AXI4 Assertion IP
User Guide

Version 2023.03, March 2023

SYNOPSYS®

Copyright Notice and Proprietary Information

Contents

Figures

Tables

Preface

This guide contains installation, setup, and usage instructions for the AMBA AXI4 SystemVerilog AIP. It is intended for use by the design or verification engineers who want to verify RTL designs with an AMBA AXI4 interface. Readers are assumed to be familiar with the AMBA AXI4 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 DesignWare (DW) AXI4 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 DesignWare (DW) AXI4 AIP.

Chapter 3,“Using AXI4 AIP in a Formal Environment”, introduces the VC Formal tool and the AIP usage in a formal environment.

Chapter 4, “The AXI4 AIP Configuration”, presents an insight into the functionality of the AXI4 AIP.

Chapter 5, “The AXI4 AIP Use Cases”, presents an example of how to install and use the Synopsys AMBA AXI4 AIP.

Web Resources

The AXI4 AIP is compliant with the following specifications:

- AMBA specification ARM IHI 0022E (ID022613)
- 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 Static
  - **Sub Product 1**: Formal
  - **Product Version**: 2016.06-SP1
  - 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 AXI4 protocol checkers available in the AXI4 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 AXI4 protocol, SystemVerilog Assertions and the SystemVerilog language.

1.2 References

AXI4 AIP is compliant with the following specifications:

- AMBA specification ARM IHI 0022E (ID022613)
- AMBA FAQ document

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

1.3 Product Overview

The AXI4 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 AXI4 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 AXI4 AIP suite supports the following languages and methodology:

- SystemVerilog Assertion
- Verilog

1.5 Feature Support

1.5.1 Protocol Features

The AXI4 AIP currently supports the following protocol functions:

- All data widths
- All address widths
- All transfer types
- All burst types and burst lengths (up to 256 in AXI4 as opposed to 16 in AXI3)
- All protection types
- All slave response types
- Exclusive transactions

1.5.2 Verification Features

The AXI4 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

- None

2 Installation and Setup

This chapter guides you through installing and setting up the AXI4 AIP. When you complete the checklist mentioned below, the provided example gets operational and you can use the AXI4 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 AXI4 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 AXI4 AIP requires.

- VCS version L-2016.06-SP1 (Simulator)
- Verdi version L-2016.06-SP1 (Debugger)
- VCF version L-2016.06-SP1 (Formal)
- VC version L-2016.06-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 AXI4 AIP requires the following license feature:

- AIP-AXI4-SVA

The following topics describe the required environment variables and path settings for the AXI4 AIP:

2.2.2.1 Running the AXI4 AIP on the VC Formal Tool

To run the AXI4 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 AXI4 AIP on VCS

To run the AXI4 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	AXI4, AXI4-Lite and AXI3
VC Formal AIP AMBA ACE	AIP-ACE-SVA	ACE, ACE-Lite, AXI4, AXI4-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 AXI4 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 AXI4 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 AXI4 AIP, see "Synopsys Common Licensing (SCL) Software" on page 15.

3 Using AXI4 AIP in a Formal Environment

This chapter describes the simulation environment for the AXI4 AIP, usage of the VC Formal tool, and the AXI4 AIP usage in a formal verification environment. This chapter discusses the following topics:

- "Introduction to the VC Formal Tool" on page 19
- "The AXI4 AIP in a Formal Environment" on page 19
- "Clock and Reset Functionality" on page 27

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 for the signals that it monitors. These signal values can be constrained either by the DUT that is driving them or by the assume properties in an AIP or by a combination of both. The VC Formal tool provides the following information:

- Number of proven properties
- Number of falsified properties
- Number of vacuous properties
- Number of covered 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 AXI4 AIP source code
- The path of the RTL source code (if validating the RTL)
- Clock information
- Reset information
- Timeout details

3.2 The AXI4 AIP in a Formal Environment

The AXI4 AIP has AXI4 properties to verify either an AXI4 Master or an AXI4 slave. These properties are connected to either an AXI4 master or a slave module (for example, AXI4 slave DUT to master properties). To verify the functional correctness of the module, use the VC Formal verification tool.

To use the AXI4 AIP in a formal verification environment, perform the following steps in a sequence:

- Instantiating the AXI4 AIP using the `bind` statement
- Creating a Tcl file
- Reading and running a Tcl file
- Analyzing results

3.2.1 Instantiating the AXI4 AIP Using bind Statement

Create the bind file to instantiate the AXI4 AIP and bind the AXI4 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 AXI4 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 `2’h0`.

3.2.2 Creating a Tcl File

To compile the DUT and the attached AXI4 AIP (including RTL files, AIP files, and the bind file), create a Tcl file, where path to RTL directory (AXI4 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.

Table 3-1 Tcl File Example

```
set AIP_HOME $::env(AIP_HOME)
```

Set AIP path

```
# Source code and tb and Tcl path
set AIP_SRC_DIR ${AIP_HOME}/esrc
set TEST_DIR ${AIP_HOME}/examples
set TB_DIR ${TEST_DIR}/B2B_SELF/tb
set TCL_DIR $AIP_HOME/tests/B2B_SELF/tcl

if { [ file exists setup.tcl ] == 1 } {
  source setup.tcl -echo -verbose
}
set hierarchy_delimiter "."

# Timeout settings
set_fml_var fml_max_time 5M
```

Either use interactive debugging command or batch regression command on the basis of your requirement.

```
# Commands to run the files in VC Static
proc compd {} {
     global AIP_HOME TEST_DIR TBH_DIR TCL_DIR
     read_file -sva -top dummy_dut_1m1s -format
sverilog -vcs "
     -sverilog +incdir+${AIP_HOME}/include
     +incdir+${AIP_SRC_DIR}
     ${AIP_SRC_DIR}/snps_axi_aip_pkg.sv
     ${AIP_SRC_DIR}/snps_axi4_aip.sv
     ${TB_DIR}/dummy_dut_1m1s.v
     ${TB_DIR}/bind_1m1s.sv
     -parameters ${TCL_DIR}/config1.param
     -assert svaext  " }
# Initialization -- reset sequence
proc initd {} {
create_clock <design clk> -period 100
create_reset <design reset> -low
sim_config -rst_wave ON -replay_all_wave ON
sim_run -stable
sim_save_reset
}
compd
initd
# Generate logs when running a batch regression
check_fv -block
report_fv
# Generate logs during interactive debugging
check_fv -run_finish
report_fv -list > $RUN_DIR/run_config1.log
```

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 24

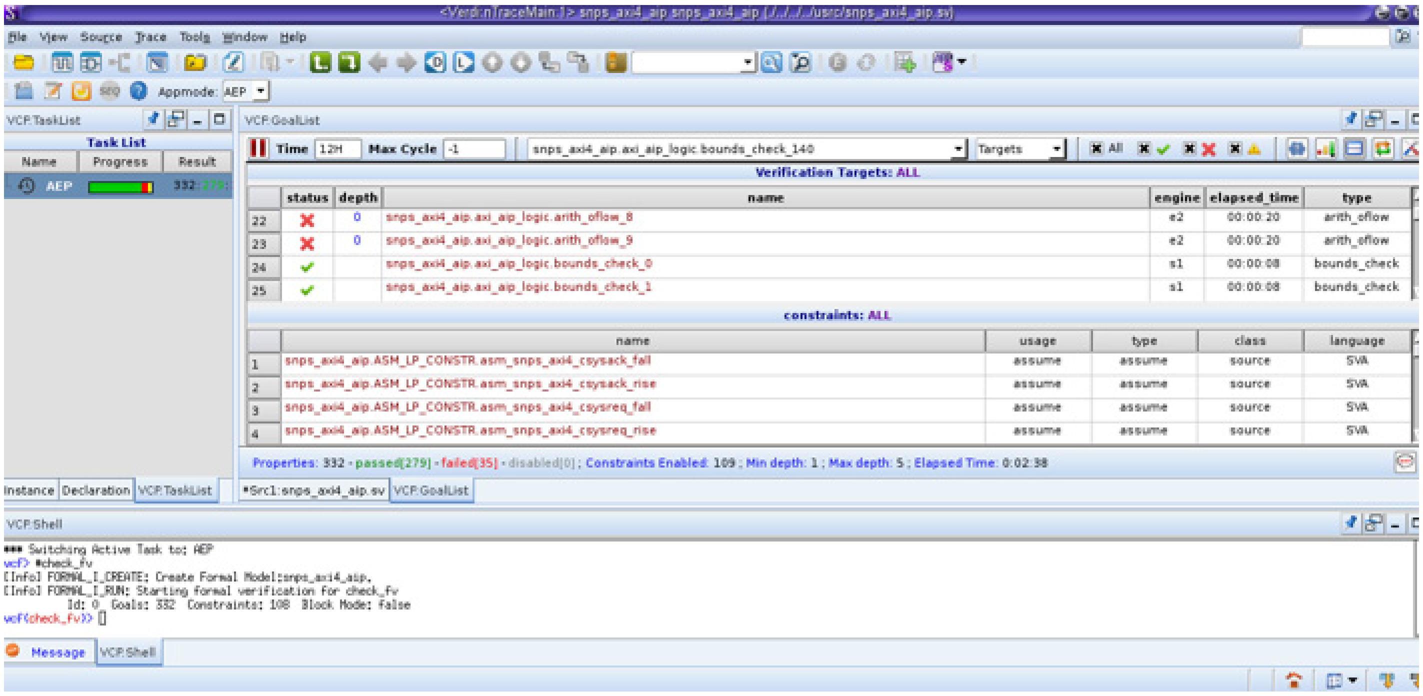
3.2.3.1 GUI Mode

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



<Verdi:nTraceMain:1> snps_axi4_aip.snps_axi4_aip (/../../../usrc/snps_axi4_aip.sv)
File View Source Trace Tools Window Help
Appmode: AEP
VCF:TaskList
Task List
VCF:GoalList
Time 12H
Max Cycle -1
snps_axi4_aip.axi_aip_logic.bounds_check_140
Targets
All
Instance Declaration VCF:TaskList
*Src1:snps_axi4_aip.sv VCF:GoalList
VCF:Shell
Message VCF:Shell

Name	Progress	Result
AEP		332: 279

Verification Targets: ALL

	status	depth	name	engine	elapsed_time	type
22	✗	0	snps_axi4_aip.axi_aip_logic.arith_oflow_8	e2	00:00:20	arith_oflow
23	✗	0	snps_axi4_aip.axi_aip_logic.arith_oflow_9	e2	00:00:20	arith_oflow
24	✓		snps_axi4_aip.axi_aip_logic.bounds_check_0	s1	00:00:08	bounds_check
25	✓		snps_axi4_aip.axi_aip_logic.bounds_check_1	s1	00:00:08	bounds_check

constraints: ALL

	name	usage	type	class	language
1	snps_axi4_aip.ASM_LP_CONSTR.asm_snps_axi4_csysack_fall	assume	assume	source	SVA
2	snps_axi4_aip.ASM_LP_CONSTR.asm_snps_axi4_csysack_rise	assume	assume	source	SVA
3	snps_axi4_aip.ASM_LP_CONSTR.asm_snps_axi4_csysreq_fall	assume	assume	source	SVA
4	snps_axi4_aip.ASM_LP_CONSTR.asm_snps_axi4_csysreq_rise	assume	assume	source	SVA

Properties: 332 • passed[279] • failed[35] • disabled[0] ; Constraints Enabled: 109 ; Min depth: 1 ; Max depth: 5 ; Elapsed Time: 0:02:38

```
*** Switching Active Task to: AEP
vcf> #check_fv
[Info] FORMAL_I_CREATE: Create Formal Model:snps_axi4_aip.
[Info] FORMAL_I_RUN: Starting formal verification for check_fv
        Id: 0  Goals: 332  Constraints: 108  Block Mode: false
vcf(check_fv)> []
```

3.2.3.1.1 Analyzing Results for the GUI Mode Run

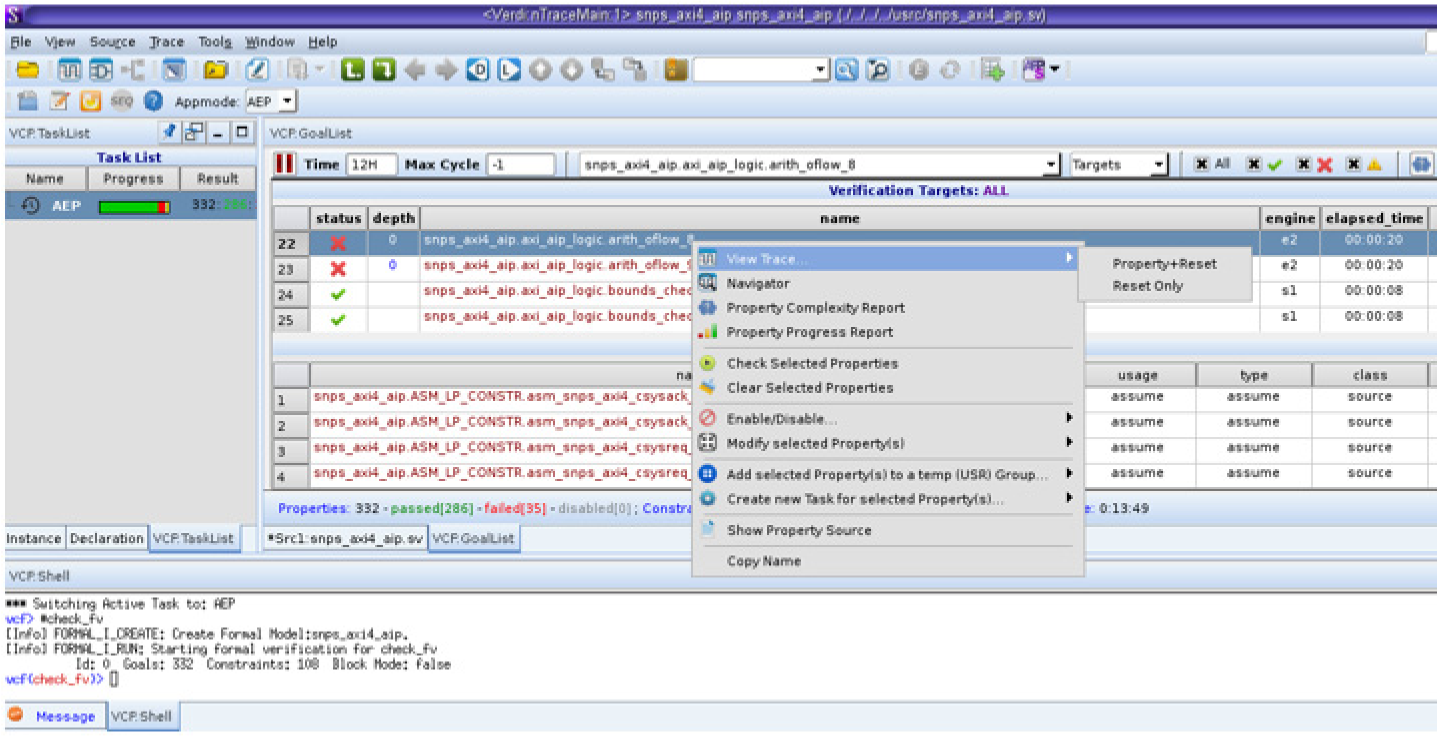
After running a session, its results are dumped into the `vcf.log` file. This log file gives the number of proven, falsified, vacuous, inconclusive, and covered properties. When there are no falsifications, the design is verified with regard to the parameters configured in a Tcl file.

