWQOSACCEPT. |
| cov_snps_axi5_varqosaccept_value | Observed specific value in VARQOSACCEPT. |
| cov_snps_axi5_awtrace_high | Observed High in AWTRACE. |
| cov_snps_axi5_awtrace_low | Observed Low in AWTRACE. |
| cov_snps_axi5_btrace_high | Observed High in BTRACE. |
| cov_snps_axi5_btrace_low | Observed Low in BTRACE. |
| cov_snps_axi5_artrace_high | Observed High in ARTRACE. |
| cov_snps_axi5_artrace_low | Observed Low in ARTRACE. |
| cov_snps_axi5_rtrace_high | Observed High in RTRACE. |
| cov_snps_axi5_rtrace_low | Observed Low in RTRACE. |
| cov_snps_axi5_awmmusecsid_high | Observed High in AWMMUSECSID. |
| cov_snps_axi5_awmmusecsid_low | Observed Low in AWMMUSECSID. |
| cov_snps_axi5_awmmussidv_high | Observed High in AWMMUSSIDV. |
| cov_snps_axi5_awmmussidv_low | Observed Low in AWMMUSSIDV. |
| cov_snps_axi5_awmmuatst_high | Observed High in AWMMUATST. |
| cov_snps_axi5_awmmuatst_low | Observed Low in AWMMUATST. |
| cov_snps_axi5_armmusecsid_high | Observed High in ARMMUSECSID. |
| cov_snps_axi5_armmusecsid_low | Observed Low in ARMMUSECSID. |
| cov_snps_axi5_armmussidv_high | Observed High in ARMMUSSIDV. |
| cov_snps_axi5_armmussidv_low | Observed Low in ARMMUSSIDV. |
| cov_snps_axi5_armmuatst_high | Observed High in ARMMUATST. |
| cov_snps_axi5_armmuatst_low | Observed Low in ARMMUATST. |
| cov_snps_axi5_awnsaid_value | Observed specific value in AWNSAID. |
| cov_snps_axi5_arnsaid_value | Observed specific value in ARNSAID. |
| cov_snps_axi5_awakeup_rose | Observed AWAKEUP transits from Low to High. |
| cov_snps_axi5_awakeup_fell | Observed AWAKEUP transits from High to Low. |

## 4.5 Behavior of Properties

Properties are grouped on the basis of categories, which depends on the configuration parameter values. Categories can be like x_check properties, configure command properties and max waits properties. Parameters have some default value, refer to Table 4-1.

- To instantiate x_check properties:

  Set `CONFIG_X_CHECK=1`
- To instantiate max wait check properties:

  Set `CONFIG_WAITS=1`

Similarly, to enable all assert or assume properties, the `ENABLE_ASSERT` and `ENABLE_ASSUME` parameters must be set to 1. See Table 4-1 for information on “Required parameters” for each property.

### 4.5.1 Properties as Assert Directives

Assert properties have the following features:

- Assert properties check the functionality of a protocol by monitoring its output as per its input, and issue an error message when the protocol is violated.
- When AGENT_TYPE is MASTER, checkers mentioned as `Master’ in the Master/Slave checkers column of Table 4-1 are declared as assume, and checkers mentioned as `Slave’ in Master/Slave column are declared as assert.
- When AGENT_TYPE is SLAVE, checkers mentioned as ‘Slave’ in Master/Slave column of Table 4-1 are declared as assume, and checkers mentioned as ‘Master’ in Master/Slave column are declared as assert.

### 4.5.2 Properties as Assume Directives

Assume properties have the following features:

- Assume properties act as a constraint for generating controlled stimulus as per a protocol because the VC Formal tool treats the inputs as free variables.
- When AGENT_TYPE is MASTER, checkers mentioned as `Master’ in Master/Slave checkers column are declared as assume, and checkers mentioned as `Slave’ in Master/Slave column are declared as assert.
- When AGENT_TYPE is SLAVE, checkers mentioned as ‘Slave’ in Master/Slave column are declared as assume, and checkers mentioned as ‘Master’ in Master/Slave column are declared as assert.

### 4.5.3 Properties In Cover Directives

Cover properties have the following features:

- Cover properties tells the number of assertions executed or covered when stimulus is generated. This number reflects the AIP coverage. Also, the cover properties Indicate the type of transactions or scenarios exercised during proof. It can be helpful in checking number of unexecuted properties.
- Cover properties are useful in checking if there are no over-constraints in environment, and if the design issues all possible transactions.
- When ENABLE_COVER = 1, cover properties are generated. Note that the cover properties are independent from the properties used as assert/assume.

## 4.6 The CONFIG_MAXOUTS Parameter Setting Change

Starting with the 2020.03-SP2-1 patch release, the CONFIG_MAXOUTS parameter setting is changed in AXI5 AIP. This enhancement allows you to adjust constraints to avoid missing bugs in design implementation.

Table 4-5 describes difference in behavior of the CONFIG_MAXOUTS parameter.

**Table 4-5 CONFIG_MAXOUTS Parameter Setting Behavior Change**

| Release | Behavior |
|---|---|
| 2020.03-SP2 and below versions | CONFIG_MAXOUTS can be either 0 or 1<br>Default value is 1 |
| 2020.03-SP2-1 and above versions | CONFIG_MAXOUTS==0: same with previous version<br>CONFIG_MAXOUTS==1: new behavior<br>CONFIG_MAXOUTS==2: same with CONFIG_MAXOUTS==1 in previous version<br>Default value is 1 |

Table 4-6 describes behavior per value:

**Table 4-6 Behavior Per Value**

| | |
|---|---|
| CONFIG_MAXOUTS==0 | No change with previous version |
| | Master AIP does not constrain AWVALID, WVALID nor ARVALID. These signals may be asserted even the number of outstanding transactions exceed WR_MAX_BURSTS or RD_MAX_BURSTS. |
| | Master AIP does not check AWREADY, WREADY nor ARREADY if they are de-asserted even when the number of outstanding transactions reaches to WR_MAX_BURSTS or RD_MAX_BURSTS. |
| | Slave AIP does not check AWVALID, WVALID nor ARVALID if they are de-asserted even when the number of outstanding transactions reaches to WR_MAX_BURSTS or RD_MAX_BURSTS. |
| | Slave AIP does not constrain AWREADY, WREADY nor ARREADY. These signals may be asserted even the number of outstanding transactions exceed WR_MAX_BURSTS or RD_MAX_BURSTS. |
| CONFIG_MAXOUTS==1 | New behavior |
| | Master AIP constrains AWVALID, WVALID and ARVALID with Low when the number of outstanding transactions reaches (WR_MAX_BURSTS+1) or (RD_MAX_BURSTS+1).<br>Use asm_snps_axi_awvalid_maxexcd, asm_snps_axi_wvalid_maxexcd and asm_snps_axi_arvalid_maxexcd. |
| | Master AIP checks AWREADY, WREADY and ARREADY if they are de-asserted when the number of outstanding transactions reaches to WR_MAX__BURSTS or RD_MAX_BURSTS.<br>Use asm_snps_axi_awvalid_maxouts, asm_snps_axi_wvalid_maxouts and asm_snps_axi_arvalid_maxouts. |

**Table 4-6 Behavior Per Value**

| | |
|---|---|
| | Slave AIP checks AWVALID, WVALID and ARVALID if they are de-asserted when the number of outstanding transactions reaches to WR_MAX__BURSTS or RD_MAX_BURSTS.<br>Use asm_snps_axi_awvalid_maxouts, asm_snps_axi_wvalid_maxouts and asm_snps_axi_arvalid_maxouts. |
| | Slave AIP constrains AWREADY, WREADY and ARREADY with Low when the number of outstanding transactions reaches (WR_MAX_BURSTS+1) or (RD_MAX_BURSTS+1).<br>Use asm_snps_axi_awvalid_maxexcd, asm_snps_axi_wvalid_maxexcd and asm_snps_axi_arvalid_maxexcd. |
| CONFIG_MAXOUTS==2 | Same with previous behavior with CONFIG_MAXOUTS==1 |
| | Master AIP constrains AWVALID, WVALID and ARVALID with Low when the number of outstanding transactions reaches WR_MAX_BURSTS or RD_MAX_BURSTS.<br>Use asm_snps_axi_awvalid_maxouts, asm_snps_axi_wvalid_maxouts and asm_snps_axi_arvalid_maxouts. |
| | Master AIP checks AWREADY, WREADY and ARREADY if they are de-asserted when the number of outstanding transactions reaches to WR_MAX__BURSTS or RD_MAX_BURSTS.<br>Use asm_snps_axi_awvalid_maxouts, asm_snps_axi_wvalid_maxouts and asm_snps_axi_arvalid_maxouts. |
| | Slave AIP checks AWVALID, WVALID and ARVALID if they are de-asserted when the number of outstanding transactions reaches to WR_MAX__BURSTS or RD_MAX_BURSTS.<br>Use asm_snps_axi_awvalid_maxouts, asm_snps_axi_wvalid_maxouts and asm_snps_axi_arvalid_maxouts. |
| | Slave AIP constrains AWREADY, WREADY and ARREADY with Low when the number of outstanding transactions reaches WR_MAX_BURSTS or RD_MAX_BURSTS.<br>Use asm_snps_axi_awvalid_maxouts, asm_snps_axi_wvalid_maxouts and asm_snps_axi_arvalid_maxouts. |

