

AHB Assertion IP User Guide

Version 2023.03, March 2023





Copyright Notice and Proprietary Information

© 2023 Synopsys, Inc. All rights reserved. This Synopsys software and all associated documentation are proprietary to Synopsys, Inc. and may only be used pursuant to the terms and conditions of a written license agreement with Synopsys, Inc. All other use, reproduction, modification, or distribution of the Synopsys software or the associated documentation is strictly prohibited.

Third-Party Software Notices

VCS® and configurations of VCS includes or is bundled with software licensed to Synopsys under free or open-source licenses. For additional information regarding Synopsys's use of free and open-source software, refer to the `third_party_notices.txt` file included within the `<install_path>/doc` directory of the installed VCS software.

Destination Control Statement

All technical data contained in this publication is subject to the export control laws of the United States of America. Disclosure to nationals of other countries contrary to United States law is prohibited. It is the reader's responsibility to determine the applicable regulations and to comply with them.

Disclaimer

SYNOPSYS, INC., AND ITS LICENSORS MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

Trademarks

Synopsys and certain Synopsys product names are trademarks of Synopsys, as set forth at <https://www.synopsys.com/company/legal/trademarks-brands.html>.

All other product or company names may be trademarks of their respective owners.

Third-Party Links

Any links to third-party websites included in this document are for your convenience only. Synopsys does not endorse and is not responsible for such websites and their practices, including privacy practices, availability, and content.

www.synopsys.com

Contents

Figures	5
Tables	7
Chapter	
Preface	9
Chapter 1	
Introduction	11
1.1 Prerequisites	11
1.2 References	11
1.3 Product Overview	11
1.4 Language and Methodology Support	12
1.5 Feature Support	12
1.5.1 Protocol Features	12
1.5.2 Verification Features	12
1.6 Features Not Supported	13
Chapter 2	
Installation and Setup	15
2.1 Verifying Hardware Requirements	15
2.2 Verifying Software Requirements	15
2.2.1 Platform/OS and Simulator Software	15
2.2.2 Synopsys Common Licensing (SCL) Software	15
2.2.3 Other Third Party Software	16
2.3 Preparing for Installation	17
Chapter 3	
The AHB AIP in a Formal Verification Environment	19
3.1 Introduction to the VC Formal Tool	19
3.2 The AHB AIP in a Formal Verification Environment	19
3.2.1 Instantiating The AHB AIP Using the bind Statement	20
3.2.2 Creating a Tcl File	20
3.2.3 Reading and Running a Tcl File	21
3.2.4 Commonly Used Configuration	25
3.2.5 The AHB AIP As a Master	25
3.2.6 The AHB AIP As a Slave	26
3.2.7 The AHB AIP As a Monitor	27
3.2.8 The AHB AIP In Constraint Mode	27
3.3 Clock and Reset Functionality	27

Chapter 4	29
The AHB AIP Configuration	29
4.1 The AHB AIP Configuration Parameters	29
4.2 The AHB AIP Interface Ports	32
4.3 The AHB AIP Properties	35
4.4 The AHB AIP Cover Properties	53
4.5 Description of Properties	56
4.6 Behavior of Properties	56
4.6.1 Properties In Assert Directives	56
4.6.2 Properties In Assume Directives	57
4.6.3 Properties In Cover Directives	57
Chapter 5	59
The AHB AIP Use Cases	59
5.1 The AHB AIP Examples	59
5.1.1 The AHB AIP with Slave DUT for AHB Lite	59
5.1.2 AHB AIP with Master DUT for AHB Lite	61

Figures

Figure 3-1:	VC Formal Tool showing Results of Properties	22
Figure 3-2:	VC Formal Tool Showing Options to Debug Failing Properties	23
Figure 3-3:	Waveforms Opened Through GUI Mode	23
Figure 3-4:	Waveforms of Failing Property Opened in the Batch Mode	24
Figure 3-5:	The AHB AIP As a Master	26
Figure 3-6:	The AHB AIP As a Slave	27
Figure 5-1:	Slave DUT: 1 Master 2 Slave (1 DUT and 1 AHB AIP)	60
Figure 5-2:	DUT is Master: 1 Master 2 Slave (Both AHB AIP)	62



Tables

Table 2-1:	AIP Licensing Key Features	16
Table 3-1:	Tcl File Example	21
Table 3-2:	Common Usage Models	25
Table 4-1:	The AHB AIP Configuration Parameters	29
Table 4-2:	The AHB AIP Interface Ports	32
Table 4-3:	: AHB AIP Properties	35
Table 4-4:	: AHB AIP Cover Properties	53
Table 5-1:	Bind Example: Slave DUT Connected to the AHB AIP	61
Table 5-2:	Bind Example: Master DUT with AHB AIP	63





Preface

This guide discusses the installation, setup, and usage for SystemVerilog Assertion IP(AIP) AMBA AHB, and is meant for design or verification engineers who want to verify RTL designs with an AMBA AHB interface. Readers are assumed to be familiar with the AMBA AHB protocol, SystemVerilog Assertions and Verilog language.