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 debug 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



<Verdi:nTraceMain:1> snps_axi4_aip snps_axi4_aip (/../../../usrc/snps_axi4_aip.sv)
File View Source Trace Tools Window Help
Appmode: AEP
VCF:TaskList
Task List
VCF:GoalList
View Trace...
Property+Reset
Reset Only
Navigator
Property Complexity Report
Property Progress Report
Check Selected Properties
Clear Selected Properties
Enable/Disable...
Modify selected Property(s)
Add selected Property(s) to a temp (USR) Group...
Create new Task for selected Property(s)...
Show Property Source
Copy Name
Instance
Declaration
*Src1:snps_axi4_aip.sv
VCF:Shell
Message

Name	Progress	Result
AEP		332

Time 12H Max Cycle -1 snps_axi4_aip.axi_aip_logic.arith_oflow_8 Targets

Verification Targets: ALL

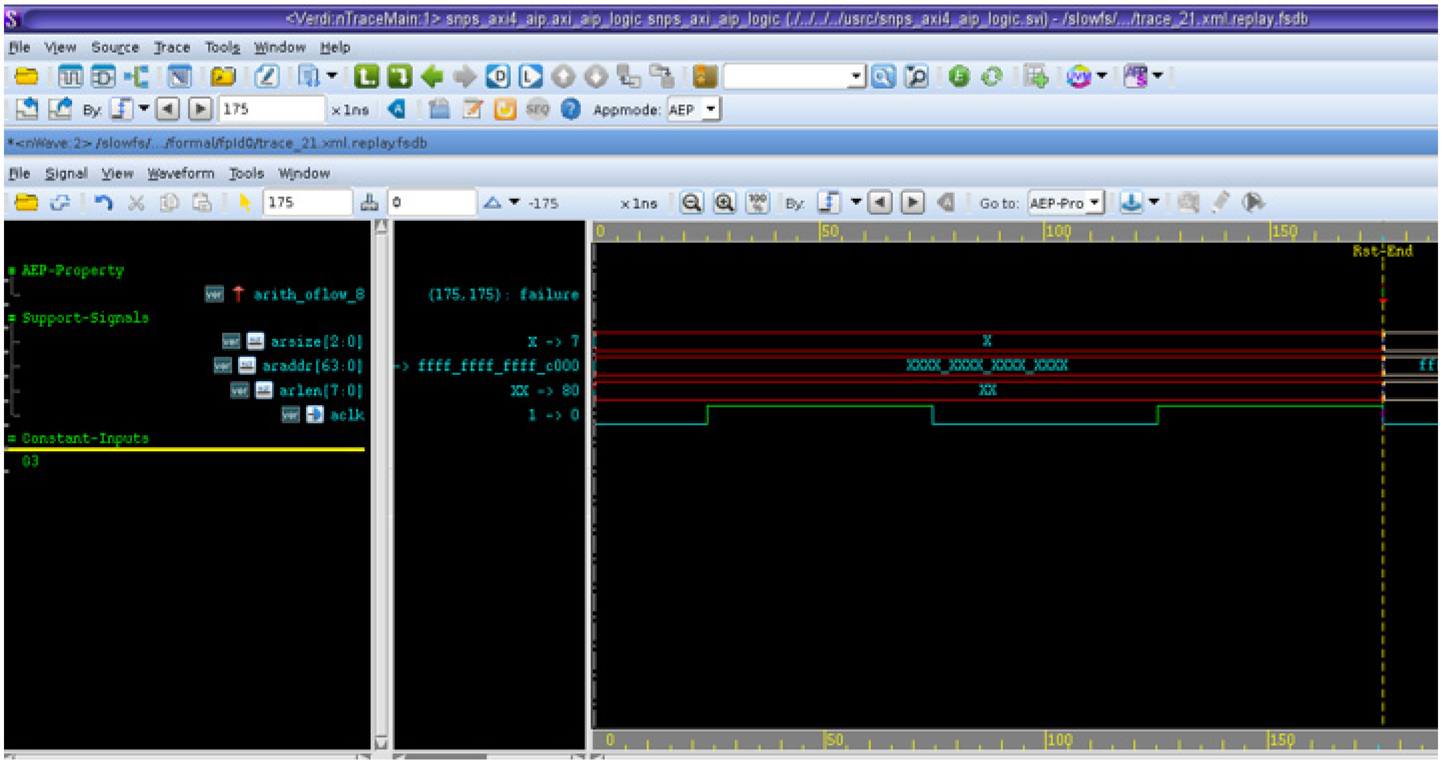
	status	depth	name	engine	elapsed_time
22	✗	0	snps_axi4_aip.axi_aip_logic.arith_oflow_8	e2	00:00:20
23	✗	0	snps_axi4_aip.axi_aip_logic.arith_oflow_	e2	00:00:20
24	✓		snps_axi4_aip.axi_aip_logic.bounds_chec	s1	00:00:08
25	✓		snps_axi4_aip.axi_aip_logic.bounds_chec	s1	00:00:08

	na	usage	type	class
1	snps_axi4_aip.ASM_LP_CONSTR.asm_snps_axi4_csysack	assume	assume	source
2	snps_axi4_aip.ASM_LP_CONSTR.asm_snps_axi4_csysack	assume	assume	source
3	snps_axi4_aip.ASM_LP_CONSTR.asm_snps_axi4_csysreq	assume	assume	source
4	snps_axi4_aip.ASM_LP_CONSTR.asm_snps_axi4_csysreq	assume	assume	source

Properties: 332 - passed[286] - failed[35] - disabled[0] ; Constra ... e: 0:13:49

```
*** Switching Active Task to: AEP
vcf> #check_fv
[Info] FORMAL_I_CREATE: Create Formal Model:snps_axi4_aip.
[Info] FORMAL_I_RUN: Starting formal verification for check_fv
        Id: 0  Goals: 332  Constraints: 108  Block Mode: false
vcf(check_fv)> []
```

Figure 3-3 Signal Behavior in Waves



<Verdi:nTraceMain:1> snps_axi4_aip.axi_aip_logic snps_axi_aip_logic (/../../../usrc/snps_axi4_aip_logic.sv) - /slowfs/.../trace_21.xml.replay.fsdb
File View Source Trace Tools Window Help
By: 175 x 1ns Appmode: AEP
*<nWave:2> /slowfs/.../formal/fpld0/trace_21.xml.replay.fsdb
File Signal View Waveform Tools Window
175 0 -175 x 1ns By: Go to: AEP-Pro
AEP-Property
arith_oflow_8
Support-Signals
arsize[2:0]
araddr[63:0]
arlen[7:0]
aclk
Constant-Inputs
03
(175,175): failure
X -> 7
-> ffff_ffff_ffff_c000
XX -> 80
1 -> 0
0
50
100
150
Rst-End
X
XXXX_XXXX_XXXX_XXXX
XX
ff

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 that are specified in the VC Formal Tcl file.

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`

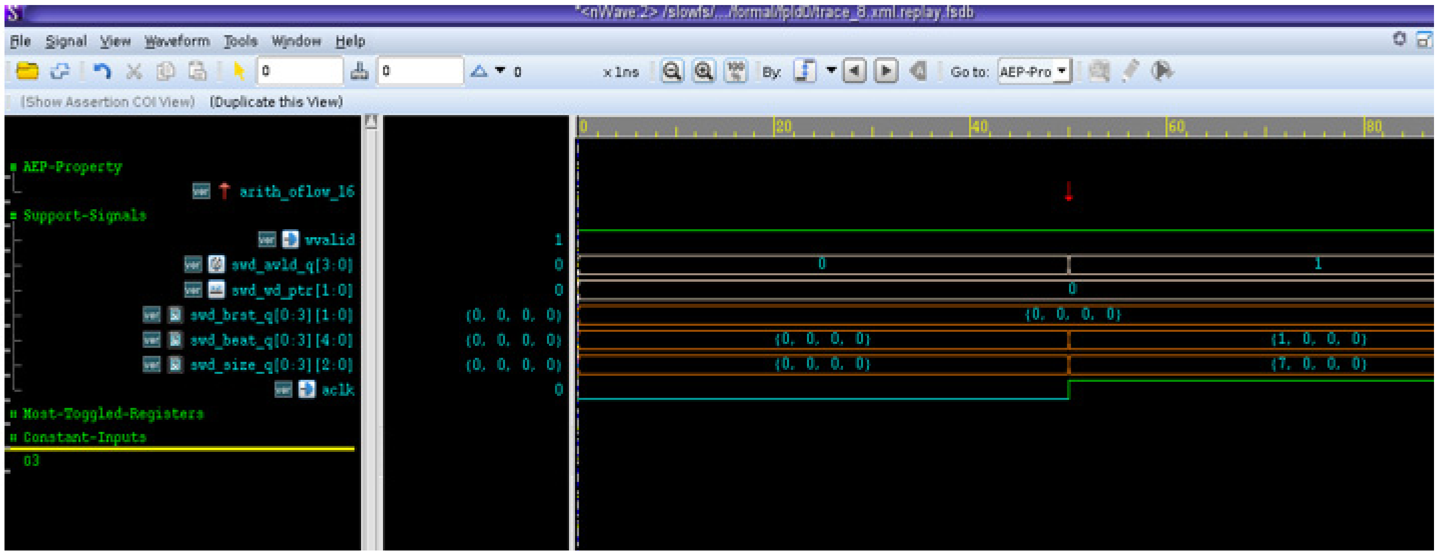
   To display the number of falsified properties.

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.

Figure 3-4 shows the waves opened in the batch mode for analyzing the results.

Figure 3-4 Waves Opened in the Batch Mode



3.2.4 Commonly Used AXI4 AIP Configurations

Table 3-2 Common Usage Models

1	Master	Instantiates the AXI4 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 AXI4 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 AXI4 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 AXI4 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 AXI4 AIP As a Master

To verify an AXI4 slave DUT, set the AGENT_TYPE parameter to MASTER during an AIP instantiation.

When the AXI4 AIP parameter is set as MASTER, all the AXI4 AIP properties that are related to the master inputs are declared as assertions, and all the AXI4 AIP properties that are related to the master outputs are declared as assumptions. This is required to make the AXI4 AIP AIP behave as a master.

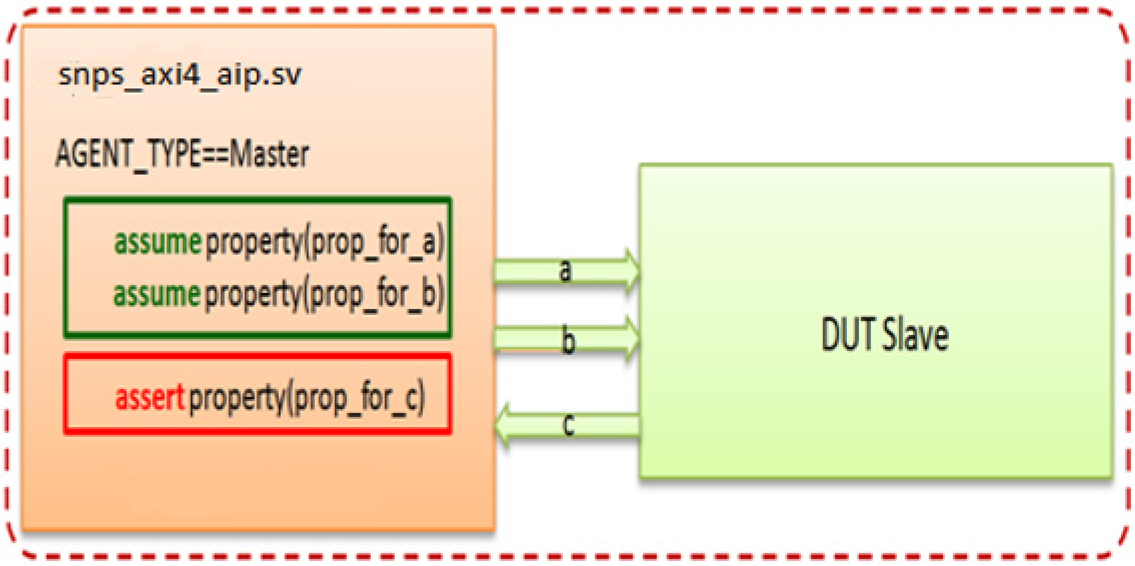
To disable all assert, assume, or cover properties, explicitly set the following parameters to value 0:

- ENABLE_ASSERT
- ENABLE_ASSUME
- ENABLE_COVER

For example, if you want to enable AXI4 slave checks (assert) only and do not want to apply constraints on AXI4 slave inputs (assume), set ENABLE_ASSERT=1 and ENABLE_ASSUME=0. This enables all assert properties related to an AXI4 slave DUT, but disables all assume properties.

Figure 3-5 checks the behavior of an AXI4 slave DUT output and constraints the inputs of an AXI4 slave DUT to valid values.

Figure 3-5 AXI4 AIP As a Master


snps_axi4_aip.sv
AGENT_TYPE==Master
assume property(prop_for_a)
assume property(prop_for_b)
assert property(prop_for_c)
a
b
c
DUT Slave

3.2.6 The AXI4 AIP As a Slave

To verify an AXI4 master DUT, set the `AGENT_TYPE` parameter to `SLAVE` during the AXI4 AIP instantiation. When this parameter is set as `SLAVE`, all the AXI4 AIP properties that are related to the slave inputs are declared as assertions, and all the AXI4 AIP properties that are related to the slave outputs are declared as assumptions. This is required to make the AXI4 AIP behave as a slave.

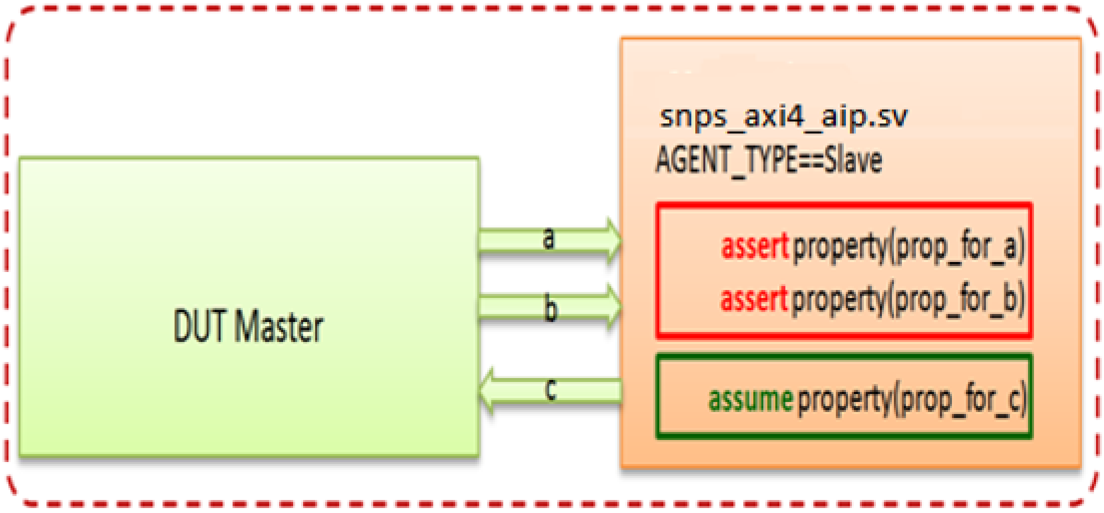
To disable all assert, assume, or cover properties, explicitly set the following parameters to value 0:

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

For example, if you want to enable the AXI4 slave checks (assert) only and do not want to apply constraints on the AXI4 master inputs (assume), set `ENABLE_ASSERT=1` and `ENABLE_ASSUME=0`. This enables all assert properties related to an AXI4 master DUT, but disables all assume properties.