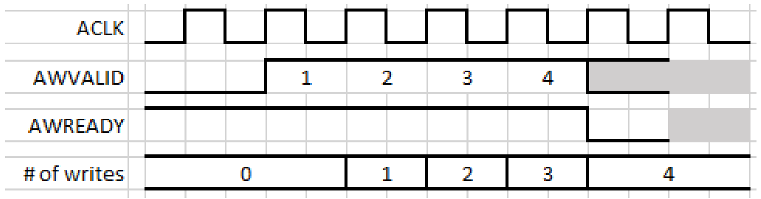
This change is applicable only for assume properties. This is not applicable for AMBA specification and depends on design implementation. Hence, assertions ast_snps_axi5_awvalid_maxouts, ast_snps_axi5_wvalid_maxouts, and ast_snps_axi5_arvalid_maxouts do not check protocol violation and are categorized as WARNING.

Setting CONFIG_MAXOUTS to 0 is not recommended because it most likely causes overflow in AIP internal data structure.

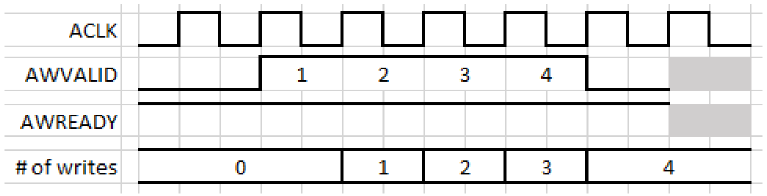
### 4.6.1 Background of this Change (when a potential bug can be missed in a slave DUT):

In previous releases, if you set WR_MAX_BURSTS with 4, MASTER AIP issues up to 4 outstanding write requests, and won’t issue 5th write request until (bvalid & bready) is received.

Let us assume slave DUT has 4 depth FIFO and slave DUT stops driving AWREADY when FIFO is full. The expected slave DUT behavior is as follows:

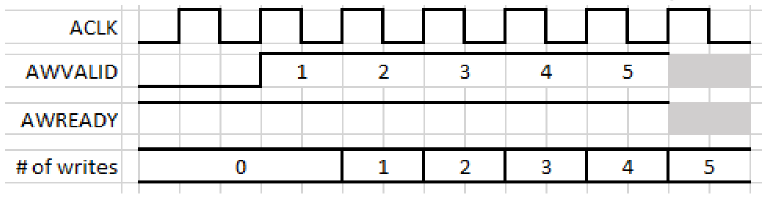


However, if MASTER AIP does not issue 5th write request, then it is not possible to verify if slave DUT stops AWREADY or not for the 5th request. As shown below, the number of outstanding transactions is 4 and slave DUT returns 5th AWEREADY.

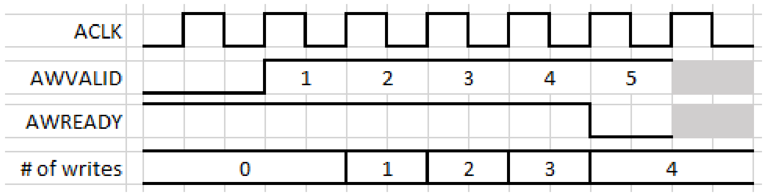


To verify if slave DUT stops to return AWREADY, you must set WR_MAX_BURSTS with 5 or bigger. However, user most likely sets WR_MAX_BURSTS with 4 in such cases. Therefore, it would be desirable to change AIP behavior that MASTER AIP issues write requests up to (WR_MAX_BURSTS+1). This is same with read requests.

As shown below, if slave DUT returns 5th AWREADY, the number of outstanding transactions reaches 5 and exceeds WR_MAX_BURTS, and AIP can detect slave DUT bug.



If slave DUT stops to return 5th AWREADY, the number of outstanding transactions reaches 4 and MASTER AIP drives 5th AWVALID as follows:.



Default behavior is changed to issue (WR_MAX_BURST+1) outstanding write requests and (RD_MAX_BURSTS+1) read requests.

This behavior change is applied only to assume properties. There is no change for assert properties.

### 4.6.2 Backwards Compatibility

In some cases, it is required to keep previous behavior. For example, AIP back-to-back environment. Another example is if DUT is AXI bridge and upstream transactions are passed through to downstream, DUT does not have any limitation regarding number of outstanding transactions. In such cases, CONFIG_MAXOUTS should be set with 2.

## 4.7 Supported Version

AXI5 AIP supports IHI 0022H.c starting from 2023.03-1 and 2022.06-SP2-3. This includes the following features:

- Atomic Transactions
- Memory Partitioning and Monitoring (MPAM)
- Unique ID indicator
- Read data chunking
- Memory Tagging Enhancement (MTE)
- Regular Transactions Only
- Maximum transaction size and boundary
- Consistent DECERR response

New signals are added in AIP ports to support these new features.

If verification environment is created based on older version of AXI5 AIP, set the AIP ports in the bind statement as follows if these signals does not exist in the design:

```
.awmmuflow  ('b0),
.awmpam     (),
.awidunq    (),
.awtagop    ('b0),
.wtag       ('b0),
.wtagupdate ('b0),
.bidunq     (),
.btagmatch  ('b0),
.bcomp      (),
.armmuflow  ('b0),
.armpam     (),
.aridunq    (),
.archunken  (),
.artagop    (),
.ridunq     (),
.rchunkv    (),
.rchunknum  (),
.rchunkstrb (),
.rtag       (),
```

# 5
# The AXI5 AIP Use Cases

This chapter discusses about the AXI5 AIP in different environments used for validation.

## 5.1 The AXI5 AIP Examples

This section describes the setup of the AXI5 AIP where it is connected with RTL through a bind file. The bind file example is shown in Figure 5-2, Figure 5-4, Figure 5-6, and Figure 5-8, where both master and slave DUT signals are connected to AIP master and slave signals.

**Note**
If a signal corresponding to the AXI5 AIP port does not exist in the DUT, set inactive value or the value expected by the DUT for the port. For example, if the DUT does not have the `awakeup` signal, set with `1’b0`.

### 5.1.1 The AXI5-Lite Master AIP With Slave DUT

The following steps describe the setup of the AXI5 AIP with a slave DUT:

1. In this setup, the AXI5-Lite Master AIP is connected with the AXI5-Lite Real Slave DUT.
2. For connecting the ports of the AXI5-Lite DUT with the AXI5-Lite AIP, create a bind file to include the instance of the AXI5-Lite AIP into the top level module of the DUT.
3. Instantiate the master AXI5-Lite AIP (snps_axi5_lite_aip) and connect with the slave DUT signal.
4. Figure 5-1 shows a slave DUT connected with the master AXI5-Lite AIP.