Guide Organization

The chapters of this guide are organized as follows:

Chapter 1, “[Introduction](#)”, introduces Synopsys AMBA AHB Assertion IP(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 AHB AIP.

Chapter 3, “[The AHB AIP in a Formal Verification Environment](#)”, introduces the formal tool usage and the AHB AIP in a formal (that is, static verification) environment.

Chapter 4, “[The AHB AIP Configuration](#)”, describes the programming or user interface ports, list of properties, and behavior of the AHB AIP.

Chapter 5, “[The AHB AIP Use Cases](#)”, shows how to install and run an example.

Web Resources

The AHB AIP is compliant with the following specifications:

- AMBA specification: AMBA ARM IHI 0011A and ARM IHI 0033A for AHB Lite and AHB Full, AMBA5 ARM IHI 0033B.b
- AMBA compliance protocol rules and coverage document
- AMBA FAQ document

Customer Support

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

- Open a case through SolvNet.
- Go to <https://onlinecase.synopsys.com/Support/OpenCase.aspx> and 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 the 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>

Introduction

This document describes the AHB protocol checkers available in the AHB Assertion IP (AIP). It also describes all parameters related to configurations, how to configure and instantiate the AHB AIP, and so on. Assertion IP is significant in reducing verification effort and improving design quality. Its value comes from the fact that assertions can passively monitor design behavior by simply monitoring a target design without modifying its RTL. In addition, AIPs are valuable because they describe design intent with the highest degree of clarity. Pre-built assertions from assertion IP provide powerful quality criteria for signoff.

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 AHB protocol, SystemVerilog Assertions, and SystemVerilog concepts.


1.2 References

The AHB AIP is compliant with the following specifications:

- AMBA specification: AMBA ARM IHI 0011A and ARM IHI 0033A for AHB Lite and AHB Full, AMBA5 ARM IHI 0033B.b
- AMBA FAQ document
(<http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.set.amba/index.html>)

1.3 Product Overview

The AHB AIP is a suite of SystemVerilog Assertions and Verilog modeling logic that are compatible for use with RTL. This AIP is used to verify RTL with the VC Formal tool.



The AHB AIP consists of the following:

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

1.4 Language and Methodology Support

The AHB AIP supports the following languages and methodology:

- SystemVerilog Assertions
- Verilog

1.5 Feature Support

1.5.1 Protocol Features

The AHB AIP currently supports the following protocol features:

- AHB and AHB Lite support
- Master, Slave, Arbiter and Decoder agents
- Decoder and multiplexer
- All data widths
- All address widths
- All transfer types
- All burst types and burst sizes
- All protection types
- All slave response types
- Support in master for rebuilding transactions
- Lock transactions

1.5.2 Verification Features

- The AHB Lite AIP supports the following verification features:
 - Single master and multiple slaves
 - AIP as master
 - AIP as slave
 - AIP as monitor
 - Protocol checks
 - Decoder, default slave and multiplexer features
 - Enabling and disabling of multiple features by controlling the parameters such as CONFIG_X_CHECK, ERROR_IDLE, LOCKIDLE_RCMND, ENABLE_ASSERT, ENABLE_ASSUME, and ENABLE_COVER.

- The AHB Full AIP supports the following verification features:
 - Multiple masters and multiple slaves
 - AIP as master
 - AIP as slave
 - AIP as monitor
 - AIP as arbiter
 - Protocol checks
 - Decoder, default slave and multiplexer features
 - Enabling and disabling of multiple features by controlling the parameters such as CONFIG_X_CHECK, ERROR_IDLE, LOCKIDLE_RCMND, ENABLE_ASSERT, ENABLE_ASSUME, and ENABLE_COVER.