Figure 3-6 checks the behavior of an AXI4 master DUT output and constraints the inputs of the AXI4 Master DUT to valid values.

Figure 3-6 AXI4 AIP As a Slave


snps_axi4_aip.sv
AGENT_TYPE==Slave
DUT Master
a
b
c
assert property(prop_for_a)
assert property(prop_for_b)
assume property(prop_for_c)

3.2.7 The AXI4 AIP As a Monitor

To verify the behavior of an AXI4 master and slave DUT, set the `AGENT_TYPE` parameter to `MONITOR` during the AIP instantiation. When this parameter is set to `MONITOR`, AXI4 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.

AXI4 AIP Use Case As A Monitor

In an RTL verification environment, the AXI4 AIP can be instantiated as a monitor on each AXI4 interface to check for protocol correctness.

3.2.8 The AXI4 AIP As a Constraint Model

If the `AGENT_TYPE` parameter is set to `CONSTRAINT`, the AXI4 AIP is instantiated to constraint the DUT slave input and DUT master input.

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

AXI4 AIP Use Case As A Constraint

To verify the RTL in a formal verification environment, the AXI4 AIP can be used in the constraint mode to generate stimulus to the RTL. You can connect the AXI4 AIP from other vendors to perform protocol checks.

3.3 Clock and Reset Functionality

- To run the AXI4 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 AXI4 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 AXI4 AIP Configuration

This chapter describes about configuration of the AXI4 AIP in the following Sections:

4.1 The AXI4 AIP Configuration Parameters

Table 4-1 The AXI4 AIP Configuration Parameters

Parameter	Default Value	Description
AGENT_TYPE	MASTER	Agent Type one of MASTER SLAVE MONITOR CONSTRAINT
READ_WRITE	BOTHRW	Check type: BOTHRW (Read and Write) or RDONLY (Read Only) or WRONLY (Write Only)
ENABLE_ASSERT	1	1: Enable 0: Disable Assertions
ENABLE_ASSUME	1	1: Enable 0: Disable Assumptions
ENABLE_COVER	1	1: Enable 0: Disable Cover Properties
CHECK_FORMAL	1	1: Use Formal 0: Use Simulation/Emulation
CHECK_PARAMETERS	0	Indicate check if parameter setting or not
CONFIG_USER	1	Indicate if use AWUSER/WUSER/BUSER/ARUSER/RUSER signals or not
CONFIG_LOWPOWER	1	Indicate check if Low Power properties or not (1: check 0: no check)

Table 4-1 The AXI4 AIP Configuration Parameters

CONFIG_WAITS	1	Indicate check if Valid/Ready wait cycles properties or not (1: check 0: no check)
CONFIG_X_CHECK	1	Indicate check if X signal properties or not (1: check 0: no check)
CONFIG_RECOMMEND	1	Indicate check if ARM recommendation properties or not (1: check 0: no check)
CONFIG_WSTRB	1	Indicate check if Write Strobe related properties (1: check 0: no check)
CONFIG_MAXOUTS	1	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.
CONFIG_QOS	1	Indicate if AWQOS/ARQOS signals are used or not.
CONFIG_REGION	1	Indicate if use AWREGION/ARREGION signals are used or not.
EXCL_DEPTH	4	Indicate Exclusive monitoring depth in slave (>=1: Exclusive depth 0: No Exclusive support)
RD_MAX_BURSTS	4	Maximum number of outstanding Read burst
WR_MAX_BURSTS	4	Maximum number of outstanding Write burst
MAXBURSTLENGTH	256	Maximum configured burst length
AW_MAX_WAITS	16	Maximum number of wait cycle from AWVALID to AWREADY, if set 0, liveness assertion or fairness constraint will be generated.
W_MAX_WAITS	16	Maximum number of wait cycle from WVALID to WREADY, if set 0, liveness assertion or fairness constraint will be generated.
B_MAX_WAITS	16	Maximum number of wait cycle from BVALID to BREADY, if set 0, liveness assertion or fairness constraint will be generated.
AR_MAX_WAITS	16	Maximum number of wait cycle from ARVALID to ARREADY, if set 0, liveness assertion or fairness constraint will be generated.
R_MAX_WAITS	16	Maximum number of wait cycle from RVALID to RREADY, if set 0, liveness assertion or fairness constraint will be generated.
WR_OUT_OF_ORDER	1	Indicate if support Write Out Of Order Response or not (1: supported 0: not allow out of order)
RD_INTERLEAVE	1	Indicate if support Read Data Interleave or not (1: allow interleaving 0: not allow interleaving)
RD_OUT_OF_ORDER	1	Indicate if support Read Out Of Order Data or not (1: supported 0: not allow out of order)
WDATA_ADVANCE	1	Indicate if Write Data can be issued earlier than Write Address or not (1: possible 0: not allow)

Table 4-1 The AXI4 AIP Configuration Parameters

WR_ALLOW_DECERR	1	Indicate if DECERR is possible or not as write response (BRESP)
WR_ALLOW_SLVERR	1	Indicate if SLVERR is possible or not as write response (BRESP)
RD_ALLOW_DECERR	1	Indicate if DECERR is possible or not as read response (RRESP)
RD_ALLOW_SLVERR	1	Indicate if SLVERR is possible or not as read response (RRESP)
WDATA_WIDTH	128	Write Data bus bit width
RDATA_WIDTH	128	Read Data bus bit width
ADDR_WIDTH	64	Address bus bit width
ID_WIDTH	8	ID bit width
LEN_WIDTH	8	Length bit width
SIZE_WIDTH	3	Size bit width
AWUSER_WIDTH	32	AWUSER user signal bit width
WUSER_WIDTH	32	WUSER user signal bit width
BUSER_WIDTH	32	BUSER user signal bit width
ARUSER_WIDTH	32	ARUSER user signal bit width
RUSER_WIDTH	32	RUSER user signal bit width
ADDR_RANGE	1	number of address ranges
MIN_ADDR	{32'b0}	Minimum address for each address range: for example {32'h0, 32'h8000} in case of ADDR_RANGE=2
MAX_ADDR	{32'hffffffff}	Maximum address for each address range: for example {32'h3fff, 32'hefffffff} in case of ADDR_RANGE=2
CONFIG_BASE	1	Indicate check if basic properties or not (1: enable all basic properties 0: disable basic properties)
INTERCHANNEL_LATENCY	0	Indicate check if Inter-channel latency properties or not (1: check 0: no check)
AW_W_MAX_LATENCY	16	Maximum latency from AWVALID to WVALID, if set 0, liveness assertion or fairness constraint will be generated.
W_AW_MAX_LATENCY	16	Maximum latency from WVALID to AWVALID, if set 0, liveness assertion or fairness constraint will be generated.
AWW_B_MAX_LATENCY	16	Maximum latency from address/data channels complete to BVALID, if set 0, liveness assertion or fairness constraint will be generated.
WLAST_MAX_LATENCY	16	Maximum latency from WVALID to WLAST, if set 0, liveness assertion or fairness constraint will be generated.

Table 4-1 The AXI4 AIP Configuration Parameters

AR_R_MAX_LATENCY	16	Maximum latency from ARVALID to RVALID, if set 0, liveness assertion or fairness constraint will be generated.
RLAST_MAX_LATENCY	16	Maximum latency from RVALID to RLAST, if set 0, liveness assertion or fairness constraint will be generated.

4.2 The AXI4 AIP Interface Ports

This Section describes interface signals of the AXI4 AIP:

Table 4-2 The AXI4 AIP Interface Ports

	Signal Name	Description	
		Source	**Destination**
1	aclk	Input clock from the system	
2	aresetn	Reset input from the system	
3	awid	Master	Slave
4	awaddr	Master	Slave
5	awlen	Master	Slave
6	awsize	Master	Slave
7	awburst	Master	Slave
8	awcache	Master	Slave
9	awprot	Master	Slave
10	awlock	Master	Slave
11	awuser	Master	Slave
12	awvalid	Master	Slave
13	awready	Slave	Master
15	wdata	Master	Slave
16	wstrb	Master	Slave
17	wuser	Master	Slave
18	wlast	Master	Slave
19	wvalid	Master	Slave
20	wready	Slave	Master
21	bid	Slave	Master

Table 4-2 The AXI4 AIP Interface Ports

	Signal Name	Description	
22	bresp	Slave	Master
23	buser	Slave	Master
24	bvalid	Slave	Master
25	bready	Master	Slave
26	arid	Master	Slave
27	araddr	Master	Slave
28	arlen	Master	Slave
29	arsize	Master	Slave
30	arburst	Master	Slave
31	arcache	Master	Slave
32	arprot	Master	Slave
33	arlock	Master	Slave
34	aruser	Master	Slave
35	arvalid	Master	Slave
36	arready	Slave	Master
37	rid	Slave	Master
38	rdata	Slave	Master
39	rresp	Slave	Master
40	ruser	Slave	Master
41	rlast	Slave	Master
42	rvalid	Slave	Master
43	rready	Master	Slave
44	cactive	System clock controller to the peripheral device	
45	csysreq	Peripheral device to system	
46	csysack	Peripheral device to system	

4.3 The AXI4 AIP Properties

Table 4-3 The AXI4 AIP Properties

Property Name	Agent	Severity	Spec Reference	Property Description
ast_snps_axi4_awaddr_boundary	MASTER	ERROR	[ARM IHI 0022E] section A3.4.1 on page A3-46.	A write burst cannot cross a 4kbyte boundary.
ast_snps_axi4_awaddr_wrap_align	MASTER	ERROR	[ARM IHI 0022E] section A3.4.1 on page A3-48.	For wrapping bursts, the start address must be aligned to the size of each transfer.
ast_snps_axi4_awburst_no_reserved	MASTER	ERROR	[ARM IHI 0022E] section A3.4.1 table A3-3 on page A3-48.	When AWVALID is high, a value of 2'b11 on AWBURST is reserved.
ast_snps_axi4_awcache_legal	MASTER	ERROR	[ARM IHI 0022E] section A4.4 table A4-5 on page A4-67.	When AWVALID is HIGH and AWCACHE[1] is LOW then AWCACHE[3:2] are also LOW
ast_snps_axi4_awlen_wrap	MASTER	ERROR	[ARM IHI 0022E] section A3.4.1 on page A3-46.	For wrapping bursts, the burst length must be 2, 4, 8, or 16.
ast_snps_axi4_awlen_fixed	MASTER	ERROR	[ARM IHI 0022E] section A3.4.1 on page A3-46.	FIXED burst type supports burst length 1 to 16.
ast_snps_axi4_awlen_lock	MASTER	ERROR	[ARM IHI 0022E] section A7.2.4 on page A7-97.	The burst length for an exclusive access must not exceed 16 transfers.
ast_snps_axi4_awsize_max	MASTER	ERROR	[ARM IHI 0022E] section A3.4.1 on page A3-47.	The size of any transfer must not exceed the data bus width of either agent in the transaction.
ast_snps_axi4_awvalid_reset	MASTER	ERROR	[ARM IHI 0022E] section A3.1.2 Figure A3-1 on page A3-38.	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.
ast_snps_axi4_awaddr_stable	MASTER	ERROR	[ARM IHI 0022E] section A3.2.1, and figure A3-2, on page A3-39.	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_axi4_awburst_stable	MASTER	ERROR	[ARM IHI 0022E] section A3.2.1, and figure A3-2, on page A3-39.	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.

Table 4-3 The AXI4 AIP Properties

Property Name	Agent	Severity	Spec Reference	Property Description
ast_snps_axi4_awid_stable	MASTER	ERROR	[ARM IHI 0022E] section A3.2.1, and figure A3-2, on page A3-39.	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_axi4_awlen_stable	MASTER	ERROR	[ARM IHI 0022E] section A3.2.1, and figure A3-2, on page A3-39.	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_axi4_awsize_stable	MASTER	ERROR	[ARM IHI 0022E] section A3.2.1, and figure A3-2, on page A3-39.	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_axi4_awlock_stable	MASTER	ERROR	[ARM IHI 0022E] section A3.2.1, and figure A3-2, on page A3-39.	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.
ast_snps_axi4_awcache_stable	MASTER	ERROR	[ARM IHI 0022E] section A3.2.1, and figure A3-2, on page A3-39.	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_axi4_awqos_stable	MASTER	ERROR	[ARM IHI 0022E] section A3.2.1, and figure A3-2, on page A3-39.	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_axi4_awregion_stable	MASTER	ERROR	[ARM IHI 0022E] section A3.2.1, and figure A3-2, on page A3-39.	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.

Table 4-3 The AXI4 AIP Properties

Property Name	Agent	Severity	Spec Reference	Property Description
ast_snps_axi4_awprot_stable	MASTER	ERROR	[ARM IHI 0022E] section A3.2.1, and figure A3-2, on page A3-39.	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_axi4_awvalid_stable	MASTER	ERROR	[ARM IHI 0022E] section A3.2.2 on page A3-40.	When asserted, AWVALID must remain asserted until the rising clock edge after the master asserts AWREADY.
ast_snps_axi4_wvalid_reset	MASTER	ERROR	[ARM IHI 0022E] section A3.1.2 Figure A3-1 on page A3-38.	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.
ast_snps_axi4_wdata_stable	MASTER	ERROR	[ARM IHI 0022E] section A3.2.1, and figure A3-2, on page A3-39.	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_axi4_wdata_stable_allbits	MASTER	ERROR	[ARM IHI 0022E] section A3.2.1, and figure A3-2, on page A3-39.	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_axi4_wlast_stable	MASTER	ERROR	[ARM IHI 0022E] section A3.2.1, and figure A3-2, on page A3-39.	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_axi4_wstrb_stable	MASTER	ERROR	[ARM IHI 0022E] section A3.2.1, and figure A3-2, on page A3-39.	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_axi4_wvalid_stable	MASTER	ERROR	[ARM IHI 0022E] section A3.2.2 on page A3-41.	When asserted, WVALID must remain asserted until the rising clock edge after the master asserts WREADY.
ast_snps_axi4_araddr_boundary	MASTER	ERROR	[ARM IHI 0022E] section A3.4.1 on page A3-46.	A burst must not cross a 4KB address boundary.