**Note**
Use the `snps_axi5_lite_aip.sv` file for AXI5-Lite AIP and configure AGENT_TYPE parameter with MASTER.

**Figure 5-1 Slave DUT with AXI5-Lite Master AIP**

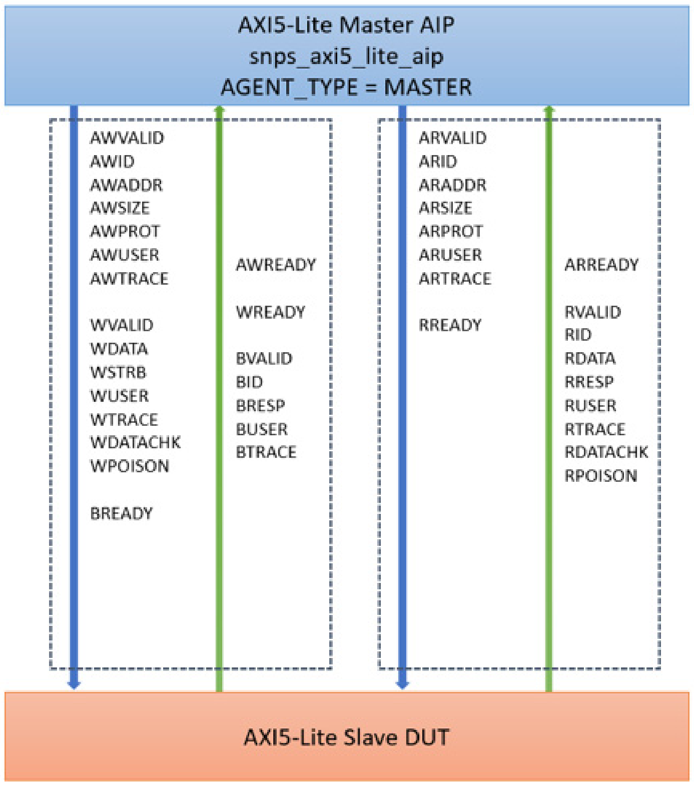
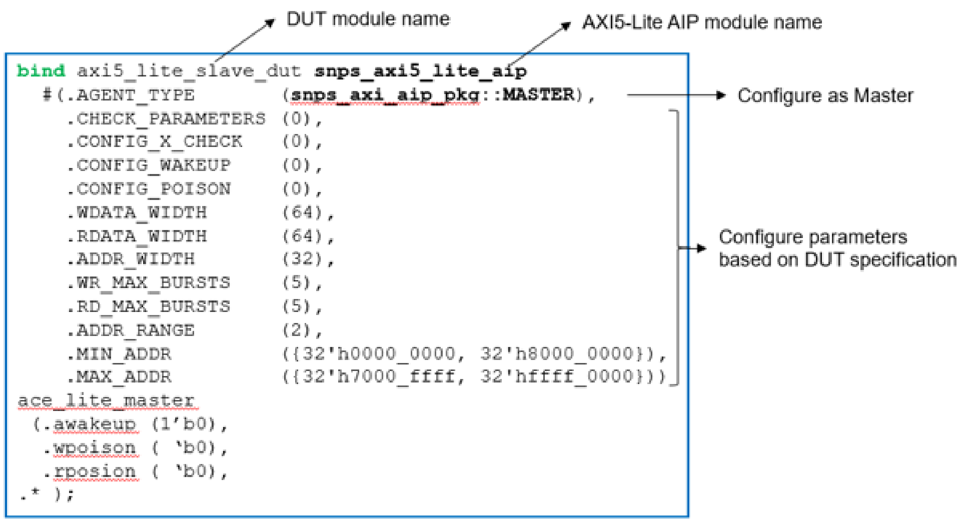


**Figure 5-2 AXI5-Lite Master AIP – Slave DUT Bind Example**



```
bind axi5_lite_slave_dut snps_axi5_lite_aip
  #(.AGENT_TYPE         (snps_axi_aip_pkg::MASTER),
     .CHECK_PARAMETERS  (0),
     .CONFIG_X_CHECK    (0),
     .CONFIG_WAKEUP     (0),
     .CONFIG_POISON     (0),
     .WDATA_WIDTH       (64),
     .RDATA_WIDTH       (64),
     .ADDR_WIDTH        (32),
     .WR_MAX_BURSTS     (5),
     .RD_MAX_BURSTS     (5),
     .ADDR_RANGE        (2),
     .MIN_ADDR          ({32'h0000_0000, 32'h8000_0000}),
     .MAX_ADDR          ({32'h7000_ffff, 32'hffff_0000}))
ace_lite_master
 (.awakeup (1'b0),
  .wpoison ( 'b0),
  .rposion ( 'b0),
.* );
```

### 5.1.2 The AXI5-Lite Slave AIP With a Master DUT

The following steps describe the setup of the AXI5-Lite Slave AIP with a master DUT:

1. In this setup, the AXI5-Lite Slave AIP is connected with the AXI5-Lite Real Master DUT.
2. For connecting the ports of the AXI5-Lite DUT with the AXI5-Lite AIP, create a bind file to include the instance of the AXI5-Lite AIP into the top level module of the DUT.
3. Instantiate the AXI5-Lite Slave AIP (snps_axi5_lite_aip) and connect with the master DUT signal.
4. Figure 5-3 shows a Master DUT connected with the AXI5-Lite Slave AIP.

**Note**
Use the `snps_axi5_lite_aip.sv` file for AXI5-Lite AIP and configure AGENT_TYPE parameter with SLAVE.

**Figure 5-3 Master DUT with AXI5-Lite Slave AIP**

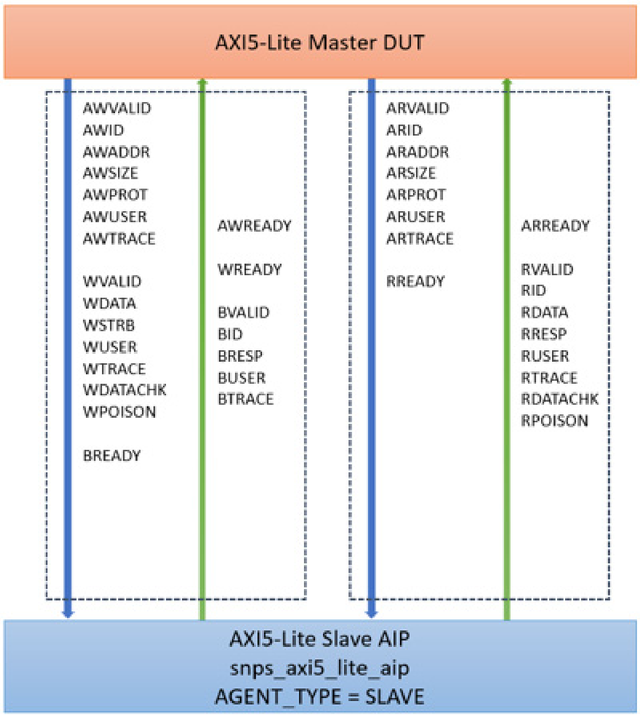
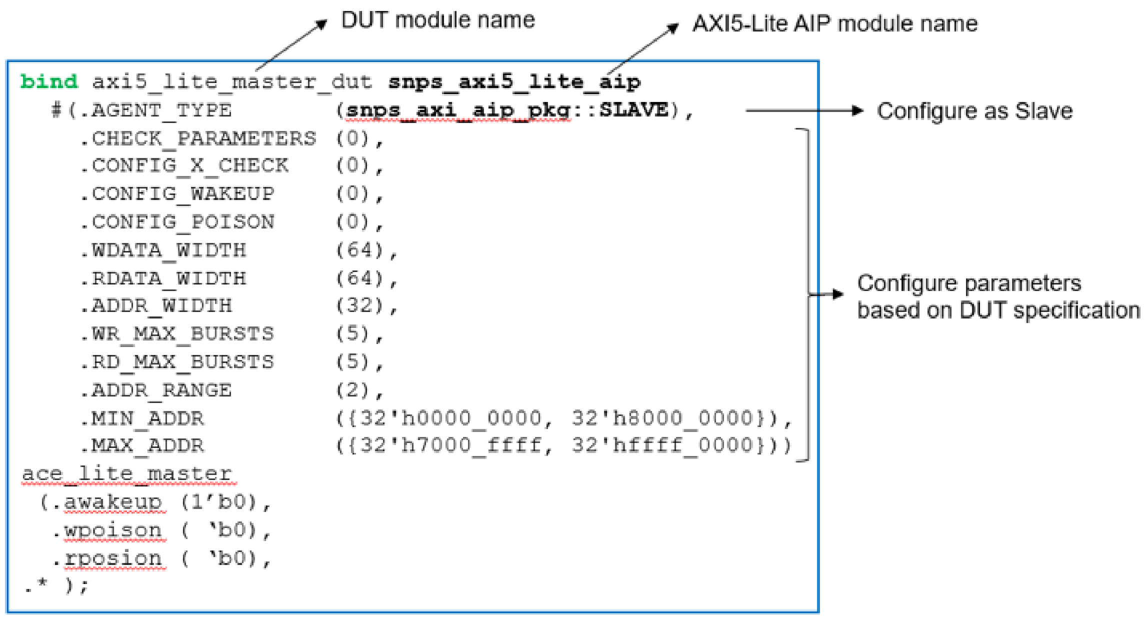