1.6 Features Not Supported

- None



Installation and Setup

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

The checklist consists of the following major steps:

- [“Verifying Hardware Requirements”](#)
- [“Verifying Software Requirements”](#)
- [“Preparing for Installation”](#)

2.1 Verifying Hardware Requirements

The AHB AIP 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 AHB 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 is required

2.2.2 Synopsys Common Licensing (SCL) Software

The AHB AIP requires the following license feature:

AIP-xxx-SVA

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

2.2.2.1 Running the AHB AIP on the VC Formal Tool

To run the AHB 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 AHB AIP on VCS

To run the AHB 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}
```

Tables 2-1 lists the AIP licensing key features:

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

- HTML browser: You can view the coverage reports of the AHB AIP in HTML using the following browsers:
 - 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 APB AIP, see [“Synopsys Common Licensing \(SCL\) Software”](#) on page 15.



The AHB AIP in a Formal Verification Environment

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

- [“Introduction to the VC Formal Tool”](#) on page 19
- [“The AHB AIP in a Formal Verification 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 generating all possible combinations of valid inputs. These inputs are valid as these are constrained by assume properties. The VC Formal tool provides the following information:

- Number of proven properties
- Number of falsified properties
- Number of properties that passed vacuously
- Number of covered properties
- Number of properties (asserts, covers, and constraints) for which a witness (cover) property is created during the formal run

The VC Formal tool is useful for debugging failing properties by running it in the GUI mode.

3.2 The AHB AIP in a Formal Verification Environment

The AHB AIP has AHB master properties and the AHB slave properties. These properties are connected to either AHB master or slave module (for example, AHB slave DUT to master properties).

To verify the functional correctness of the module, use the VC Formal verification tool. To use the AHB AIP in a formal verification environment, perform the following steps in a sequence:

- Instantiating the AHB AIP using the `bind` statement
- Creating a Tcl file
- Reading and running a Tcl file

- Analyzing results

3.2.1 Instantiating The AHB AIP Using the bind Statement