Table 4-3 The AXI4 AIP Properties

Property Name	Agent	Severity	Spec Reference	Property Description
ast_snps_axi4_araddr_wrap_align	MASTER	ERROR	[ARM IHI 0022E] section A3.4.1 on page A3-48.	For wrapping bursts, the start address must be aligned to the size of each transfer.
ast_snps_axi4_arburst_no_reserved	MASTER	ERROR	[ARM IHI 0022E] section A3.4.1 table A3-3 on page A3-48.	When ARVALID is high, a value of 2'b11 on ARBURST is reserved.
ast_snps_axi4_arcache_legal	MASTER	ERROR	[ARM IHI 0022E] section A4.4 table A4-5 on page A4-67.	When ARVALID is HIGH and ARCACHE[1] is LOW then ARCACHE[3:2] are also LOW
ast_snps_axi4_arlen_wrap	MASTER	ERROR	[ARM IHI 0022E] section A3.4.1 on page A3-46.	For wrapping bursts, the burst length must be 2, 4, 8, or 16.
ast_snps_axi4_arlen_fixed	MASTER	ERROR	[ARM IHI 0022E] section A3.4.1 on page A3-46.	FIXED burst type supports burst length 1 to 16.
ast_snps_axi4_arsize_max	MASTER	ERROR	[ARM IHI 0022E] section A3.4.1 on page A3-47.	The size of any transfer must not exceed the data bus width of either agent in the transaction.
ast_snps_axi4_arvalid_reset	MASTER	ERROR	[ARM IHI 0022E] section A3.1.2 Figure A3-1 on page A3-38.	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_axi4_araddr_stable	MASTER	ERROR	[ARM IHI 0022E] section A3.2.1, and figure A3-2, on page A3-39.	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_axi4_arburst_stable	MASTER	ERROR	[ARM IHI 0022E] section A3.2.1, and figure A3-2, on page A3-39.	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_axi4_arid_stable	MASTER	ERROR	[ARM IHI 0022E] section A3.2.1, and figure A3-2, on page A3-39.	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 The AXI4 AIP Properties

Property Name	Agent	Severity	Spec Reference	Property Description
ast_snps_axi4_arlen_stable	MASTER	ERROR	[ARM IHI 0022E] section A3.2.1, and figure A3-2, on page A3-39.	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_axi4_arsize_stable	MASTER	ERROR	[ARM IHI 0022E] section A3.2.1, and figure A3-2, on page A3-39.	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_axi4_arlock_stable	MASTER	ERROR	[ARM IHI 0022E] section A3.2.1, and figure A3-2, on page A3-39.	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_axi4_arcache_stable	MASTER	ERROR	[ARM IHI 0022E] section A3.2.1, and figure A3-2, on page A3-39.	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_axi4_arqos_stable	MASTER	ERROR	[ARM IHI 0022E] section A3.2.1, and figure A3-2, on page A3-39.	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_axi4_arregion_stable	MASTER	ERROR	[ARM IHI 0022E] section A3.2.1, and figure A3-2, on page A3-39.	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.
ast_snps_axi4_arprot_stable	MASTER	ERROR	[ARM IHI 0022E] section A3.2.1, and figure A3-2, on page A3-39.	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_axi4_arvalid_stable	MASTER	ERROR	[ARM IHI 0022E] section A3.2.2 on page A3-41.	When asserted, ARVALID must remain asserted until the rising clock edge after the master asserts ARREADY.

Table 4-3 The AXI4 AIP Properties

Property Name	Agent	Severity	Spec Reference	Property Description
ast_snps_axi4_awaddr_x	MASTER	ERROR	[ARM IHI 0022E] section A3.2.1 on page A3-39.	When AWVALID is high, a value of X/Z on AWADDR is not permitted.
ast_snps_axi4_awburst_x	MASTER	ERROR	[ARM IHI 0022E] section A3.2.1 on page A3-39.	When AWVALID is high, a value of X/Z on AWBURST is not permitted.
ast_snps_axi4_awid_x	MASTER	ERROR	[ARM IHI 0022E] section A3.2.1 on page A3-39.	When AWVALID is high, a value of X/Z on AWID is not permitted.
ast_snps_axi4_awlen_x	MASTER	ERROR	[ARM IHI 0022E] section A3.2.1 on page A3-39.	When AWVALID is high, a value of X/Z on AWLEN is not permitted.
ast_snps_axi4_awsize_x	MASTER	ERROR	[ARM IHI 0022E] section A3.2.1 on page A3-39.	When AWVALID is high, a value of X/Z on AWSIZE is not permitted.
ast_snps_axi4_awlock_x	MASTER	ERROR	[ARM IHI 0022E] section A3.2.1 on page A3-39.	When AWVALID is high, a value of X/Z on AWLOCK is not permitted.
ast_snps_axi4_awcache_x	MASTER	ERROR	[ARM IHI 0022E] section A3.2.1 on page A3-39.	When AWVALID is high, a value of X/Z on AWCACHE is not permitted.
ast_snps_axi4_awqos_x	MASTER	ERROR	[ARM IHI 0022E] section A3.2.1 on page A3-39.	When AWVALID is high, a value of X/Z on AWQOS is not permitted.
ast_snps_axi4_awregion_x	MASTER	ERROR	[ARM IHI 0022E] section A3.2.1 on page A3-39.	When AWVALID is high, a value of X/Z on AWREGION is not permitted.
ast_snps_axi4_awprot_x	MASTER	ERROR	[ARM IHI 0022E] section A3.2.1 on page A3-39.	When AWVALID is high, a value of X/Z on AWPROT is not permitted.
ast_snps_axi4_awvalid_x	MASTER	ERROR	[ARM IHI 0022E] section A3.2.1 on page A3-39.	When not in reset, a value of X/Z on AWVALID is not permitted.
ast_snps_axi4_wdata_x	MASTER	ERROR	[ARM IHI 0022E] section A3.2.1 on page A3-39.	When WVALID is high, a value of X/Z on WDATA is not permitted.
ast_snps_axi4_wlast_x	MASTER	ERROR	[ARM IHI 0022E] section A3.2.1 on page A3-39.	When WVALID is high, a value of X/Z on WLAST is not permitted.
ast_snps_axi4_wstrb_x	MASTER	ERROR	[ARM IHI 0022E] section A3.2.1 on page A3-39.	When WVALID is high, a value of X/Z on WSTRB is not permitted.
ast_snps_axi4_wvalid_x	MASTER	ERROR	[ARM IHI 0022E] section A3.2.1 on page A3-39.	When not in reset, a value of X/Z on WVALID is not permitted.
ast_snps_axi4_bready_x	MASTER	ERROR	[ARM IHI 0022E] section A3.2.1 on page A3-39.	When not in reset, a value of X/Z on BREADY is not permitted.
ast_snps_axi4_araddr_x	MASTER	ERROR	[ARM IHI 0022E] section A3.2.1 on page A3-39.	When ARVALID is high, a value of X/Z on ARADDR is not permitted.
ast_snps_axi4_arburst_x	MASTER	ERROR	[ARM IHI 0022E] section A3.2.1 on page A3-39.	When ARVALID is high, a value of X/Z on ARBURST is not permitted.

Table 4-3 The AXI4 AIP Properties

Property Name	Agent	Severity	Spec Reference	Property Description
ast_snps_axi4_arid_x	MASTER	ERROR	[ARM IHI 0022E] section A3.2.1 on page A3-39.	When ARVALID is high, a value of X/Z on ARID is not permitted.
ast_snps_axi4_arlen_x	MASTER	ERROR	[ARM IHI 0022E] section A3.2.1 on page A3-39.	When ARVALID is high, a value of X/Z on ARLEN is not permitted.
ast_snps_axi4_arsize_x	MASTER	ERROR	[ARM IHI 0022E] section A3.2.1 on page A3-39.	When ARVALID is high, a value of X/Z on ARSIZE is not permitted.
ast_snps_axi4_arlock_x	MASTER	ERROR	[ARM IHI 0022E] section A3.2.1 on page A3-39.	When ARVALID is high, a value of X/Z on ARLOCK is not permitted.
ast_snps_axi4_arcache_x	MASTER	ERROR	[ARM IHI 0022E] section A3.2.1 on page A3-39.	When ARVALID is high, a value of X/Z on ARCACHE is not permitted.
ast_snps_axi4_arqos_x	MASTER	ERROR	[ARM IHI 0022E] section A3.2.1 on page A3-39.	When ARVALID is high, a value of X/Z on ARQOS is not permitted.
ast_snps_axi4_arregion_x	MASTER	ERROR	[ARM IHI 0022E] section A3.2.1 on page A3-39.	When ARVALID is high, a value of X/Z on ARREGION is not permitted.
ast_snps_axi4_arprot_x	MASTER	ERROR	[ARM IHI 0022E] section A3.2.1 on page A3-39.	When ARVALID is high, a value of X/Z on ARPROT is not permitted.
ast_snps_axi4_arvalid_x	MASTER	ERROR	[ARM IHI 0022E] section A3.2.1 on page A3-39.	When not in reset, a value of X/Z on ARVALID is not permitted.
ast_snps_axi4_rready_x	MASTER	ERROR	[ARM IHI 0022E] section A3.2.1 on page A3-39.	When not in reset, a value of X/Z on RREADY is not permitted.
ast_snps_axi4_wlast_beat	MASTER	ERROR	[ARM IHI 0022E] section A3.2.2 on page A3-41.	The master must assert the WLAST signal while it is driving the final write transfer in the burst.
ast_snps_axi4_wstrb_align	MASTER	ERROR	[ARM IHI 0022E] section A3.4.3 on page A3-52.	A master must ensure that the write strobes are HIGH only for byte lanes that contains valid data.
ast_snps_axi4_wstrb_fixed	MASTER	ERROR	[ARM IHI 0022E] section A3.4.1 on page A3-47.	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 differ for each beat in the burst.
ast_snps_axi4_bready_max_waits	MASTER	WARNING	Recommendation to avoid deadlock.	BREADY should be asserted within B_MAX_WAITS cycles of BVALID being asserted.
ast_snps_axi4_bready_eventually	MASTER	WARNING	Recommendation to avoid deadlock.	BREADY should be asserted eventually when BVALID being asserted.

Table 4-3 The AXI4 AIP Properties

Property Name	Agent	Severity	Spec Reference	Property Description
ast_snps_axi4_rready_max_waits	MASTER	WARNING	Recommendation to avoid deadlock.	RREADY should be asserted within R_MAX_WAITS cycles of RVALID being asserted.
ast_snps_axi4_rready_eventually	MASTER	WARNING	Recommendation to avoid deadlock.	RREADY should be asserted eventually when RVALID being asserted.
ast_snps_axi4_awvalid_maxouts	MASTER	WARNING	Maximum number of outstanding transactions.	Master issued Write Requests for more than WR_MAX_BURSTS.
ast_snps_axi4_wvalid_maxouts	MASTER	WARNING	Maximum number of outstanding transactions.	Master issued Write Data for more than WR_MAX_BURSTS.
ast_snps_axi4_arvalid_maxouts	MASTER	WARNING	Maximum number of outstanding transactions.	Master issued Read Requests for more than RD_MAX_BURSTS.
ast_snps_axi4_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_axi4_excl_awcache	MASTER	ERROR	[ARM IHI 0022E] section A7.2.4 on page A7-97.	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_axi4_excl_arcache	MASTER	ERROR	[ARM IHI 0022E] section A7.2.4 on page A7-97.	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_axi4_excl_addr_align	MASTER	ERROR	[ARM IHI 0022E] section A7.2.4 on page A7-97.	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_axi4_excl_length	MASTER	ERROR	[ARM IHI 0022E] section A7.2.4 on page A7-97.	In AXI4, the burst length for an exclusive access must not exceed 16 transfers.
ast_snps_axi4_excl_transfer_size	MASTER	ERROR	[ARM IHI 0022E] section A7.2.4 on page A7-97.	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.

Table 4-3 **The AXI4 AIP Properties**

Property Name	Agent	Severity	Spec Reference	Property Description
ast_snps_axi4_excl_max_bytes	MASTER	ERROR	[ARM IHI 0022E] section A7.2.4 on page A7-97.	The maximum number of bytes that can be transferred in an exclusive burst is 128.
ast_snps_axi4_excl_max_depth	MASTER	ERROR	EXCL_DEPTH overflow	Outstanding Exclusive Reads issued more than EXCL_DEPTH.
ast_snps_axi4_excl_no_conc_rw	MASTER	ERROR	[ARM IHI 0022E] section A7.2.2	It is Recommended that exclusive reads and writes with the same ID not be issued at the same time
ast_snps_axi4_excl_addr_match	MASTER	ERROR	[ARM IHI 0022E] section A7.2.4 on page A7-97.	The address for the exclusive read and the exclusive write must be identical.
ast_snps_axi4_excl_size_match	MASTER	ERROR	[ARM IHI 0022E] section A7.2.4 on page A7-97.	The size for the exclusive read and exclusive write transactions must be identical.
ast_snps_axi4_excl_length_match	MASTER	ERROR	[ARM IHI 0022E] section A7.2.4 on page A7-97.	The length for the exclusive read and exclusive write transactions must be identical.
ast_snps_axi4_excl_burst_match	MASTER	ERROR	[ARM IHI 0022E] section A7.2.4 on page A7-97.	The BURST field for the exclusive read and exclusive write transactions must be identical.
ast_snps_axi4_excl_cache_match	MASTER	ERROR	[ARM IHI 0022E] section A7.2.4 on page A7-97.	The CACHE field for the exclusive read and exclusive write transactions must be identical.
ast_snps_axi4_excl_region_match	MASTER	ERROR	[ARM IHI 0022E] section A7.2.4 on page A7-97.	The REGION field for the exclusive read and exclusive write transactions must be identical.
ast_snps_axi4_excl_prot_match	MASTER	ERROR	[ARM IHI 0022E] section A7.2.4 on page A7-97.	The PROT field for the exclusive read and exclusive write transactions must be identical.
ast_snps_axi4_excl_user_match	MASTER	ERROR	[ARM IHI 0022E] section A7.2.4 on page A7-97.	The USER field for the exclusive read and exclusive write transactions must be identical.
ast_snps_axi4_excl_read_done	MASTER	ERROR	[ARM IHI 0022E] section A7.2.2 on page A7-96.	A master must not start the write part of an exclusive access sequence until the read part is complete.

Table 4-3 The AXI4 AIP Properties

Property Name	Agent	Severity	Spec Reference	Property Description
ast_snps_axi4_awuser_stable	MASTER	ERROR	[ARM IHI 0022E] section A3.2.1, and figure A3-2, on page A3-39.	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_axi4_wuser_stable	MASTER	ERROR	[ARM IHI 0022E] section A3.2.1, and figure A3-2, on page A3-39.	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_axi4_aruser_stable	MASTER	ERROR	[ARM IHI 0022E] section A3.2.1, and figure A3-2, on page A3-39.	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.
ast_snps_axi4_awuser_x	MASTER	ERROR	[ARM IHI 0022E] section A3.2.1 on page A3-39.	When AWVALID is high, a value of X/Z on AWUSER is not permitted.
ast_snps_axi4_wuser_x	MASTER	ERROR	[ARM IHI 0022E] section A3.2.1 on page A3-39.	When WVALID is high, a value of X/Z on WUSER is not permitted.
ast_snps_axi4_aruser_x	MASTER	ERROR	[ARM IHI 0022E] section A3.2.1 on page A3-39.	When ARVALID is high, a value of X/Z on ARUSER is not permitted.
ast_snps_axi4_awready_x	SLAVE	ERROR	[ARM IHI 0022E] section A3.2.1 on page A3-39.	When not in reset, a value of X/Z on AWREADY is not permitted.
ast_snps_axi4_wready_x	SLAVE	ERROR	[ARM IHI 0022E] section A3.2.1 on page A3-39.	When not in reset, a value of X/Z on WREADY is not permitted.
ast_snps_axi4_bresp_after_waddr	SLAVE	ERROR	[ARM IHI 0022E] section A3.3.1 on page A3-44.	The slave must wait for AWVALID, AWREADY, WVALID, and WREADY to be asserted before asserting BVALID.
ast_snps_axi4_bresp_after_wdata	SLAVE	ERROR	[ARM IHI 0022E] section A3.3.1 on page A3-44.	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.