**Figure 5-4 AXI5-Lite Salve AIP – Master DUT Bind Example**


DUT module name
AXI5-Lite AIP module name
Configure as Slave
Configure parameters based on DUT specification

```
bind axi5_lite_master_dut snps_axi5_lite_aip
  #(.AGENT_TYPE         (snps_axi_aip_pkg::SLAVE),
    .CHECK_PARAMETERS (0),
    .CONFIG_X_CHECK    (0),
    .CONFIG_WAKEUP     (0),
    .CONFIG_POISON     (0),
    .WDATA_WIDTH       (64),
    .RDATA_WIDTH       (64),
    .ADDR_WIDTH        (32),
    .WR_MAX_BURSTS     (5),
    .RD_MAX_BURSTS     (5),
    .ADDR_RANGE        (2),
    .MIN_ADDR          ({32'h0000_0000, 32'h8000_0000}),
    .MAX_ADDR          ({32'h7000_ffff, 32'hffff_0000}))
ace_lite_master
 (.awakeup (1'b0),
  .wpoison ( 'b0),
  .rposion ( 'b0),
.* );
```

### 5.1.3 The AXI5 Master AIP With Slave DUT

The following steps describe the setup of the AXI5-Lite Slave AIP with a master DUT:

1. In this setup, the AXI5 Master AIP is connected with the AXI5 Real Slave DUT.
2. For connecting the ports of the AXI5 DUT with the AXI5 AIP, create a bind file to include the instance of the AXI5 AIP into the top level module of the DUT.
3. Instantiate the master AXI5 AIP (snps_axi5_aip) and connect with the slave DUT signal.
4. Figure 5-5 shows a slave DUT connected with the master AXI5 AIP.

**Note**
Use the `snps_axi5_aip.sv` file for AXI5 AIP and configure AGENT_TYPE parameter with MASTER.

**Figure 5-5 Slave DUT with AXI5 Master AIP**

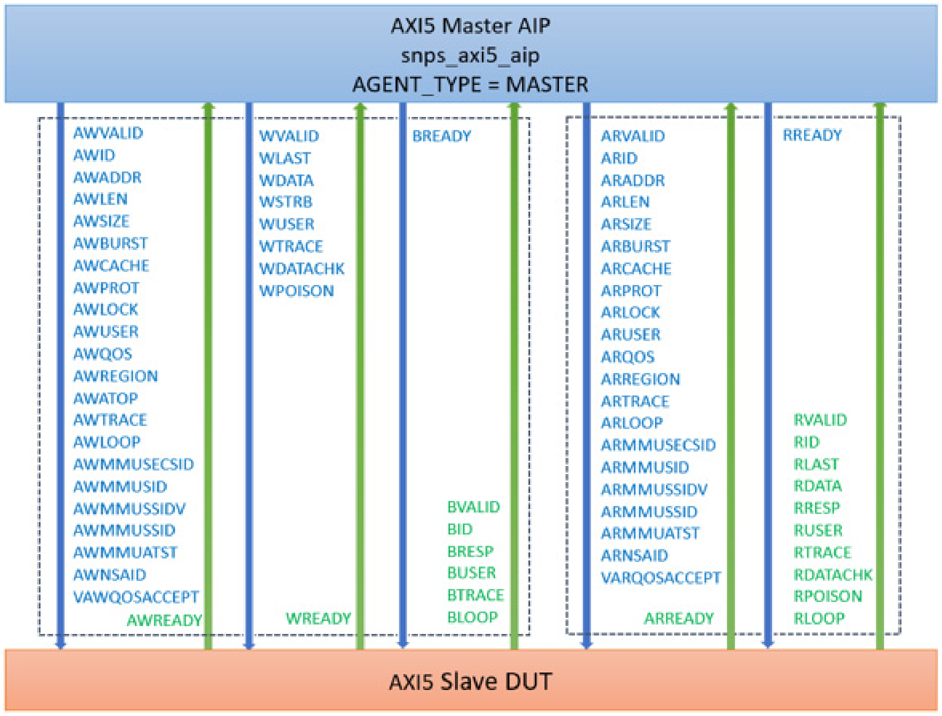
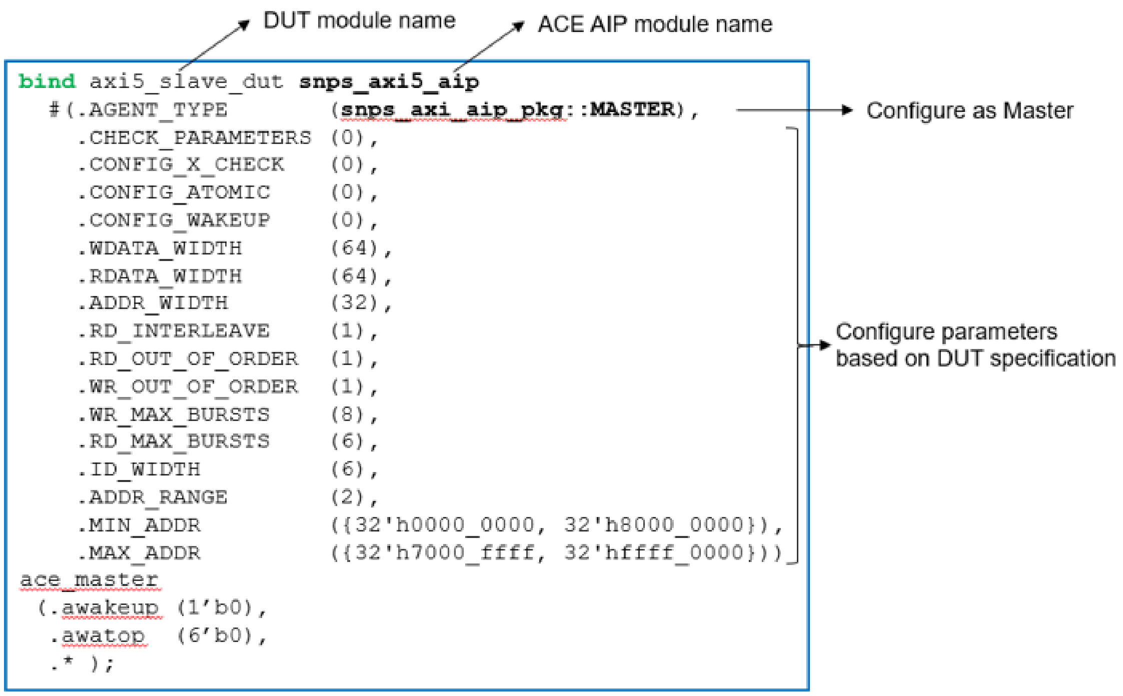


**Figure 5-6 AXI5 Master AIP – Slave DUT Bind Example**


DUT module name
ACE AIP module name
Configure as Master
Configure parameters based on DUT specification

```
bind axi5_slave_dut snps_axi5_aip
  #(.AGENT_TYPE        (snps_axi_aip_pkg::MASTER),
    .CHECK_PARAMETERS  (0),
    .CONFIG_X_CHECK    (0),
    .CONFIG_ATOMIC     (0),
    .CONFIG_WAKEUP     (0),
    .WDATA_WIDTH       (64),
    .RDATA_WIDTH       (64),
    .ADDR_WIDTH        (32),
    .RD_INTERLEAVE     (1),
    .RD_OUT_OF_ORDER   (1),
    .WR_OUT_OF_ORDER   (1),
    .WR_MAX_BURSTS     (8),
    .RD_MAX_BURSTS     (6),
    .ID_WIDTH          (6),
    .ADDR_RANGE        (2),
    .MIN_ADDR          ({32'h0000_0000, 32'h8000_0000}),
    .MAX_ADDR          ({32'h7000_ffff, 32'hffff_0000}))
ace_master
 (.awakeup  (1'b0),
  .awatop   (6'b0),
  .* );
```

### 5.1.4 The AXI5 Slave AIP With a Master DUT

The following steps describe the setup of the AXI5 Slave AIP with a master DUT:

1. In this setup, the AXI5 Slave AIP is connected with the AXI5 Real Master DUT.
2. For connecting the ports of the AXI5 DUT with the AXI5 AIP, create a bind file to include the instance of the AXI5 AIP into the top level module of the DUT.
3. Instantiate the AXI5 Slave AIP (snps_axi5_aip) and connect with the master DUT signal.
4. Figure 5-7 shows a Master DUT connected with the AXI5 Slave AIP.