Create a bind file to bind the AHB AIP with a design. Map modules and port names in the design with those of the AHB AIP in the bind statement. Pass valid values to the configuration parameters of the AHB AIP. The next step is to compile files. See [Table 3-1](#).



Note

If a signal corresponding to the AHB 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 `hprot` signal, set with `4'h0`.

3.2.1.1 Contents of Tcl File

To run the properties in the VC Formal tool, the following information specified in a Tcl script must be compiled in order to build the formal model:

- Path of the AHB AIP source files
- RTL source files for which protocol adherence needs to be verified
- Clock and reset information
- Proven or falsified property/protocol checks

3.2.2 Creating a Tcl File

To compile RTL files, AHB AIP files, and the bind file, create a Tcl file to set the path of the RTL directory, AHB AIP directory, and the bind file. The DUT clock and reset are initialized with the `create_clock` and `create_reset` commands, as shown in [Table 3-1](#). The VC Formal tool can report assertion status (proven, falsified, vacuous or inclusive using the `report_fv` command). For more command options, see the VC Formal User Guide.

3.2.2.1 Contents of Tcl File

[Table 3-1](#) describes the contents of a Tcl file.

Table 3-1 Tcl File Example

<pre> set AIP_HOME \$::env(AIP_HOME) # 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}/ahb_lite_test/tb set TCL_DIR \${TEST_DIR}/ahb_lite_test/tcl if { [file exists setup.tcl] == 1 } { source setup.tcl -echo -verbose } set hierarchy_delimiter "." # Timeout settings set_fml_var fml_max_time 5M </pre> <div data-bbox="248 940 719 1146"> <p>Either use the interactive debugging command or the batch regression command on the basis of your requirement.</p> </div>	<pre> # Commands to run the files in VC Static proc compd {} { global AIP_HOME AIP_SRC_DIR TEST_DIR TB_DIR read_file -sva -top dummy_dut_1m1s -format sverilog -vcs -sverilog +incdir+\${AIP_SRC_DIR} \${AIP_SRC_DIR}/snps_ahb_lite_master_aip.sv \${AIP_SRC_DIR}/snps_ahb_lite_slave_aip.sv \${AIP_SRC_DIR}/snps_ahb_aip_decoder_mux.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 # To generate logs when running a batch regression check_fv -block report_fv # To generate logs during an interactive debugging check_fv -run_finish { report_fv -list > \$RUN_DIR/run_config1.log } </pre>
---	--

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 23

3.2.3.1 GUI Mode

To read a 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, the VC Formal tool displays the list of properties (see [Figure 3-1](#)). In [Figure 3-1](#), the red cross indicates a falsified property and the green tick marks indicate proven properties.