Table 4-3 The AXI4 AIP Properties

Property Name	Agent	Severity	Spec Reference	Property Description
ast_snps_axi4_no_extra_bid	SLAVE	ERROR	[ARM IHI 0022E] section A3.2.2 on page A3-41 and A5.1 on page A5-78	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_axi4_bvalid_reset	SLAVE	ERROR	[ARM IHI 0022E] section A3.1.2 on page A3-38.	During reset, a slave interface must drive BVALID LOW.
ast_snps_axi4_bid_stable	SLAVE	ERROR	[ARM IHI 0022E] section A3.2.1, and figure A3-2, on page A3-39.	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_axi4_bresp_stable	SLAVE	ERROR	[ARM IHI 0022E] section A3.2.1, and figure A3-2, on page A3-39.	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_axi4_bvalid_stable	SLAVE	ERROR	[ARM IHI 0022E] section A3.2.2 on page A3-41.	When asserted, BVALID must remain asserted until the rising clock edge after the master asserts BREADY.
ast_snps_axi4_bresp_as_decerr	SLAVE	ERROR	[ARM IHI 0022E] section A3.4.4, and Table A3-4, on page A3-58.	If the interconnect cannot successfully decode a slave access, it must return the DECERR response.
ast_snps_axi4_bresp_no_decerr	SLAVE	WARNING	Design specific.	This design expects not to respond with DECERR.
ast_snps_axi4_bresp_no_slverr	SLAVE	WARNING	Design specific.	This design expects not to respond with SLVERR.
ast_snps_axi4_bid_x	SLAVE	ERROR	[ARM IHI 0022E] section A3.2.1 on page A3-39.	When BVALID is High, a value of X/Z on BID is not permitted.
ast_snps_axi4_bresp_x	SLAVE	ERROR	[ARM IHI 0022E] section A3.2.1 on page A3-39.	When BVALID is High, a value of X/Z on BRESP is not permitted.
ast_snps_axi4_bvalid_x	SLAVE	ERROR	[ARM IHI 0022E] section A3.2.1 on page A3-39.	When not in reset, a value of X/Z on BVALID is not permitted.
ast_snps_axi4_arready_x	SLAVE	ERROR	[ARM IHI 0022E] section A3.2.1 on page A3-39.	When not in reset, a value of X/Z on ARREADY is not permitted.

Table 4-3 The AXI4 AIP Properties

Property Name	Agent	Severity	Spec Reference	Property Description
ast_snps_axi4_no_extra_rid	SLAVE	ERROR	[ARM IHI 0022E] section A3.2.2 on page A3-41 and A5.1 on page A5-78	The slave can assert the RVALID signal only when it drives a valid write response. Slaves are required to reflect on the appropriate RID response an AXI ID received from a master.
ast_snps_axi4_rvalid_reset	SLAVE	ERROR	[ARM IHI 0022E] section A3.1.2 on page A3-38.	During reset, a slave interface must drive RVALID LOW.
ast_snps_axi4_rdata_stable	SLAVE	ERROR	[ARM IHI 0022E] section A3.2.1, and figure A3-2, on page A3-39.	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_axi4_rdata_stable_allbits	SLAVE	ERROR	[ARM IHI 0022E] section A3.2.1, and figure A3-2, on page A3-39.	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_axi4_rid_stable	SLAVE	ERROR	[ARM IHI 0022E] section A3.2.1, and figure A3-2, on page A3-39.	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_axi4_rlast_stable	SLAVE	ERROR	[ARM IHI 0022E] section A3.2.1, and figure A3-2, on page A3-39.	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.
ast_snps_axi4_rresp_stable	SLAVE	ERROR	[ARM IHI 0022E] section A3.2.1, and figure A3-2, on page A3-39.	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_axi4_rvalid_stable	SLAVE	ERROR	[ARM IHI 0022E] section A3.2.2 on page A3-41.	When asserted, RVALID must remain asserted until the rising clock edge after the master asserts RREADY.
ast_snps_axi4_rresp_as_decerr	SLAVE	ERROR	[ARM IHI 0022E] section A3.4.4, and Table A3-4, on page A3-58.	If the interconnect cannot successfully decode a slave access, it must return the DECERR response.
ast_snps_axi4_rresp_no_decerr	SLAVE	WARNING	Design specific.	This design expects not to respond with DECERR.

Table 4-3 The AXI4 AIP Properties

Property Name	Agent	Severity	Spec Reference	Property Description
ast_snps_axi4_rresp_no_slverr	SLAVE	WARNING	Design specific.	This design expects not to respond with SLVERR.
ast_snps_axi4_rdata_x	SLAVE	ERROR	[ARM IHI 0022E] section A3.2.1 on page A3-39.	When RVALID is High, a value of X/Z on RDATA is not permitted.
ast_snps_axi4_rid_x	SLAVE	ERROR	[ARM IHI 0022E] section A3.2.1 on page A3-39.	When RVALID is High, a value of X/Z on RID is not permitted.
ast_snps_axi4_rlast_x	SLAVE	ERROR	[ARM IHI 0022E] section A3.2.1 on page A3-39.	When RVALID is High, a value of X/Z on RLAST is not permitted.
ast_snps_axi4_rresp_x	SLAVE	ERROR	[ARM IHI 0022E] section A3.2.1 on page A3-39.	When RVALID is High, a value of X/Z on RRESP is not permitted.
ast_snps_axi4_rvalid_x	SLAVE	ERROR	[ARM IHI 0022E] section A3.2.1 on page A3-39.	When not in reset, a value of X/Z on RVALID is not permitted.
ast_snps_axi4_rlast_beat	SLAVE	ERROR	[ARM IHI 0022E] section A3.2.2 on page A3-41.	The slave must assert the RLAST signal while it is driving the final write transfer in the burst.
ast_snps_axi4_bresp_exokay_allow	SLAVE	ERROR	[ARM IHI 0022E] section A7.2.3 on page A7-97.	An EXOKAY write response can only be given to an exclusive write access.
ast_snps_axi4_bresp_exokay_not_allow	SLAVE	ERROR	[ARM IHI 0022E] section A7.2.3 on page A7-97.	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_axi4_rresp_exokay_allow	SLAVE	ERROR	[ARM IHI 0022E] section A7.2.3 on page A7-97.	An EXOKAY read response can only be given to an exclusive read access.
ast_snps_axi4_rresp_exokay_not_allow	SLAVE	ERROR	[ARM IHI 0022E] section A7.2.3 on page A7-97.	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_axi4_awready_max_waits	SLAVE	WARNING	Recommendation to avoid deadlock.	AWREADY should be asserted within AW_MAX_WAITS cycles of AWVALID being asserted.
ast_snps_axi4_awready_eventually	SLAVE	WARNING	Recommendation to avoid deadlock.	AWREADY should be asserted eventually when AWVALID being asserted.
ast_snps_axi4_wready_max_waits	SLAVE	WARNING	Recommendation to avoid deadlock.	WREADY should be asserted within W_MAX_WAITS cycles of WVALID being asserted.

Table 4-3 The AXI4 AIP Properties

Property Name	Agent	Severity	Spec Reference	Property Description
ast_snps_axi4_wready_ eventually	SLAVE	WARNING	Recommendation to avoid deadlock.	WREADY should be asserted eventually when WVALID being asserted.
ast_snps_axi4_arready_ max_waits	SLAVE	WARNING	Recommendation to avoid deadlock.	ARREADY should be asserted within AR_MAX_WAITS cycles of ARVALID being asserted.
ast_snps_axi4_arready_ eventually	SLAVE	WARNING	Recommendation to avoid deadlock.	ARREADY should be asserted eventually when ARVALID being asserted.
ast_snps_axi4_buser_st able	SLAVE	ERROR	[ARM IHI 0022E] section A3.2.1, and figure A3-2, on page A3-39.	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_axi4_ruser_sta ble	SLAVE	ERROR	[ARM IHI 0022E] section A3.2.1, and figure A3-2, on page A3-39.	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_axi4_buser_x	SLAVE	ERROR	[ARM IHI 0022E] section A3.2.1 on page A3-39.	When BVALID is high, a value of X/Z on BUSER is not permitted.
ast_snps_axi4_ruser_x	SLAVE	ERROR	[ARM IHI 0022E] section A3.2.1 on page A3-39.	When RVALID is high, a value of X/Z on RUSER is not permitted.
ast_snps_axi4_awready _maxouts	SLAVE	WARNING	Maximum number of outstanding transactions.	Slave should not accept Write Requests for more than WR_MAX_BURSTS.
ast_snps_axi4_wready_ maxouts	SLAVE	WARNING	Maximum number of outstanding transactions.	Slave should not accept Write Data for more than WR_MAX_BURSTS.
ast_snps_axi4_arready_ maxouts	SLAVE	WARNING	Maximum number of outstanding transactions.	Slave should not accept Read Requests for more than RD_MAX_BURSTS.
ast_snps_axi4_aw_max bursts	MASTER	ERROR	Maximum Burst Length.	Master cannot issue AWLEN greater than the configured maximum burst length.
ast_snps_axi4_ar_maxb ursts	MASTER	ERROR	Maximum Burst Length.	Master cannot issue ARLEN greater than the configured maximum burst length.
ast_snps_axi4_param_w data_width		ERROR	Parameter Setting	"WDATA_WIDTH should be either one of 32, 64, 128, 256, 512 or 1024."

Table 4-3 The AXI4 AIP Properties

Property Name	Agent	Severity	Spec Reference	Property Description
ast_snps_axi4_param_rdata_width		ERROR	Parameter Setting	"RDATA_WIDTH should be either one of 32, 64, 128, 256, 512 or 1024."
ast_snps_axi4_param_maxwbursts		ERROR	Parameter Setting	WR_MAX_BURSTS should be greater than or equal to 1.
ast_snps_axi4_param_maxrbursts		ERROR	Parameter Setting	RD_MAX_BURSTS should be greater than or equal to 1.
ast_snps_axi4_param_maxburstlength		ERROR	Parameter Setting	MAXBURSTLENGTH should be within 1 and 256.
ast_snps_axi4_csysreq_fall	MASTER	ERROR	[ARM IHI 0022E] A9.2.2 Figure A9-1 on page A9-107.	When CSYSREQ transitions from high to low, CSYSACK must be high.
ast_snps_axi4_csysreq_rise	MASTER	ERROR	[ARM IHI 0022E] A9.2.2 Figure A9-1 on page A9-107.	When CSYSREQ transitions from low to high, CSYSACK must be low.
ast_snps_axi4_csysack_fall	MASTER	ERROR	[ARM IHI 0022E] A9.2.2 Figure A9-1 on page A9-107.	When CSYSACK transitions from high to low, CSYSREQ must be low.
ast_snps_axi4_csysack_rise	MASTER	ERROR	[ARM IHI 0022E] A9.2.2 Figure A9-1 on page A9-107.	When CSYSACK transitions from low to high, CSYSREQ must be high.
ast_snps_axi4_cactive_x	MASTER	ERROR	[ARM IHI 0022E] A9.2.2 Figure A9-1 on page A9-107.	When not in reset, a value of X/Z on CACTIVE is not permitted.
ast_snps_axi4_csysreq_x	MASTER	ERROR	[ARM IHI 0022E] A9.2.2 Figure A9-1 on page A9-107.	When not in reset, a value of X/Z on CSYSREQ is not permitted.
ast_snps_axi4_csysack_x	MASTER	ERROR	[ARM IHI 0022E] A9.2.2 Figure A9-1 on page A9-107.	When not in reset, a value of X/Z on CSYSACK is not permitted.
ast_snps_axi4_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_axi4_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.

Table 4-3 The AXI4 AIP Properties

Property Name	Agent	Severity	Spec Reference	Property Description
ast_snps_axi4_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_axi4_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_axi4_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.
ast_snps_axi4_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_axi3_aw_w_eventually	MASTER	WARNING	Recommendation to avoid deadlock.	If AWVALID is asserted, corresponding WVALID should be eventually asserted.
ast_snps_axi3_w_aw_eventually	MASTER	WARNING	Recommendation to avoid deadlock.	If WVALID is asserted, corresponding AWVALID should be eventually asserted.
ast_snps_axi3_wlast_eventually	MASTER	WARNING	Recommendation to avoid deadlock.	If WVALID is asserted without WLAST, corresponding WLAST should be eventually asserted.
ast_snps_axi3_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_axi3_ar_r_eventually	SLAVE	WARNING	Recommendation to avoid deadlock.	If ARVALID is asserted, corresponding RVALID should be eventually asserted.
ast_snps_axi3_rlast_eventually	SLAVE	WARNING	Recommendation to avoid deadlock.	If RVALID is asserted without RLAST, corresponding RLAST should be eventually asserted.

4.4 The AXI4 AIP Cover Properties

Table 4-4 The AXI4 AIP Cover Properties

Cover Property Name	Property Description
cov_snps_axi4_awburst_fixed	Observed FIXED write burst
cov_snps_axi4_awburst_incr	Observed INCR write burst
cov_snps_axi4_awburst_wrap	Observed WRAP write burst
cov_snps_axi4_awcache_noncacheable_nonbufferable	Observed write burst with AWCACHE = Noncacheable and non-bufferable
cov_snps_axi4_awcache_bufferable_only	Observed write burst with AWCACHE = Bufferable only
cov_snps_axi4_awcache_cacheable_not_allocate	Observed write burst with AWCACHE = Cacheable, but do not allocate
cov_snps_axi4_awcache_cacheable_bufferable_not_allocate	Observed write burst with AWCACHE = Cacheable and bufferable, but do not allocate
cov_snps_axi4_awcache_cacheable_writethrough_allocate_readonly	Observed write burst with AWCACHE = Cacheable write-through, allocate on reads only
cov_snps_axi4_awcache_cacheable_writeback_allocate_readonly	Observed write burst with AWCACHE = Cacheable write-back, allocate on reads only
cov_snps_axi4_awcache_cacheable_writethrough_allocate_writeonly	Observed write burst with AWCACHE = Cacheable write-through, allocate on writes only
cov_snps_axi4_awcache_cacheable_writeback_allocate_writeonly	Observed write burst with AWCACHE = Cacheable write-back, allocate on writes only
cov_snps_axi4_awcache_cacheable_writethrough_allocate_read_write	Observed write burst with AWCACHE = Cacheable write-through, allocate on both reads and writes
cov_snps_axi4_awcache_cacheable_writeback_allocate_read_write	Observed write burst with AWCACHE = Cacheable write-back, allocate on both reads and writes
cov_snps_axi4_awsize_8_bits	Observed write burst with AWSIZE = 8 bits
cov_snps_axi4_awsize_16_bits	Observed write burst with AWSIZE = 16 bits
cov_snps_axi4_awsize_32_bits	Observed write burst with AWSIZE = 32 bits
cov_snps_axi4_awsize_64_bits	Observed write burst with AWSIZE = 64 bits
cov_snps_axi4_awsize_128_bits	Observed write burst with AWSIZE = 128 bits
cov_snps_axi4_awsize_256_bits	Observed write burst with AWSIZE = 256 bits
cov_snps_axi4_awsize_512_bits	Observed write burst with AWSIZE = 512 bits
cov_snps_axi4_awsize_1024_bits	Observed write burst with AWSIZE = 1024 bits