**Note**
Use the `snps_axi5_aip.sv` file for AXI5 AIP and configure AGENT_TYPE parameter with SLAVE.

**Figure 5-7 Master DUT with AXI5 Slave AIP**

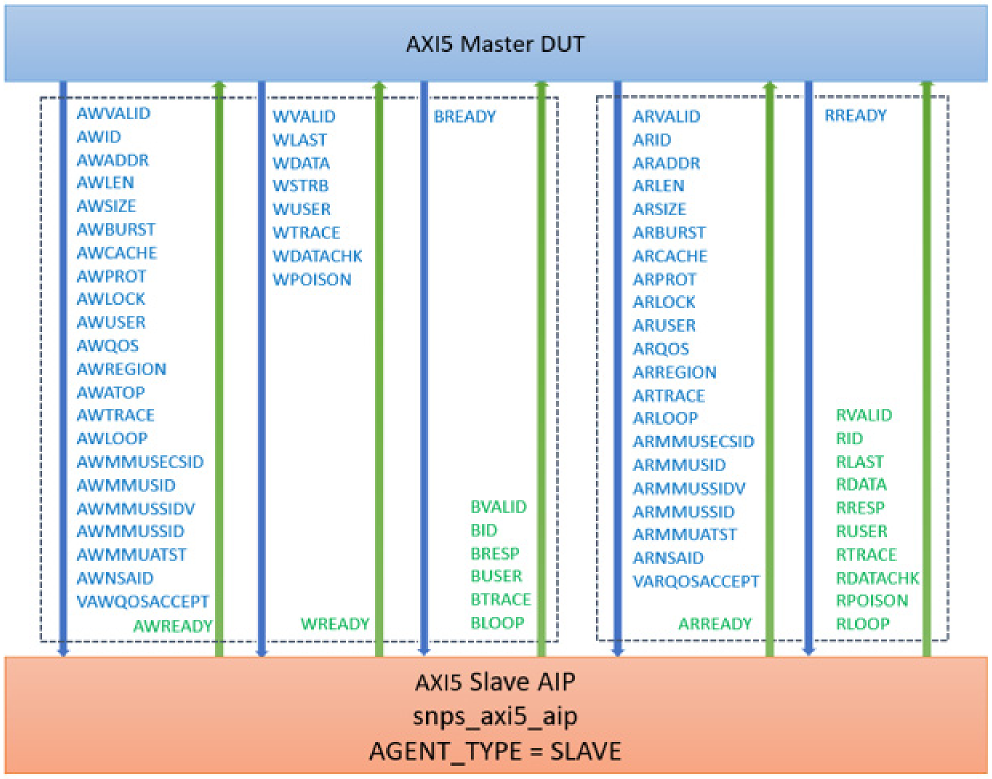
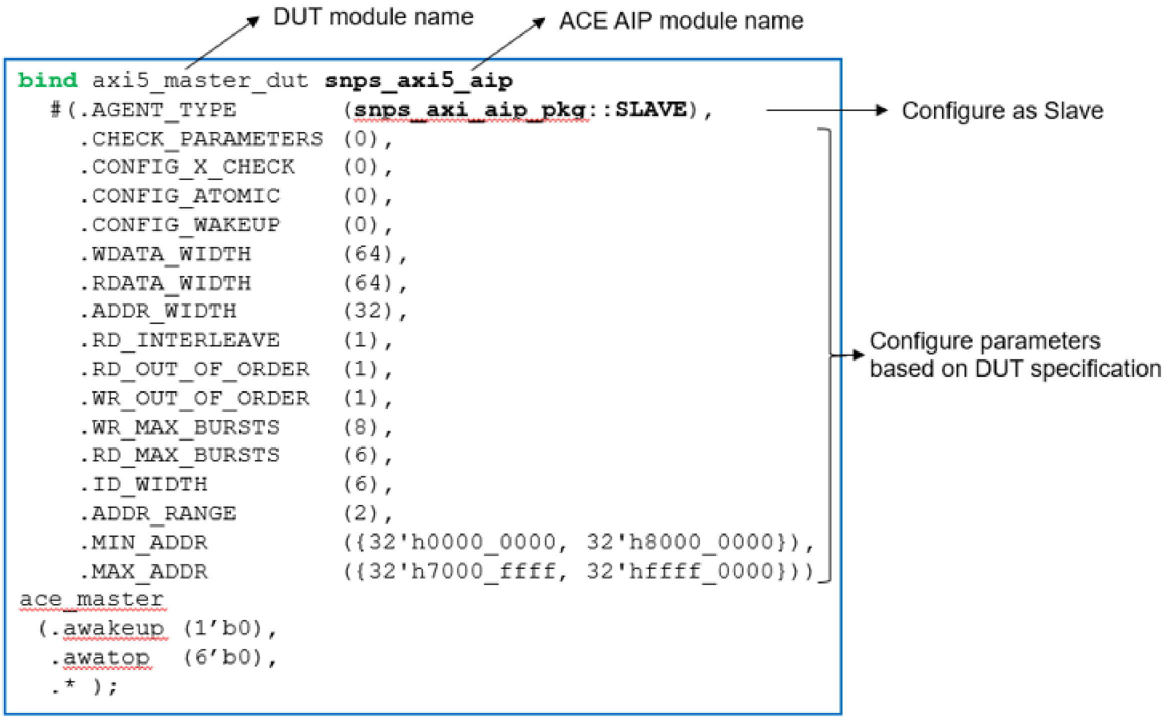


**Figure 5-8 AXI5 Slave AIP – Master DUT Bind Example**



```
bind axi5_master_dut snps_axi5_aip
  #(.AGENT_TYPE         (snps_axi_aip_pkg::SLAVE),
    .CHECK_PARAMETERS (0),
    .CONFIG_X_CHECK   (0),
    .CONFIG_ATOMIC    (0),
    .CONFIG_WAKEUP    (0),
    .WDATA_WIDTH      (64),
    .RDATA_WIDTH      (64),
    .ADDR_WIDTH       (32),
    .RD_INTERLEAVE    (1),
    .RD_OUT_OF_ORDER  (1),
    .WR_OUT_OF_ORDER  (1),
    .WR_MAX_BURSTS    (8),
    .RD_MAX_BURSTS    (6),
    .ID_WIDTH         (6),
    .ADDR_RANGE       (2),
    .MIN_ADDR         ({32'h0000_0000, 32'h8000_0000}),
    .MAX_ADDR         ({32'h7000_ffff, 32'hffff_0000}))
ace_master
 (.awakeup (1'b0),
  .awatop  (6'b0),
  .* );
```

## 5.2 Deadlock Properties

This section describes the details for deadlock properties.

### 5.2.1 Deadlock Configurations and Properties

AXI5 AIP has the following configuration parameters for deadlock related checks:

**Table 5-1 Configuration Parameters for Deadlock Related Checks**

| Parameter Name | Default | Description |
|---|---|---|
| CONFIG_WAITS | 1 | 1: Enable VALID-READY checks<br>0: Disable VALID-READY checks |
| AW_MAX_WAITS | 16 | The maximum wait cycles from AWVALID to AWREADY |
| W_MAX_WAITS | 16 | The maximum wait cycles from AWVALID to AWREADY |
| B_MAX_WAITS | 16 | The maximum wait cycles from AWVALID to AWREADY |
| AR_MAX_WAITS | 16 | The maximum wait cycles from AWVALID to AWREADY |
| R_MAX_WAITS | 16 | The maximum wait cycles from RVALID to RREADY |
| INTERCHANNEL_LATENCY | 0 | 1: Enable inter-channel latency checks<br>0: Disable inter-channel latency checks |
| AW_W_MAX_LATENCY | 16 | The maximum latency from AWVALID to WVALID (when AW is issued early) |
| W_AW_MAX_LATENCY | 16 | The maximum latency from WVALID to AWVALID (when WDATA is advanced) |
| AWW_B_MAX_LATENCY | 16 | The maximum latency from the completion of address and data channels to response channel |
| WLAST_MAX_LATENCY | 16 | The maximum latency from the first WVALID to WLAST |
| AR_R_MAX_LATENCY | 16 | The maximum latency from ARVALID to RVALID |
| RLAST_MAX_LATENCY | 16 | The maximum latency from the first RVALID to RLAST |

Liveness properties are generated when the *_MAX_WAITS or *_MAX_LATENCY parameters are set with 0. When the *_MAX_WAITS or *_MAX_LATENCY parameters are set with positive integer, safety properties are generated.