Figure 3-1 VC Formal Tool showing Results of Properties

The screenshot shows the VC Formal Tool interface. The 'Task List' pane on the left shows a task 'AEP' with a progress bar. The main pane displays a table of verification targets and constraints.

status	depth	name	engine	elapsed_time	type
✓		snps_ahb_lite_master_aip.ahb_aip_logic.arith_overflow_1	e1	00:00:06	arith_overflow
✓		snps_ahb_lite_master_aip.ahb_aip_logic.arith_overflow_10	e1	00:00:06	arith_overflow
✓		snps_ahb_lite_master_aip.ahb_aip_logic.arith_overflow_11	t1	00:00:03	arith_overflow
✗	0	snps_ahb_lite_master_aip.ahb_aip_logic.arith_overflow_12	e2	00:00:03	arith_overflow
✗	0	snps_ahb_lite_master_aip.ahb_aip_logic.arith_overflow_13	e2	00:00:03	arith_overflow
✗	0	snps_ahb_lite_master_aip.ahb_aip_logic.arith_overflow_14	e2	00:00:03	arith_overflow
✗	0	snps_ahb_lite_master_aip.ahb_aip_logic.arith_overflow_15	e2	00:00:03	arith_overflow
✗	32	snps_ahb_lite_master_aip.ahb_aip_logic.arith_overflow_16	e1	00:00:12	arith_overflow

	name	usage	type	class	language
1	..._lite_master_aip.CFG_MASTER_ASSUME asm_snps_ahb_busy_allowed	assume	assume	source	SVA
2	...aster_aip.CFG_MASTER_ASSUME asm_snps_ahb_busy_last_only_incr	assume	assume	source	SVA
3	..._aip.CFG_MASTER_ASSUME asm_snps_ahb_fixlen_burst_seq_or_busy	assume	assume	source	SVA
4	...p.CFG_MASTER_ASSUME asm_snps_ahb_fixlen_burst_terminates_seq	assume	assume	source	SVA
5	...ster_aip.CFG_MASTER_ASSUME asm_snps_ahb_haddr_1kb_boundary	assume	assume	source	SVA
6	...lite_master_aip.CFG_MASTER_ASSUME asm_snps_ahb_haddr_aligned	assume	assume	source	SVA
7	...aster_aip.CFG_MASTER_ASSUME asm_snps_ahb_haddr_incr_address	assume	assume	source	SVA
8	..._master_aip.CFG_MASTER_ASSUME asm_snps_ahb_haddr_seq_burst	assume	assume	source	SVA

Properties: 19 - passed[10] - failed[9] - disabled[0]; Constraints Enabled: 1; Min depth: 1; Max depth: 32; Elapsed Time: 0:00:00

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, 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:

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, waveform is opened, providing signal details and falsification depth. You can dump other signals required for debugging into a wave as well.

You can also explore the options in the VC 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 falsified properties and waveform in the GUI mode respectively.

1

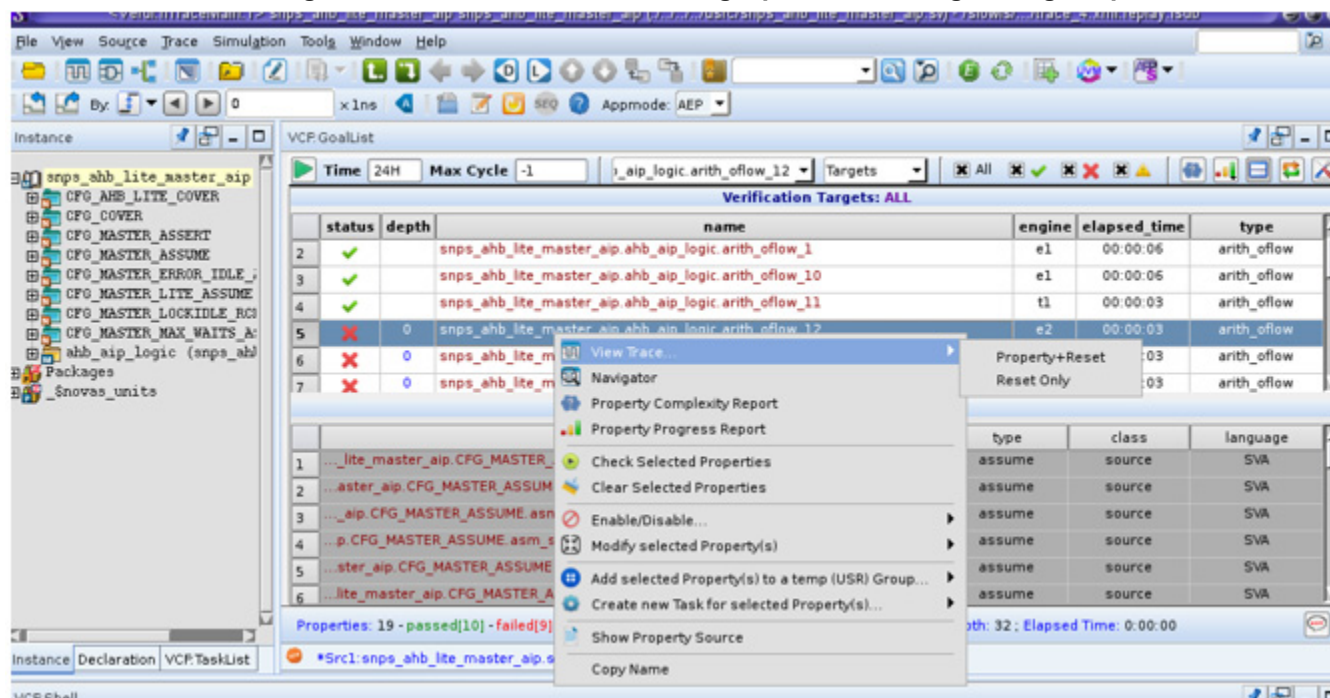
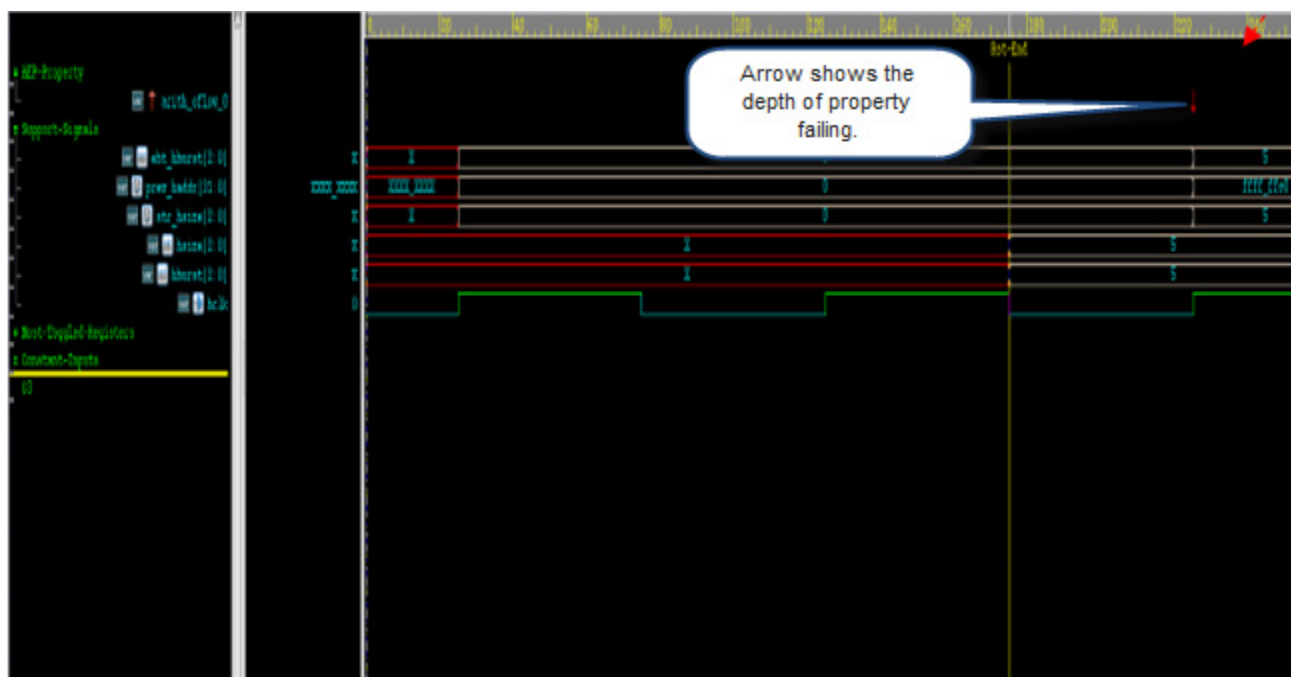


Figure 3-3 Waveforms Opened Through GUI Mode



3.2.3.2 Reading and Running Tcl File in the Batch Mode

To read a 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 vc_static_shell
```