Table 4-4 The AXI4 AIP Cover Properties

Cover Property Name	Property Description
cov_snps_axi4_awlen_len	Observed write burst with AWLEN = N (N: 0 to MAXBURSTLENGTH-1)
cov_snps_axi4_awprot_normal_access	Observed write burst with AWPROT = normal access
cov_snps_axi4_awprot_privileged_access	Observed write burst with AWPROT = privileged access
cov_snps_axi4_awprot_secure_access	Observed write burst with AWPROT = secure access
cov_snps_axi4_awprot_nonsecure_access	Observed write burst with AWPROT = nonsecure access
cov_snps_axi4_awprot_data_access	Observed write burst with AWPROT = data access
cov_snps_axi4_awprot_instruction_access	Observed write burst with AWPROT = instruction access
cov_snps_axi4_awvalid_wait_awready	Observed AWVALID = High and AWREADY = Low
cov_snps_axi4_awready_wait_awvalid	Observed AWVALID = Low and AWREADY = High
cov_snps_axi4_awvalid_awready_both	Observed AWVALID = High and AWREADT = High
cov_snps_axi4_awvalid_awready_idle	Observed AWVALID = Low and AWREADT = Low
cov_snps_axi4_wvalid_wait_wready	Observed WVALID = High and WREADT = Low
cov_snps_axi4_wready_wait_wvalid	Observed WVALID = Low and WREADT = High
cov_snps_axi4_wvalid_wready_both	Observed WVALID = High and WREADT = High
cov_snps_axi4_wvalid_wready_idle	Observed WVALID = Low and WREADT = Low
cov_snps_axi4_bresp_okay	Observed write response with BRESP = OKAY
cov_snps_axi4_bvalid_wait_bready	Observed BVALID = High and BREADY = Low
cov_snps_axi4_bready_wait_bvalid	Observed BVALID = Low and BREADY = High
cov_snps_axi4_bvalid_bready_both	Observed BVALID = High and BREADY = High
cov_snps_axi4_bvalid_bready_idle	Observed BVALID = Low and BREADY = Low
cov_snps_axi4_bresp_slverr	Observed write response with BRESP = SLVERR
cov_snps_axi4_bresp_decerr	Observed write response with BRESP = DECERR
cov_snps_axi4_write_out_of_order	Observed out of order write response
cov_snps_axi4_write_addr_resp_outstands	Observed the maximum number of outstanding write transactions reach to WR_MAX_BURSTS (address channel)
cov_snps_axi4_write_data_resp_outstands	Observed the maximum number of outstanding write transactions reach to WR_MAX_BURSTS (data channel)
cov_snps_axi4_arburst_fixed	Observed FIXED read burst

Table 4-4 The AXI4 AIP Cover Properties

Cover Property Name	Property Description
cov_snps_axi4_arburst_incr	Observed INCR read burst
cov_snps_axi4_arburst_wrap	Observed WRAP read burst
cov_snps_axi4_arcache_noncacheable_nonbufferable	Observed read burst with ARCACHE = Noncacheable and nonbufferable
cov_snps_axi4_arcache_bufferable_only	Observed read burst with ARCACHE = Bufferable only
cov_snps_axi4_arcache_cacheable_not_allocate	Observed read burst with ARCACHE = Cacheable, but do not allocate
cov_snps_axi4_arcache_cacheable_bufferable_not_allocate	Observed read burst with ARCACHE = Cacheable and bufferable, but do not allocate
cov_snps_axi4_arcache_cacheable_writethrough_allocate_readonly	Observed read burst with ARCACHE = Cacheable write-through, allocate on reads only
cov_snps_axi4_arcache_cacheable_writeback_allocate_readonly	Observed read burst with ARCACHE = Cacheable write-back, allocate on reads only
cov_snps_axi4_arcache_cacheable_writethrough_allocate_writeonly	Observed read burst with ARCACHE = Cacheable write-through, allocate on writes only
cov_snps_axi4_arcache_cacheable_writeback_allocate_writeonly	Observed read burst with ARCACHE = Cacheable write-back, allocate on writes only
cov_snps_axi4_arcache_cacheable_writethrough_allocate_read_write	Observed read burst with ARCACHE = Cacheable write-through, allocate on both reads and writes
cov_snps_axi4_arcache_cacheable_writeback_allocate_read_write	Observed read burst with ARCACHE = Cacheable write-back, allocate on both reads and writes
cov_snps_axi4_arsize_8_bits	Observed read burst with ARSIZE = 8 bits
cov_snps_axi4_arsize_16_bits	Observed read burst with ARSIZE = 16 bits
cov_snps_axi4_arsize_32_bits	Observed read burst with ARSIZE = 32 bits
cov_snps_axi4_arsize_64_bits	Observed read burst with ARSIZE = 64 bits
cov_snps_axi4_arsize_128_bits	Observed read burst with ARSIZE = 128 bits
cov_snps_axi4_arsize_256_bits	Observed read burst with ARSIZE = 256 bits
cov_snps_axi4_arsize_512_bits	Observed read burst with ARSIZE = 512 bits
cov_snps_axi4_arsize_1024_bits	Observed read burst with ARSIZE = 1024 bits
cov_snps_axi4_arlen_len	Observed read burst with ARLEN = N (N: 0 to MAXBURSTLENGTH-1)
cov_snps_axi4_arprot_normal_access	Observed read burst with ARPROT = normal access
cov_snps_axi4_arprot_privileged_access	Observed read burst with ARPROT = privileged access

Table 4-4 The AXI4 AIP Cover Properties

Cover Property Name	Property Description
cov_snps_axi4_arprot_secure_access	Observed read burst with ARPROT = secure access
cov_snps_axi4_arprot_nonsecure_access	Observed read burst with ARPROT = nonsecure access
cov_snps_axi4_arprot_data_access	Observed read burst with ARPROT = data access
cov_snps_axi4_arprot_instruction_access	Observed read burst with ARPROT = instruction access
cov_snps_axi4_arvalid_wait_arready	Observed ARVALID = High and ARREADY = Low
cov_snps_axi4_arready_wait_arvalid	Observed ARVALID = Low and ARREADY = High
cov_snps_axi4_arvalid_arready_both	Observed ARVALID = High and ARREADT = High
cov_snps_axi4_arvalid_arready_idle	Observed ARVALID = Low and ARREADT = Low
cov_snps_axi4_rresp_okay	Observed read response with RRESP = OKAY
cov_snps_axi4_rvalid_wait_rready	Observed RVALID = High and RREADY = Low
cov_snps_axi4_rready_wait_rvalid	Observed RVALID = Low and RREADY = High
cov_snps_axi4_rvalid_rready_both	Observed RVALID = High and RREADY = High
cov_snps_axi4_rvalid_rready_idle	Observed RVALID = Low and RREADY = Low
cov_snps_axi4_rresp_slverr	Observed read response with RRESP = SLVERR
cov_snps_axi4_rresp_decerr	Observed read response with RRESP = DECERR
cov_snps_axi4_read_interleaved	Observed interleaved read bursts
cov_snps_axi4_read_out_of_order	Observed out of order read response
cov_snps_axi4_read_addr_data_outstands	Observed the maximum number of outstanding read transactions reach to RD_MAX_BURSTS
cov_snps_axi4_awid_bit_l	Observed write burst with AWID[n] = 0
cov_snps_axi4_awid_bit_h	Observed write burst with AWID[n] = 1
cov_snps_axi4_bid_bit_l	Observed write response with BID[n] = 0
cov_snps_axi4_bid_bit_h	Observed write response with BID[n] = 1
cov_snps_axi4_arid_bit_l	Observed read burst with ARID[n] = 0
cov_snps_axi4_arid_bit_h	Observed read burst with ARID[n] = 1
cov_snps_axi4_rid_bit_l	Observed read data with RID[n] = 0
cov_snps_axi4_rid_bit_h	Observed read data with RID[n] = 1
cov_snps_axi4_awid_value	Observed write burst with AWID = N (all possible IDs)
cov_snps_axi4_bid_value	Observed write response with BID = N (all possible IDs)

Table 4-4 The AXI4 AIP Cover Properties

Cover Property Name	Property Description
cov_snps_axi4_arid_value	Observed read burst with ARID = N (all possible IDs)
cov_snps_axi4_rid_value	Observed read data with RID = N (all possible IDs)
cov_snps_axi4_awlock_normal	Observed write burst with AWLOCK = Normal access
cov_snps_axi4_awlock_excl	Observed write burst with AWLOCK = Exclusive access
cov_snps_axi4_arlock_normal	Observed read burst with ARLOCK = Normal access
cov_snps_axi4_arlock_excl	Observed read burst with ARLOCK = Exclusive access
cov_snps_axi4_bresp_exokay	Observed write response with BRESP = EXOKAY
cov_snps_axi4_rresp_exokay	Observed read response with RRESP = EXOKAY
cov_snps_axi4_awqos_value	Observed write request with AWQOS = N
cov_snps_axi4_arqos_value	Observed read request with ARQOS = N
cov_snps_axi4_awregion_value	Observed write request with AWREGION = N
cov_snps_axi4_arregion_value	Observed read request with ARREGION = N

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 write strobe check properties:

  Set `CONFIG_WSTRB=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.

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 SLAVE, checkers mentioned as `Slave' in Master/Slave checkers column are declared as assume, and checkers mentioned as `Master' 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 AXI4 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 Default value is 1
2020.03-SP2-1 and above versions	CONFIG_MAXOUTS==0: same with previous version CONFIG_MAXOUTS==1: new behavior CONFIG_MAXOUTS==2: same with CONFIG_MAXOUTS==1 in previous version 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). 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. 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. 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). 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. Use asm_snps_axi_awvalid_maxouts, asm_snps_axi_wvalid_maxouts and asm_snps_axi_arvalid_maxouts.

Table 4-6 Behavior Per Value

	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. 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. 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. Use asm_snps_axi_awvalid_maxouts, asm_snps_axi_wvalid_maxouts and asm_snps_axi_arvalid_maxouts.

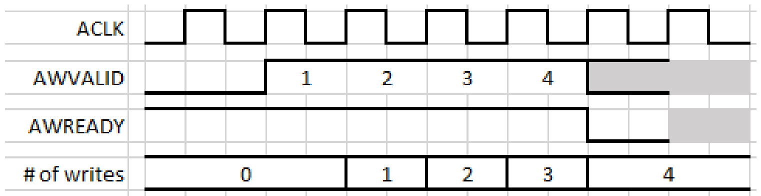
This change is applicable only for assume properties. This is not applicable for AMBA specification and depends on design implementation. Hence, assertions ast_snps_axi_awvalid_maxouts, ast_snps_axi_wvalid_maxouts, and ast_snps_axi_arvalid_maxouts do not check protocol violation and are categorized as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

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

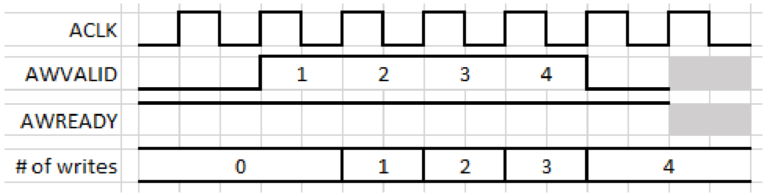
4.6.1 Background of this Change (when a potential bug can be missed in a slave DUT):

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:

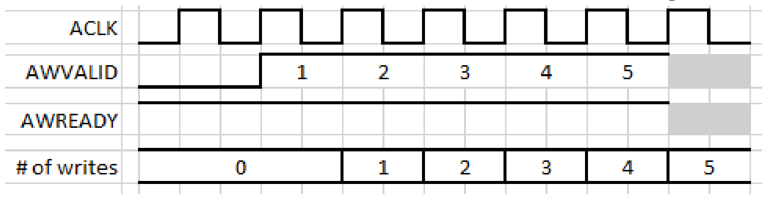


However, if MASTER AIP does not issue 5th write request, then it is not possible to verify if slave DUT stops AWREADY or not for the 5th request. As shown below, the number of outstanding transactions is 4 and slave DUT returns 5th AWEREADY.

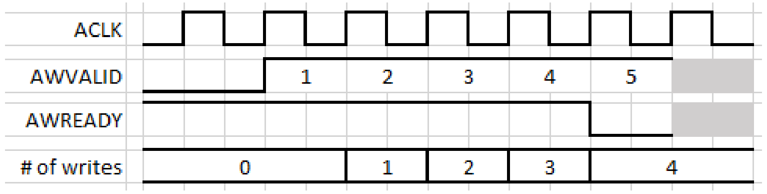


To verify if slave DUT stops to return AWREADY, you must set WR_MAX_BURSTS with 5 or bigger. However, user most likely sets WR_MAX_BURSTS with 4 in such cases. Therefore, it would be desirable to change AIP behavior that MASTER AIP issues write requests up to (WR_MAX_BURSTS+1). This is same with read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

5
The AXI4 AIP Use Cases

This chapter discusses about the AXI4 AIP in different environments used for validation.

5.1 The AXI4 AIP Examples

This section describes the setup of the AXI4 AIP where it is connected with RTL through a bind file. The bind file example is shown in Table 5-1 and Table 5-2, where both master and slave DUT signals are connected to AIP master and slave signals.

Note
If a signal corresponding to the AXI4 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 `2’h0`.

5.1.1 The AXI4 AIP With a Slave DUT

The following steps describe the setup of the AXI4 AIP with a slave DUT:

1. In this setup, the AXI4 AIP is connected with the AXI4 Slave DUT.
2. For connecting the ports of the AXI4 DUT with the AXI4 AIP, create a bind file to include the instance of the AXI4 AIP into the top level module of the DUT.
3. Instantiate the master AXI4 AIP and connect with the slave DUT signal.
4. Figure 5-1 shows a slave DUT connected with the master AXI4 AIP.