In general, safety property is better than liveness property in convergence, however, there are some cases where safety property is not applicable. The latency setting in safety property reduces design state space and it does not verify exhaustively. For example, FIFO full condition may not occur depending on the relation between latency setting and design implementation. Also, the latency setting in safety property may cause false failures depending on the relation between latency setting and design internal latency. Also, it may be quite difficult to find the setting which avoids false failures.

axi5 AIP has the following properties for deadlock related checks:

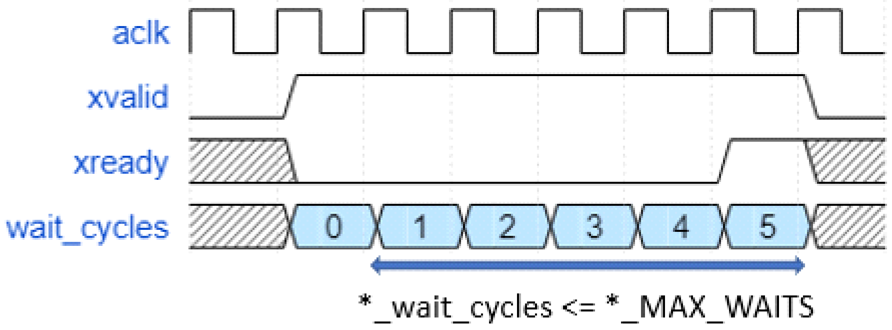
**Table 5-2 Properties for Deadlock Related Checks**

| Check Description | Property Name | Enable Condition | Parameter to Indicate Cycles |
|---|---|---|---|
| AWREADY should be asserted within AW_MAX_WAITS cycles after AWVALID is asserted. | ast_snps_axi5_awready_max_waits | CONFIG_WAITS==1 | AW_MAX_WAITS |
| WREADY should be asserted within W_MAX_WAITS cycles after WVALID is asserted. | ast_snps_axi5_wready_max_waits | CONFIG_WAITS==1 | W_MAX_WAITS |
| BREADY should be asserted within B_MAX_WAITS cycles after BVALID is asserted. | ast_snps_axi5_bready_max_waits | CONFIG_WAITS==1 | B_MAX_WAITS |
| ARREADY should be asserted within AR_MAX_WAITS cycles after ARVALID is asserted. | ast_snps_axi5_arready_max_waits | CONFIG_WAITS==1 | AR_MAX_WAITS |
| RREADY should be asserted within R_MAX_WAITS cycles after RVALID is asserted. | ast_snps_axi5_rready_max_waits | CONFIG_WAITS==1 | R_MAX_WAITS |
| If AWVALID is asserted, corresponding WVALID should be asserted within AW_W_MAX_LATENCY cycles. | ast_snps_axi5_aw_w_max_latency | INTERCHANNEL_LATENCY==1 | AW_W_MAX_LATENCY |
| If WVALID is asserted, corresponding AWVALID should be asserted within W_AW_MAX_LATENCY cycles. | ast_snps_axi5_w_aw_max_latency | INTERCHANNEL_LATENCY==1 | W_AW_MAX_LATENCY |
| If AW/W pair is done, corresponding BVALID should be asserted within AWW_B_MAX_LATENCY cycles. | ast_snps_axi5_aww_b_max_latency | INTERCHANNEL_LATENCY==1 | AWW_B_MAX_LATENCY |
| If WVALID is asserted without WLAST, corresponding WLAST should be asserted within WLAST_MAX_LATENCY cycles. | ast_snps_axi5_wlast_max_latency | INTERCHANNEL_LATENCY==1 | WLAST_MAX_LATENCY |
| If ARVALID is asserted, corresponding RVALID should be asserted within AR_R_MAX_LATENCY cycles. | ast_snps_axi5_ar_r_max_latency | INTERCHANNEL_LATENCY==1 | AR_R_MAX_LATENCY |
| If RVALID is asserted, corresponding RLAST should be asserted within RLAST_MAX_LATENCY cycles. | ast_snps_axi5_rlast_max_latency | INTERCHANNEL_LATENCY==1 | RLAST_MAX_LATENCY |

### 5.2.2 Deadlock Properties Timing Chart

1. The following diagram shows an example for the relation between xvalid and xready. Where, 'x' indicates 'aw', 'w', 'b', 'ar' or 'r'.

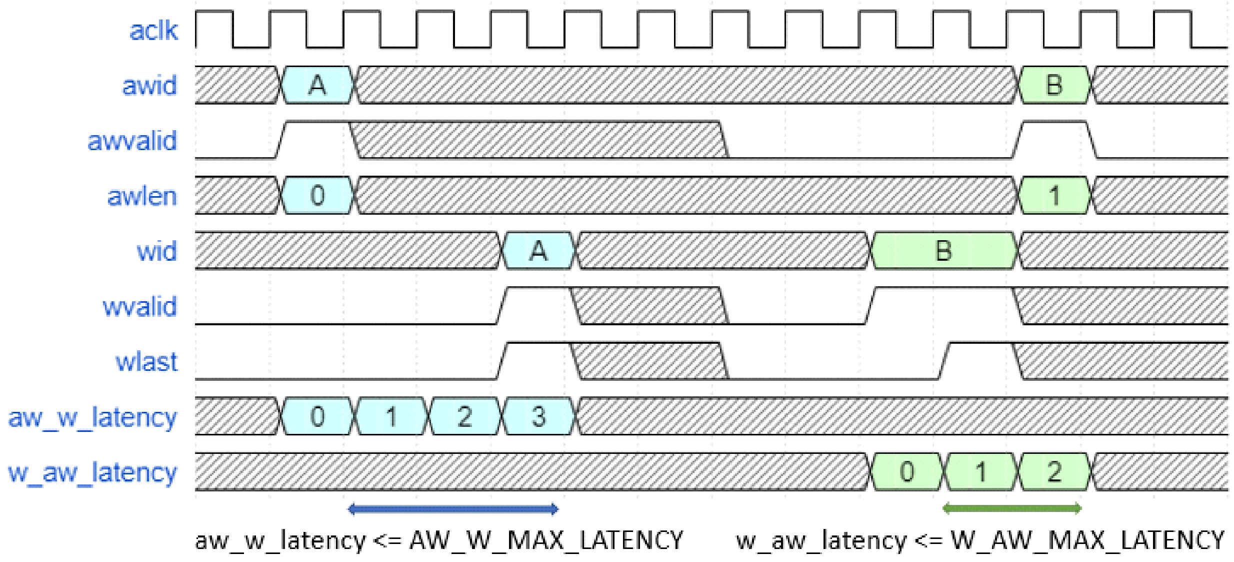
**Figure 5-9 Example for the Relation Between xvalid and xready**



The assertion ast_snps_axi5_awready_max_waits checks if 'awready' should be returned within AW_MAX_WAITS cycles once 'awvalid' is asserted. Similarly, ast_snps_axi5_wready_max_waits checks maximum wait cycles of 'wready', ast_snps_axi5_bready_max_waits checks maximum wait cycles of 'bready', ast_snps_axi5_arready_max_waits checks maximum wait cycles of 'arready' and ast_snps_axi5_rready_max_waits checks maximum wait cycles of 'rready'.

1. The following diagram shows an example for the relation between write address channel and write data channel:

**Figure 5-10 Example for the Relation Between Write Address Channel and Write Data Channel**



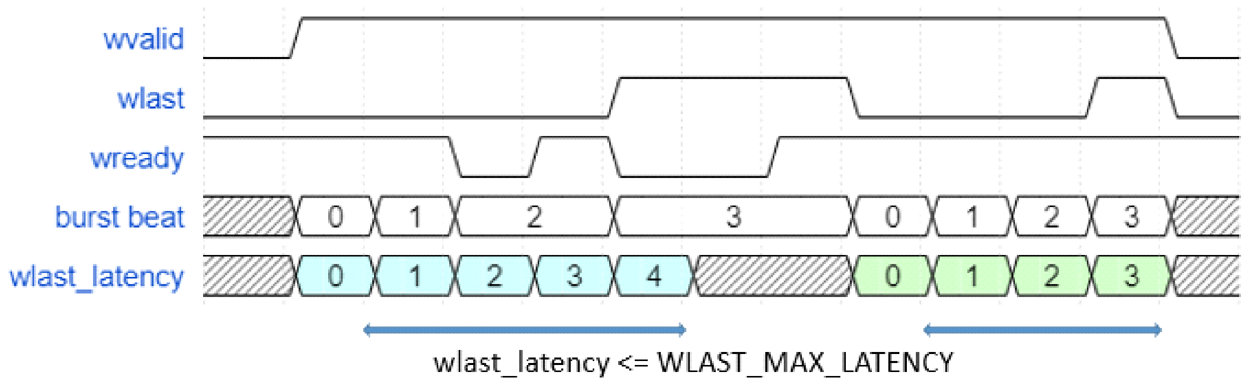
The assertion ast_snps_axi5_aw_w_max_latency checks if the latency from the start of write address channel to the start of write data channel should be within AW_W_MAX_LATENCY cycles when write address channel is issued before write data channel.

The assertion ast_snps_axi5_w_aw_max_latency checks if the latency from the start of write data channel to the start of write address channel should be within W_AW_MAX_LATENCY cycles when write data channel is issued before write address channel.