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>
```

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 static 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`

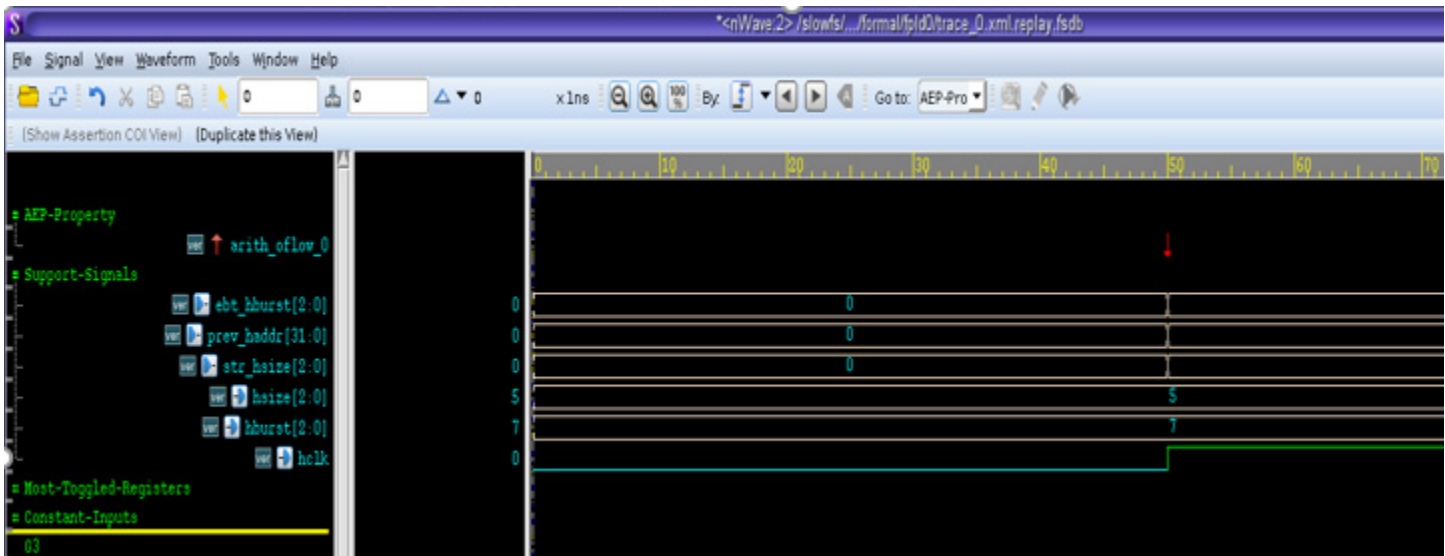
To display the number of falsified properties.

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

To open the VC Formal tool and to display the waveform of a falsified property which you want to debug.

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

Figure 3-4 Waveforms of Failing Property Opened in the Batch Mode



3.2.4 Commonly Used Configuration

Table 3-2 Common Usage Models

1	Master	Instantiates the AHB 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 AHB 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 AHB 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 AHB 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 AHB AIP As a Master

To verify an AHB slave DUT, set the `AGENT_TYPE` parameter to `MASTER` during the AHB AIP instantiation. When the AHB AIP parameter is set as `MASTER`, all properties that are related to the master inputs are declared as `assert`, and all properties that are related to the master outputs are declared as `assume`. This is required to make the AHB AIP behave as a master.

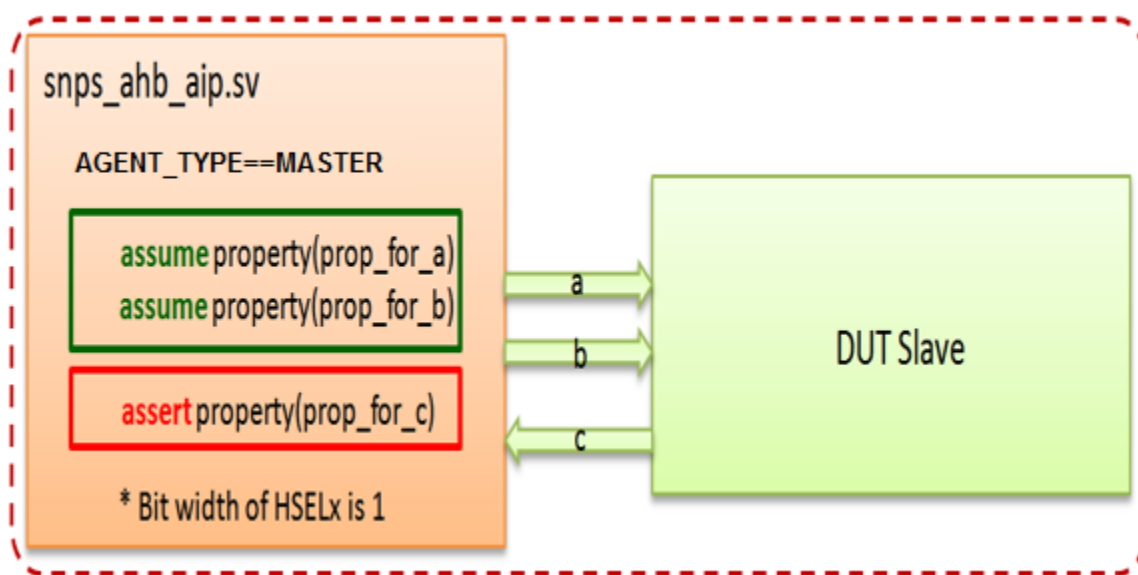
To disable all `assert`, `assume`, or `cover` properties, explicitly set the following parameters to value 0:

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

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

[Figure 3-5](#) checks the behavior of an AHB slave DUT output and constraints the inputs of an AHB slave DUT to valid values.

Figure 3-5 The AHB AIP As a Master



3.2.6 The AHB AIP As a Slave

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

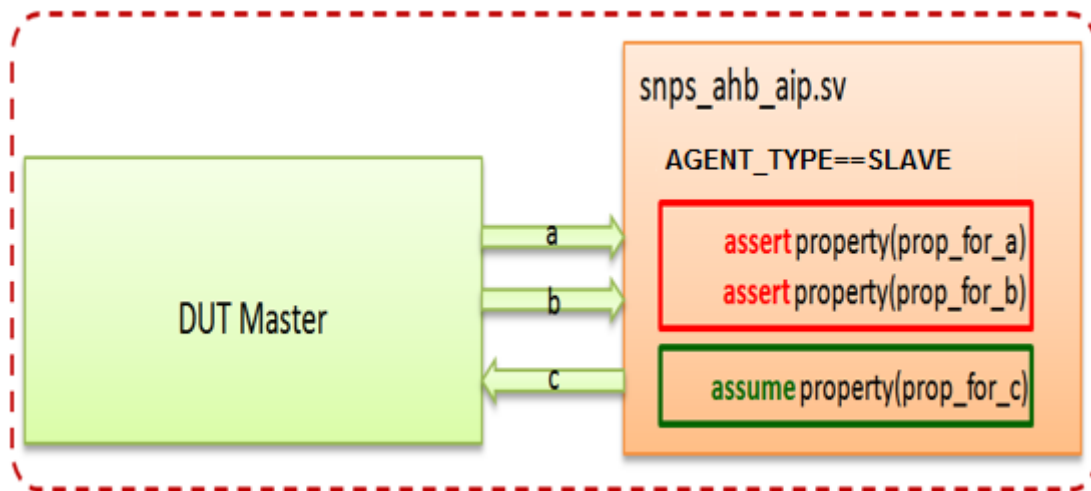
To disable all `assert`, `assume`, or `cover` properties, explicitly set the following parameters to value 0:

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

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

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

Figure 3-6 The AHB AIP As a Slave



3.2.7 The AHB AIP As a Monitor

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

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

3.2.8 The AHB AIP In Constraint Mode

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

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

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

3.3 Clock and Reset Functionality

- To run the AHB 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 AHB 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.



The AHB AIP Configuration

This chapter describes about configuration of the AHB AIP in the following Sections