Figure 5-1   A Slave DUT With the Master AXI4 AIP

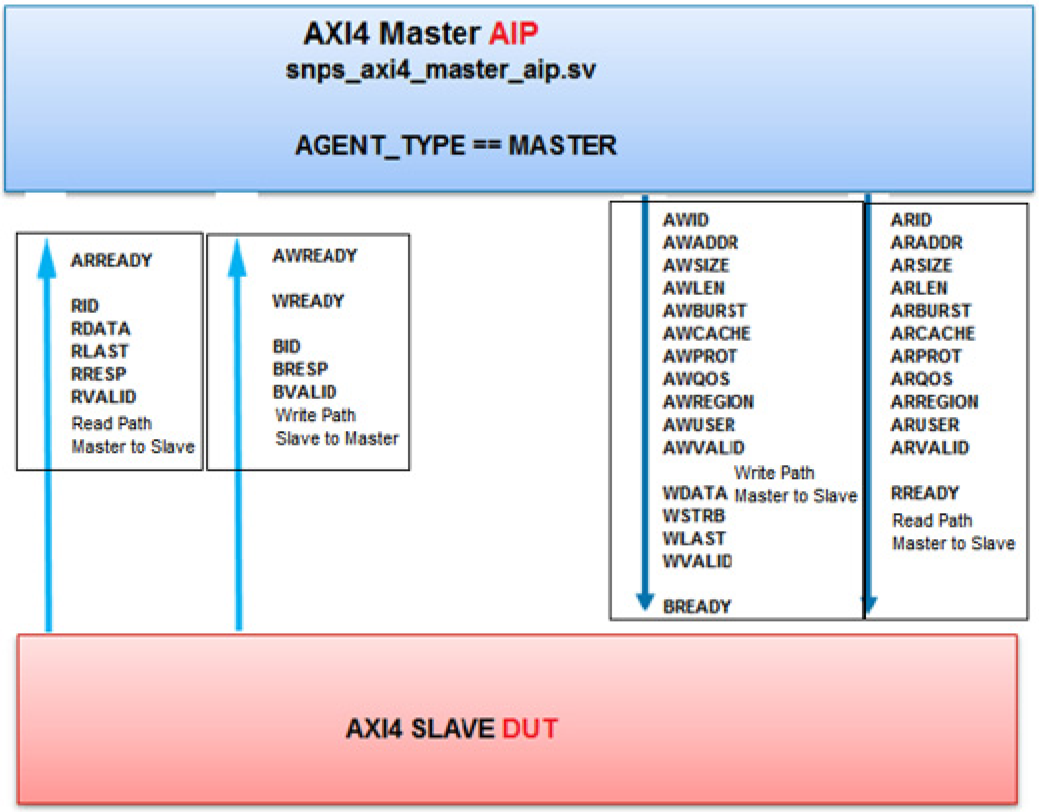


Table 5-1 The AXI4 Master AIP – Slave DUT Setup

```
module axi4_slave_dut (
//<channel signals>
csysreq,
csysack,
cactive,
busy_status,
aclk
aresetn,
awready,
awvalid,
awid,
awaddr,
awlen,
awsize,
awburst,
awlock,
awcache,
awprot,
awqos,
awregion,
awsideband,
wready,
wvalid,
wlast,
wdata,
wstrb,
wsideband,
bready,
bvalid,
bid,
bresp,
bsideband,
arready,
arvalid,
arid,
araddr,
arlen,
arsize,
arburst,
arlock,
arcache,
arprot,
arqos,
arregion,
arsideband,
rready,
rvalid,
rlast,
rid,
rdata,
rresp,
rsideband)
//Parameter Declaration
parameter A2X_BRESP_MODE = 1;
parameter A2X_BRESP_ORDER = 1;
parameter A2X_READ_ORDER = 1;
parameter A2X_READ_INTLEV = 1;
//<other parameter declarations>
//<logic part>
endmodule
```

```
bind slave_dut
snps_axi4_master_aip  #(
.AGENT_TYPE           (MASTER),
.ENABLE_ASSERT      (1),
.ENABLE_ASSUME    (1),
// < Other Parameter Connection>
.AWADDR_WIDTH                    (A2X_PP_AWIDTH),
.ARADDR_WIDTH                    (A2X_PP_AWIDTH),
.WUSER_WIDTH                     (A2X_INT_AWSBW),
.RUSER_WIDTH                     (A2X_INT_ARSBW),
u_axi4_mst_aip (
.aclk                        (clk ),
.aresetn               (resetn),
.awid                    (awid),
.awaddr                (awaddr),
.awlen                   (awlen),
.awsize                 (awsize),
.awburst               (awburst),
.awlock                 (awlock),
.awcache              (awcache),
.awprot                 (awprot),
.awuser                 (awuser),
.awqos                  (awqos),
.awregion              (awregion),
.awvalid                  (awvalid),
.awready               (awready),
.wlast                       (wlast ),
.wdata                    (wdata),
.wstrb                      (wstrb),
.wuser                    (wuser),
.wvalid                   (wvalid),
.wready                 (wready),
.bid                          (bid),
.bresp                    (bresp),
.buser                    (buser),
.bvalid                   (bvalid),
.bready                 (bready),
.arid                       (arid),
.araddr                 (araddr),
.arlen                     (arlen),
.arsize                   (arsize),
.arburst                (arburst),
.arlock                   (arlock),
.arcache               (arcache),
.arprot                   (arprot),
.arqos                    (arqos),
.arregion              (arregion),
.aruser                   (aruser),
.arvalid                  (arvalid),
.arready               (arready),
.rid                          (rid),
.rlast                      (rlast),
.rdata                    (rdata),
.rresp                    (rresp),
.ruser                    (ruser),
.rvalid                   (rvalid),
.rready                 (rread),
.cactive                 (1'b0),
.csysreq               (1'b0),
.csysack               (1'b0));
```

5.1.2 The AXI4 AIP With a Master DUT

The following steps describe the setup of the AXI4 AIP with a master DUT:

- In this setup, the AXI4 AIP is connected with the AXI4 Master DUT.
- For connecting the ports of the AXI4 DUT with the AXI4 AIP, create a bind file to include the instance of the AXI4 AIP into the top level module of the DUT.
- Instantiate the slave AXI4 AIP and connect with the master DUT signal.
- Figure 5-2 shows a Master DUT connected with the slave AXI4 AIP.

Figure 5-2 A Master DUT With the Slave AXI4 AIP

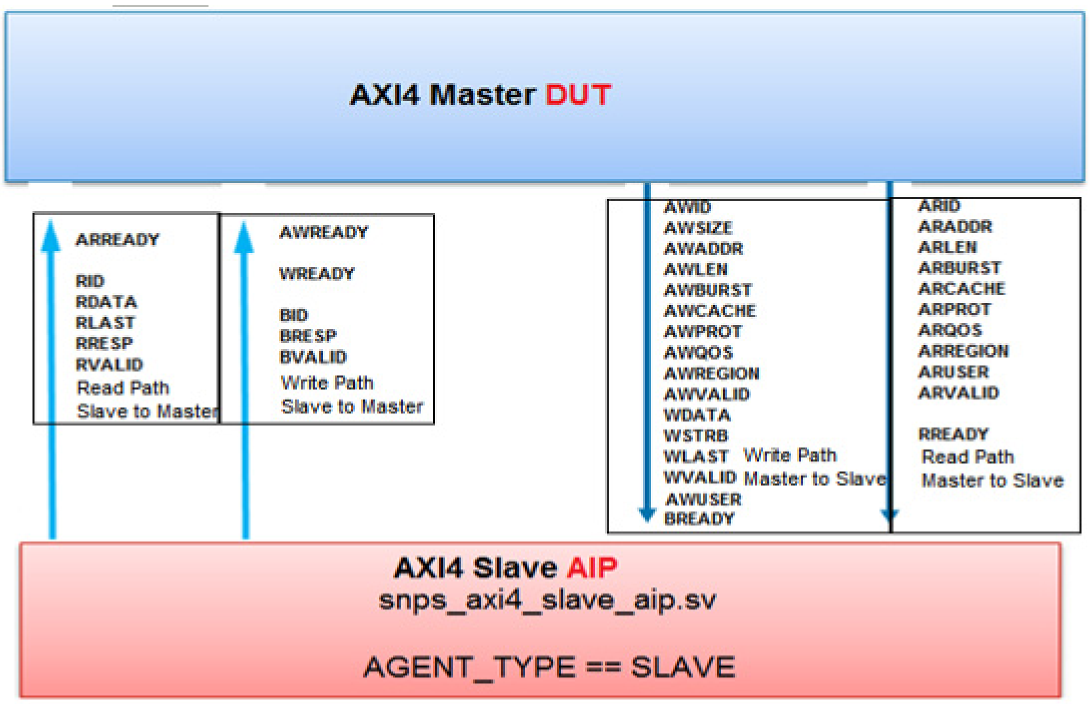


Table 5-2 The AXI4 Master DUT – Slave AIP Setup

```
module AXI4_master_dut (
//<channel signals>
                   csysreq,
                   csysack,
                   cactive,
                   busy_status,
                   aclk,
                   aresetn,
                   awready,
                   awvalid,
                   awid,
                   awaddr,
                   awlen,
                   awsize,
                   awburst,
                   awlock,
                   awcache,
                   awprot,
                   awqos,
                   awregion,
                   awsideband,
                    wready,
                   wvalid,
                   wlast,
                   wdata,
                   wstrb,
                   wsideband,
                   bready,
                   bvalid,
                   bid,
                   bresp,
                   bsideband,
                    arready,
                   arvalid,
                   arid,
                   araddr,
                   arlen,
                   arsize,
                   arburst,
                   arlock,
                   arcache,
                   arprot,
                   arqos,
                   arregion,
                   arsideband,
                   rready,
                   rvalid,
                   rlast,
                   rid,
                   rdata,
                   rresp,
                   rsideband)
//Parameter Declaration
  parameter A2X_BRESP_MODE = 1;
  parameter A2X_BRESP_ORDER = 1;
  parameter A2X_READ_ORDER = 1;
  parameter A2X_READ_INTLEV = 1;
 //<other parameter declarations>
//<logic part>
endmodule
```

```
 bind master_dut
      snps_AXI4_slave_aip  #(
.AGENT_TYPE           (SLAVE),
       .ENABLE_ASSERT     (1),
       .ENABLE_ASSUME   (1),
// < Other Parameter Connection>
       .AWADDR_WIDTH
(A2X_PP_AWIDTH),
        .ARADDR_WIDTH
(A2X_PP_AWIDTH),
        .WUSER_WIDTH
(A2X_INT_AWSBW),
        .RUSER_WIDTH
(A2X_INT_ARSBW),
 u_AXI4_slv_aip (
        .aclk                    (aclk),
        .aresetn              (aresetn),
        .awid                    (awid),
        .awaddr                (awaddr),
        .awlen                  (awlen),
        .awsize                (awsize),
        .awburst              (awburst),
        .awlock                (awlock),
        .awcache              (awcache),
        .awprot                (awprot),
        .awuser                (awuser),
        .awqos                  (awqos),
        .awregion            (awregion),
        .awvalid              (awvalid),
        .awready              (awready),
        .wlast                  (wlast),
        .wdata                  (wdata),
        .wstrb                  (wstrb),
        .wuser                  (wuser),
        .wvalid                (wvalid),
        .wready                (wready),
        .bid                      (bid),
        .bresp                  (bresp),
        .buser                  (buser),
        .bvalid                (bvalid),
        .bready                (bready),
        .arid                    (arid),
        .araddr                (araddr),
        .arlen                  (arlen),
        .arsize                (arsize),
        .arburst              (arburst),
        .arlock                (arlock),
        .arcache              (arcache),
        .arprot                (arprot),
        .aruser                (aruser),
        .arqos                  (arqos),
       .arregion             (arregion),
       .arvalid               (arvalid),
       .arready              (arready),
        .rid                      (rid),
        .rlast                  (rlast),
        .rdata                  (rdata),
        .rresp                  (rresp),
        .ruser                  (ruser),
        .rvalid                (rvalid),
        .rready                (rready),
        .cactive                (1'b0),
        .csysreq              (1'b0),
        .csysack              (1'b0));
```

5.2 Deadlock Properties

This section describes the details for deadlock properties.

5.2.1 Deadlock Configurations and Properties

AXI4 AIP has the following configuration parameters for deadlock related checks:

Table 5-3 Configuration Parameters for Deadlock Related Checks

Parameter Name	Default	Description
CONFIG_WAITS	1	1: Enable VALID-READY checks 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 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.

AXI4 AIP has the following properties for deadlock related checks:

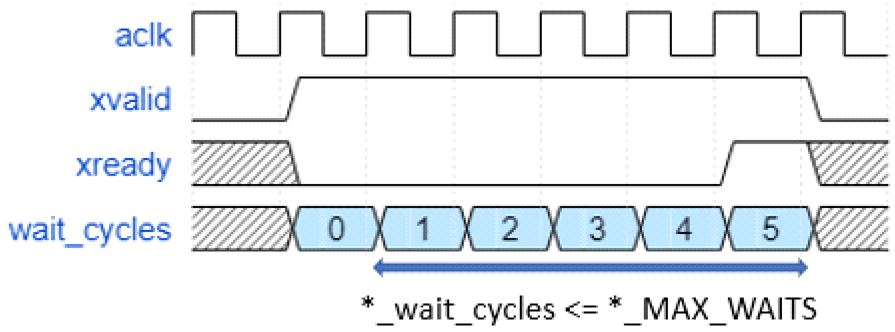
Table 5-4 **Properties for Deadlock Related Checks**

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_axi4_awready_max_waits	CONFIG_WAITS==1	AW_MAX_WAITS
WREADY should be asserted within W_MAX_WAITS cycles after WVALID is asserted.	ast_snps_axi4_wready_max_waits	CONFIG_WAITS==1	W_MAX_WAITS
BREADY should be asserted within B_MAX_WAITS cycles after BVALID is asserted.	ast_snps_axi4_bready_max_waits	CONFIG_WAITS==1	B_MAX_WAITS
ARREADY should be asserted within AR_MAX_WAITS cycles after ARVALID is asserted.	ast_snps_axi4_arready_max_waits	CONFIG_WAITS==1	AR_MAX_WAITS
RREADY should be asserted within R_MAX_WAITS cycles after RVALID is asserted.	ast_snps_axi4_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_axi4_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_axi4_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_axi4_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_axi4_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_axi4_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_axi4_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’.

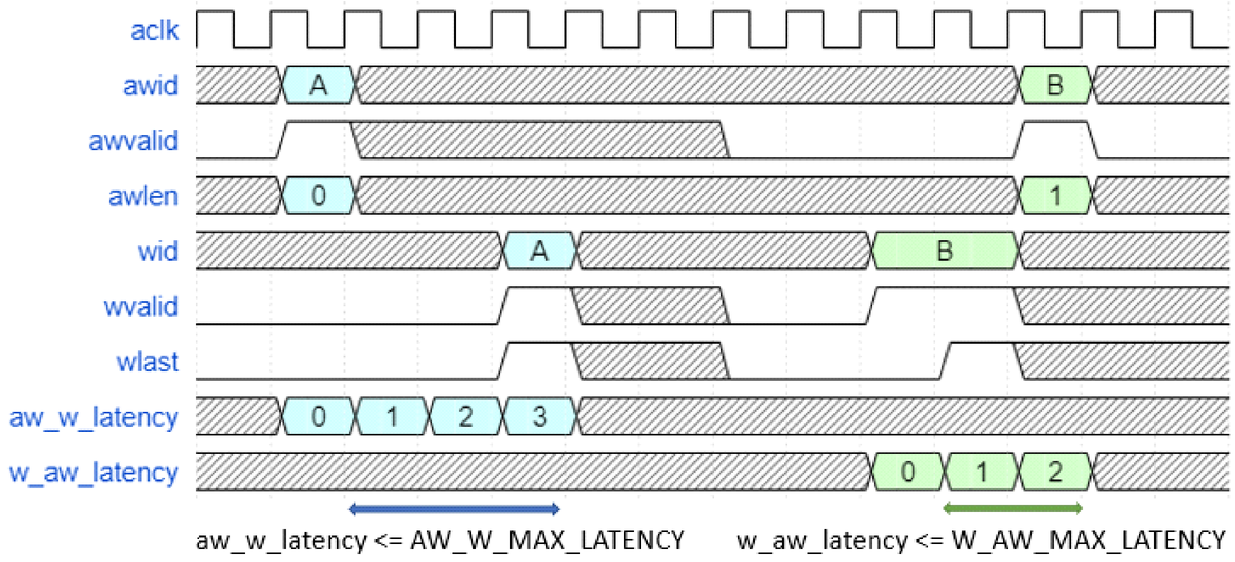
Figure 5-3 Example for the Relation Between xvalid and xready



The assertion ast_snps_axi4_awready_max_waits checks if ‘awready’ should be returned within AW_MAX_WAITS cycles once ‘awvalid’ is asserted. Similarly, ast_snps_axi4_wready_max_waits checks maximum wait cycles of ‘wready’, ast_snps_axi4_bready_max_waits checks maximum wait cycles of ‘bready’, ast_snps_axi4_arready_max_waits checks maximum wait cycles of ‘arready’ and ast_snps_axi4_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 5-4 Example for the Relation Between Write Address Channel and Write Data Channel



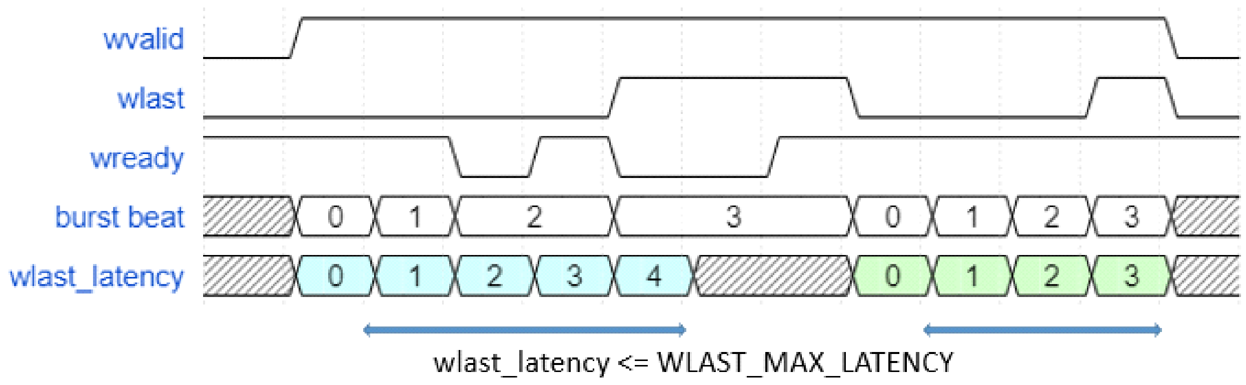
The assertion ast_snps_axi4_aw_w_max_latency checks if the latency from the start of write address channel to the start of write data channel should be within AW_W_MAX_LATENCY cycles when write address channel is issued before write data channel.