Note that these checks don't care 'awready' and 'wready'.

The following diagram shows an example for the relation between 'wvalid' and 'wlast'.

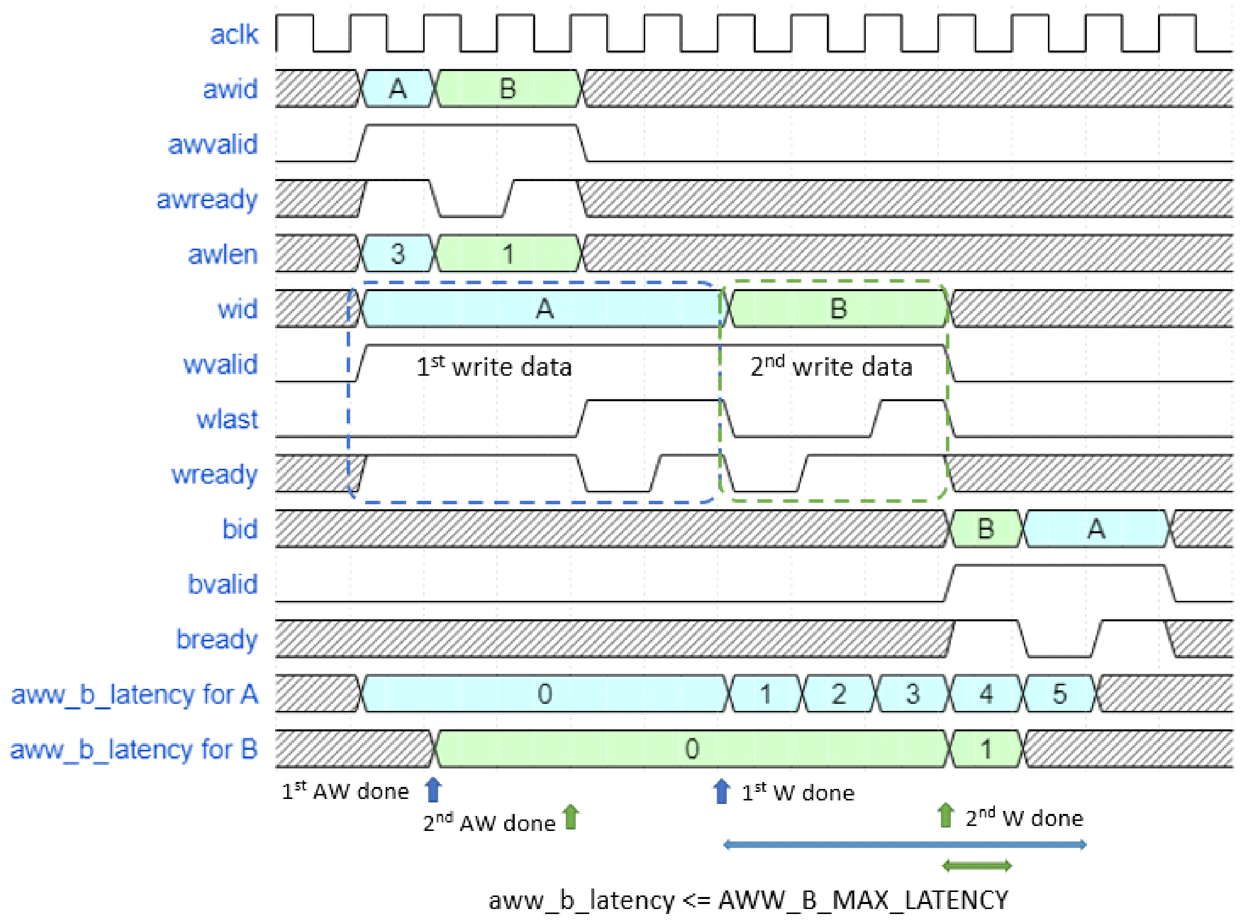
**Figure 5-11 Example for the Relation Between wvalid and wlast**



The assertion ast_snps_axi5_wlast_max_latency checks if the latency from the start of write data channel to the last beat of write data should be within WLAST_MAX_LATENCY cycles.

The following diagram shows and example for the relation between the completion of write address/data channels and write response:

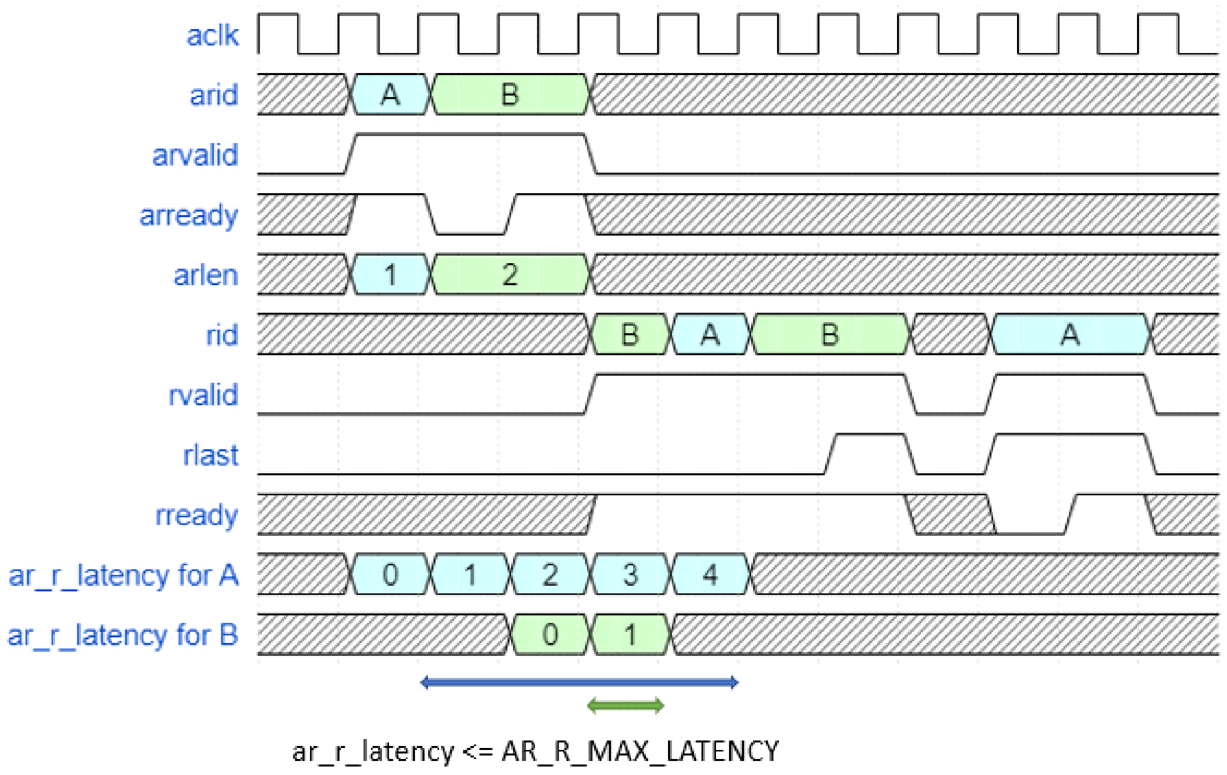
**Figure 5-12 Example for the Relation Between the Completion of Write Address/Data Channels and Write Response**



The assertion ast_snps_axi5_aww_b_max_latency checks if the latency from the completion of both write address and data channels to write response channel should be within AWW_B_MAX_LATENCY cycles.