- [“The AHB AIP Configuration Parameters”](#) on page 29
- [“The AHB AIP Interface Ports”](#) on page 32
- [“The AHB AIP Properties”](#) on page 35
- [“Description of Properties”](#) on page 56
- [“Behavior of Properties”](#) on page 56

4.1 The AHB AIP Configuration Parameters

[Table 4-1](#) shows the user-defined parameters for setting the interface characteristics of AHB Lite and AHB Full. You can change these parameters to match your design specification.

Table 4-1 The AHB AIP Configuration Parameters

Parameter	Default Value	Description
AGENT_TYPE	MASTER	agent type: either MASTER or MONITOR
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
CONFIG_X_CHECK	0	1: Enable 0: Disable X-checks
LOCKIDLE_RCMND	1	1: Issue IDLE after Locked transfer (Recommendation)
CHECK_FORMAL	1	1: Use Formal 0: Use Simulation/Emulation
ADDR_WIDTH	32	Bit width of HADDR
DATA_WIDTH	32	Bit width of HWDATA/HRDATA
NUM_MASTERS	16	Indicates how many masters exist
ERROR_IDLE	1	For ERROR response 1: Master stops burst and drives IDLE 0: Master continues burst

Table 4-1The AHB AIP Configuration Parameters

Parameter	Default Value	Description
CHECK_MAX_WAITS	1	1: Enable 0: Disable HREADY latency
MAX_WAITS	16	Maximum number of cycles for HREADY latency. >0: Check HREADY should be returned within the specified cycles (Safety property). 0: Check HREADY should be returned eventually (Liveness property).
CHECK_MIN_WAITS	0	1: Enable 0: Disable HREADY minimum latency
MIN_WAITS	0	Minimum number of cycles for HREADY latency
ALLOW_ERROR	1	1: allow ERROR response 0: no ERROR
ARB_ALLOW_EBT	1	1: arb_allow EBT (Early Burst Termination) 0: no EBT (Except for AHB5)
ALLOW_EBT	1	1: arb_allow EBT (Early Burst Termination) 0: no EBT (AHB5 only)
CHECK_BUSY_MAX	1	1: Enable 0: Disable maximum BUSY cycle check
BUSY_MAX	16	Maximum number of BUSY cycles
CHECK_INCR_MAX	1	1: Enable 0: Disable maximum INCR (NONSEQ) burst check
INCR_MAX	16	Maximum number of continuous INCR burst
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'hfffffff}	Maximum address for each address range: for example {32'h3fff, 32'hfffffff} in case of ADDR_RANGE=2
STRB_MODE	0	data stable check strobe calculation mode: 0: all bits 1: LE8/BE8 (Little Endian) 2: BE32 (Big Endian)
HMASTER_BIT	4	Bit width of HMASTER
ALLOW_SPLIT	1	1: allow SPLIT response 0: no SPLIT (AHB Full only)
ALLOW_RETRY	1	1: allow RETRY response 0: no RETRY (AHB Full only)
PROT_WIDTH	7	Bit width of HPROT (AMBA2: 4 ARMv6: 6 AMBA5: 7)
NUM_EXCL	4	Maximum number of outstanding Exclusive transaction (AHB5 only)
EXCL_ADDR_LSB	2	Indicates Exclusive access granularity (for example 2: 32 bits 3: 64 bits 4: 128 bits) (AHB5 only)