The assertion ast_snps_axi4_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'.

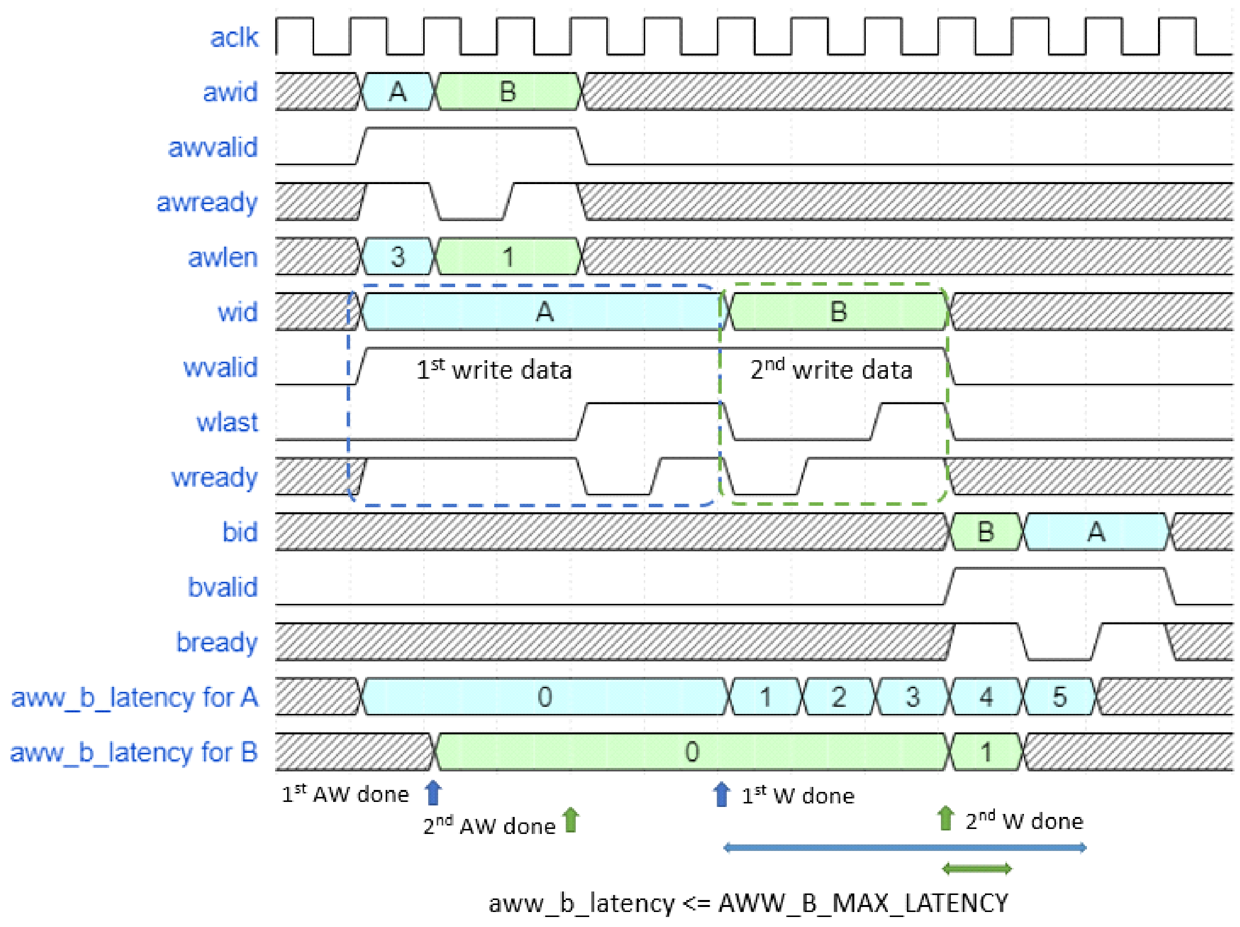
Figure 5-5 Example for the Relation Between wvalid and wlast



The assertion ast_snps_axi4_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:

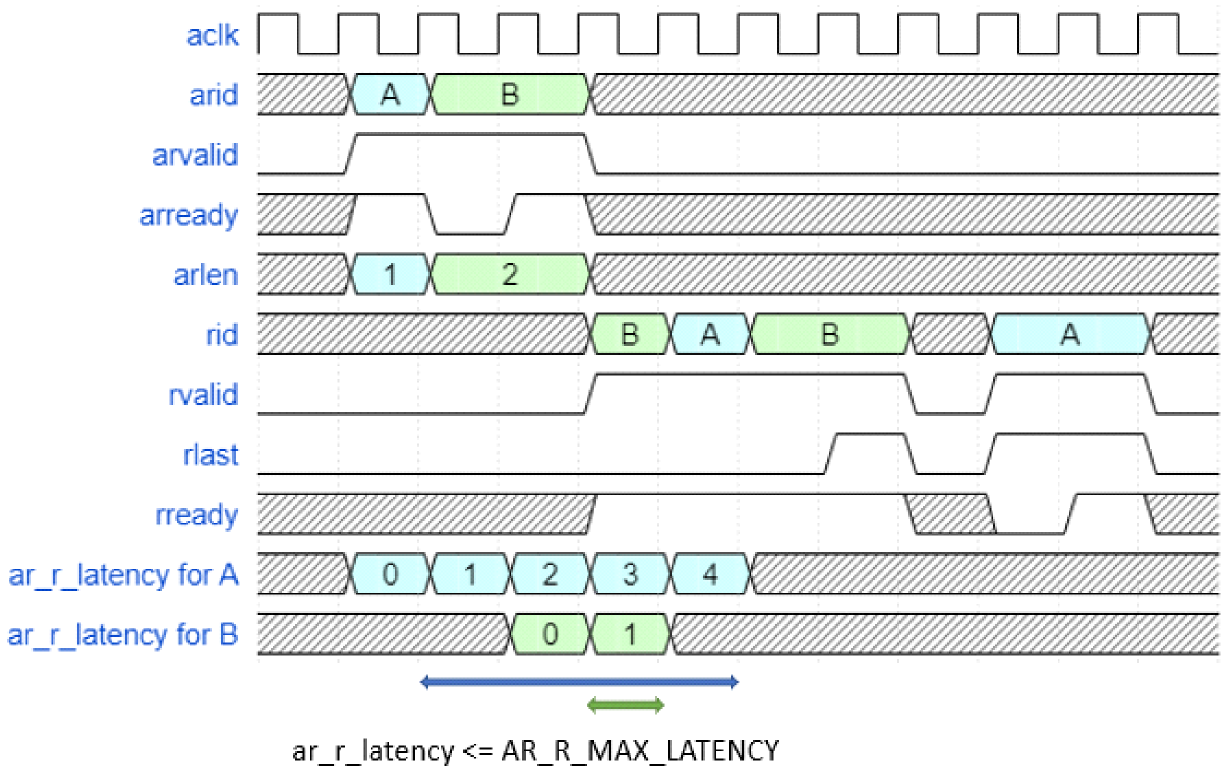
Figure 5-6 Example for the Relation Between the Completion of Write Address/Data Channels and Write Response



The assertion ast_snps_axi4_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:

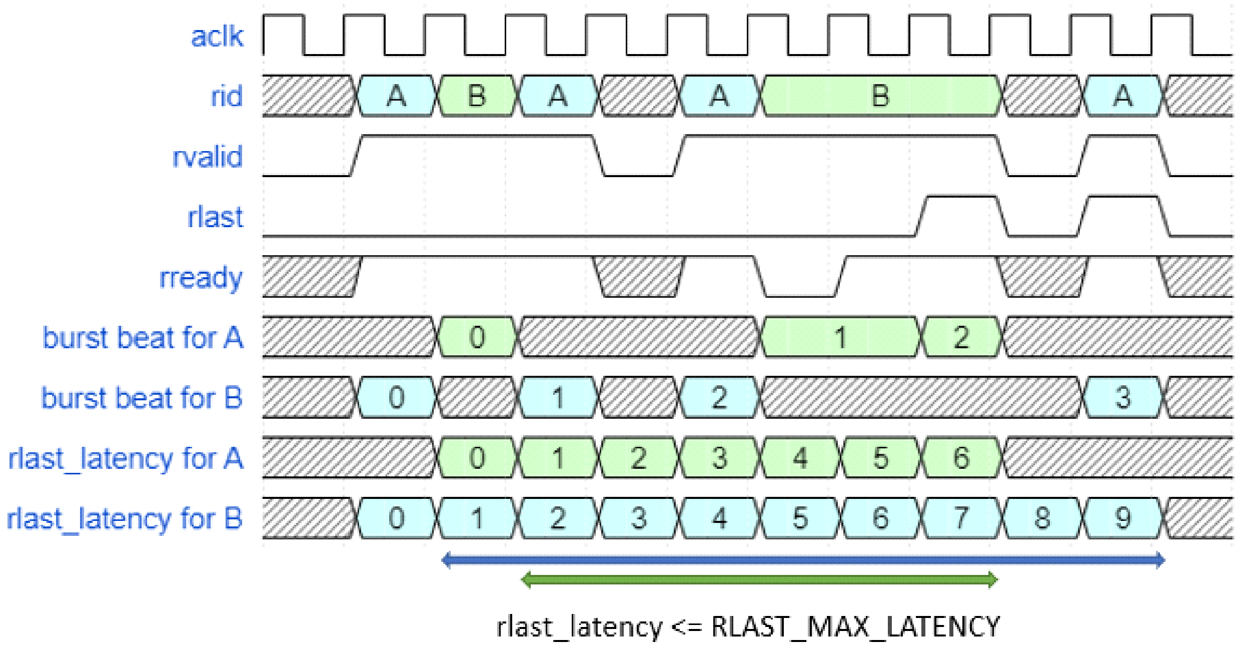
Figure 5-7 Example for the Relation Between the Completion of Read Address Channel and the Start of Read Response Channel



The assertion ast_snps_axi4_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 5-8 Example for the Relation Between the Start of Read Response Channel and the Last Beat of Read Response Channel



The assertion ast_snps_axi4_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.

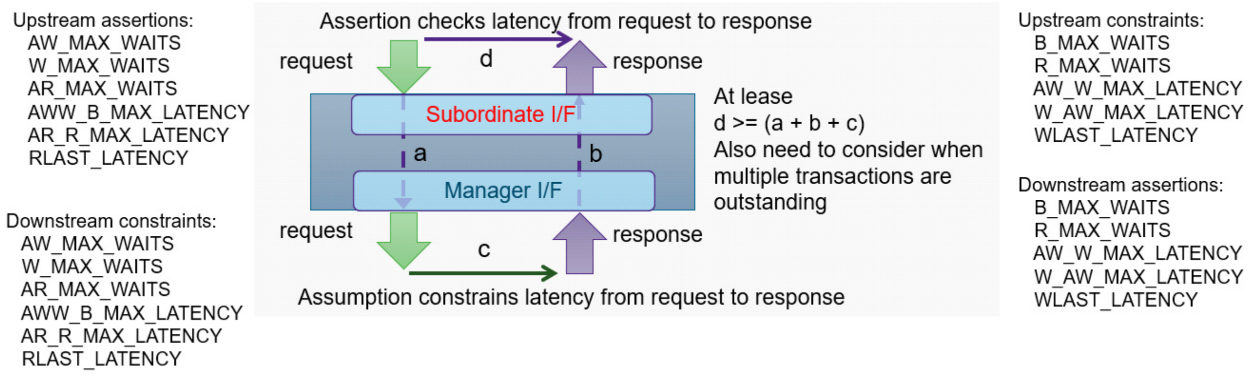
For the bug hunting purpose, when returning of response depends on other interface latency, try 2 types of configurations:

- 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.

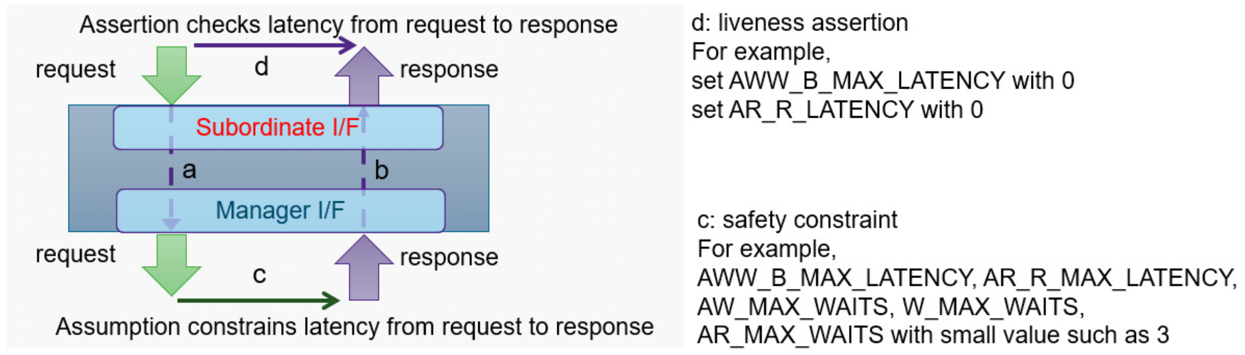
Figure 5-9 Setting Requester Latency Bigger Than Responder Latency



5.2.3.0.2 Example 2

Minimize safety constraints latencies and check with liveness assertions.

Figure 5-10 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 5-5 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 5-5 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.