The following diagram shows and example for the relation between the completion of read address channel and the start of read response channel:

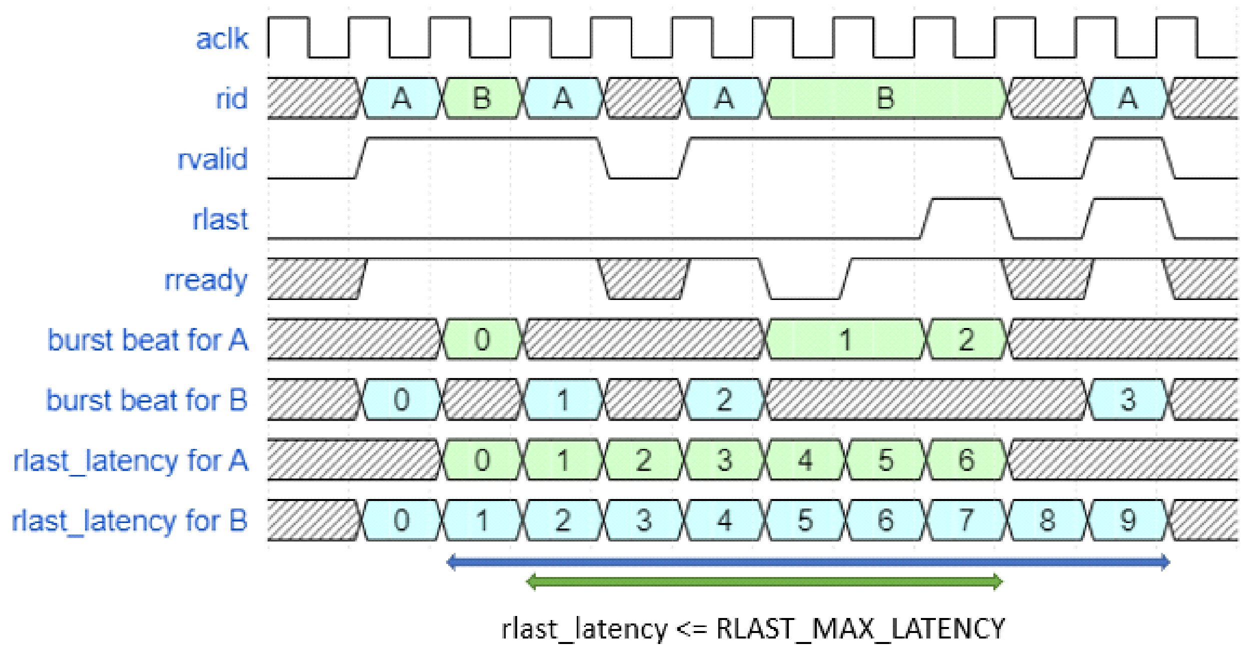
**Figure 5-13 Example for the Relation Between the Completion of Read Address Channel and the Start of Read Response Channel**



The assertion ast_snps_axi5_ar_r_max_latency checks if the latency from the completion of read address channel to the start of read response channel should be within AR_R_MAX_LATENCY cycles.

The following diagram shows an example for the relation between the start of read response channel and the last beat of read response channel:

**Figure 5-14 Example for the Relation Between the Start of Read Response Channel and the Last Beat of Read Response Channel**



The assertion ast_snps_axi5_rlast_max_latency checks if the latency from the start of read response channel to the last beat of read response channel should be within RLAST_MAX_LATENCY cycles.

## 5.2.3 Deadlock Properties Recommended Configurations

The configuration is straight forward, however, it requires careful consideration depending on DUT and test bench.

If the maximum latency in DUT is known and DUT returns response itself without other dependency, use safety property, that is, specify *_MAX_WAITS or *_MAX_LATENCY with positive integer.

For example, DUT returns ‘awready’ within 3 cycles and ‘wready’ within 2 cycles, parameters can be set AW_MAX_WAITS with 3 and W_MAX_WAITS with 2.

If the maximum latency in DUT is known, but return of response depends on other interface latency, there are 2 possible cases:

- Use safety property by setting requester latency bigger than responder latency. Please refer Example 1.
- Use liveness property for both requester and respond.

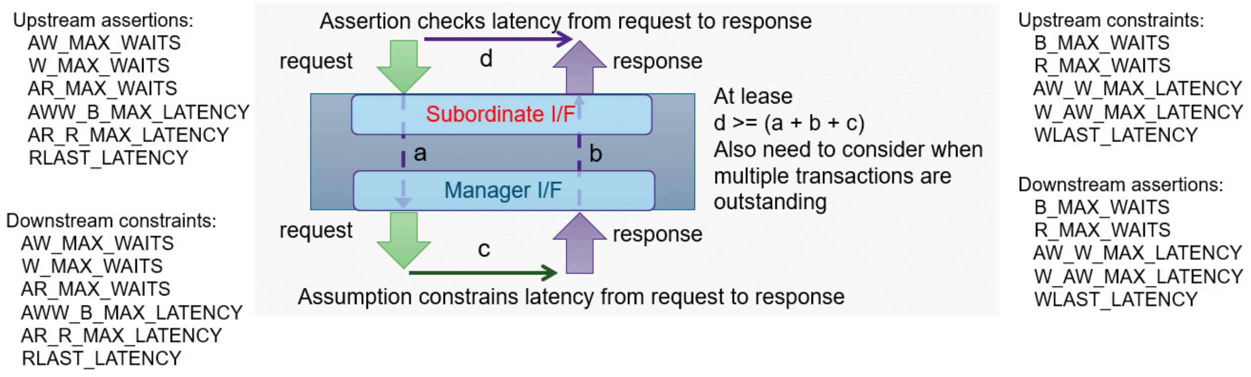
For the bug hunting purpose, when returning of response depends on other interface latency, try 2 types of configurations:

- Minimize response-related latencies to increase the maximum number of transactions (constraints with safety properties in responder), and use liveness assertions. Please refer Example 2.
- Use liveness properties for both requester and responder.

#### 5.2.3.0.1 Example 1

Use safety property by setting requester latency bigger than responder latency.

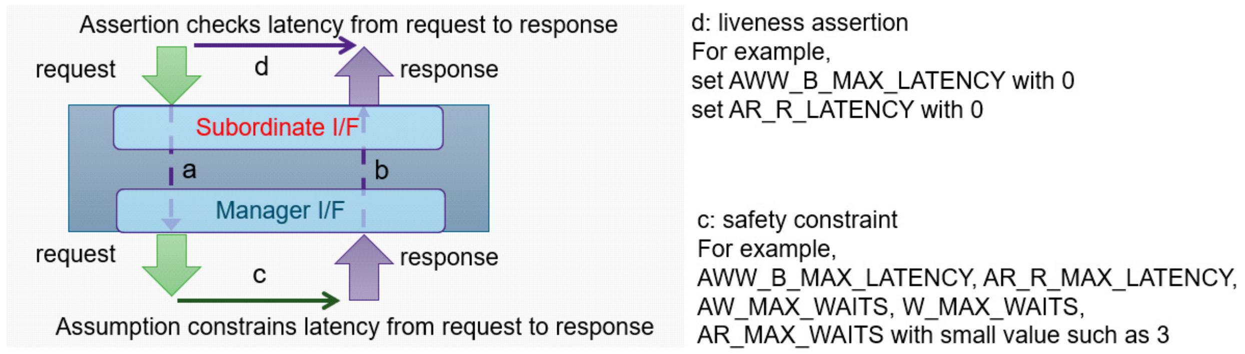
**Figure 5-15 Setting Requester Latency Bigger Than Responder Latency**



#### 5.2.3.0.2 Example 2

Minimize safety constraints latencies and check with liveness assertions.

**Figure 5-16 Minimize Safety Constraints Latencies**


Assertion checks latency from request to response
request
d
response
Subordinate I/F
a
b
Manager I/F
request
c
response
Assumption constrains latency from request to response
d: liveness assertion
For example,
set AWW_B_MAX_LATENCY with 0
set AR_R_LATENCY with 0
c: safety constraint
For example,
AWW_B_MAX_LATENCY, AR_R_MAX_LATENCY,
AW_MAX_WAITS, W_MAX_WAITS,
AR_MAX_WAITS with small value such as 3

### 5.2.4 Configurations to Improve Convergence

There is no known load to improve convergence, however, AIP has formal optimized assertions and there are some tips to improve convergence in configuration.

Most formal optimized assertions are enabled by default (when CHECK_FORMAL = 1). There are still some other parameters possible to improve convergence depending on case. These settings could be trade off between preciseness and better performance.

**Table 5-3 Configurations to Improve Convergence**

| Parameter Name | Default | Comments |
|---|---|---|
| CONFIG_WSTRB | 1 | If WSTRB related checks are not required or not important, please set with 0. WSTRB checks ‘wstrb_align’ and ‘wstrb_fixed’ are complexed. |

Table 5-3 **Configurations to Improve Convergence**

| Parameter Name | Default | Comments |
|---|---|---|
| WDATA_STROBE | 1 | When these parameters are 1, valid byte lanes are checked. When these parameters are 0, all data bits are checked. Valid byte lanes calculation is complexed, and convergence is much better when set with 0. |
| RDATA_STROBE | 1 | |

Note that there is exception for applying formal optimized assertions. For example, if DUT is AXI-AXI bridge and passing through transactions between 2 interfaces, it may be worth to try setting CHECK_FORMAL with 0. In this case, 2 AIPs are instantiated in both interfaces. Therefore, AIP assertions in one side check AIP constraints in other side. Also, the properties used as constraints in one side and the properties used as assertions in other side are exactly same.