EXWR_AFTER_EXRD	0	0: disable 1: enable check Exclusive Write should occur only after corresponding Exclusive Read (AHB5 only)

Table 4-1The AHB AIP Configuration Parameters

Parameter	Default Value	Description
CONFIG_USER	0	0: disable 1: enable user signals (AUSER WUSER RUSER) related checks (AHB5 only)
AUSER_WIDTH	32	Bit width of AUSER (AHB5 only)
WUSER_WIDTH	32	Bit width of WUSER (AHB5 only)
RUSER_WIDTH	32	Bit width of RUSER (AHB5 only)
FRV	0	Enable or disable FRV interface 0: Disable FRV interface 1: Enable FRV interface
FRV_RSVD_MODE	1	Reserved mode 0: no check when ACCESS_TYPE==ACS_RSVD 1: 0 should be read when ACCESS_TYPE==ACS_RSVD and CHECK_INIT==0 2: 1 should be read when ACCESS_TYPE==ACS_RSVD and CHECK_INIT==0
FRV_WR_LATENCY	1	Default Write Latency bus write propagates to BackDoor signal
FRV_RD_LATENCY	1	Default Read Latency from BackDoor signal to bus read
FRV_LATENCY_MD	0	Latency handling 0: WR/RD_LATENCY parameters are used only in back door check 1: WR/RD_LATENCY parameters are used in all checks
FRV_OUTABLK_MD	2	Out of addressBlock range check 0: check read data should be 0 for out of addressBlock range 1: check read data should be 1 for out of addressBlock range others: no check for out of addressBlock range
FRV_RD_ACT_LATENCY	1	readAction Latency
FRV_RO_RD_CHK	0	Setting for read-only field check. 0: Check if read data should be reset value (default). Enable read data check only when the reset value is specified. 1: Check if read data should be the same as the previously read data. Enable read data check regardless of whether the reset value is specified or not.
FRV_COMPOSITE	1	Type of assertion per field for initial value and after write. 1: Generate 1 common assertion (better convergence). 0: Generate 2 individual assertions.
FRV_CHECK_FORMAL	0	Use (1) or do not use (0) Formal optimized property.

Table 4-1 The AHB AIP Configuration Parameters

Parameter	Default Value	Description
FRV_SNPS_ND_ABS	0	Synopsys Non-Deterministic Abstractions 0: check all possible scenarios 1: check only specific scenario in each step
FRV_COVER_WRITE	1	Enable/Disable cover properties for write transactions 0: Disable all write related cover properties 1: Enable write related cover properties
FRV_COVER_READ	1	Enable/Disable cover properties for read transactions 0: Disable all read related cover properties 1: Enable read related cover properties
FRV_COVER_TYPE	3'h7	Enable/Disable cover properties per type FRV_COVER_WRITE==1 3'bx1: Enable cover write between write to read window 3'bx1x: Enable cover write before first write 3'b1xx: Enable cover write before first read FRV_COVER_WRITE==0 Disable all write cover properties FRV_COVER_READ==1 3'bx1: Enable cover read between write to read window 3'bx1x: Enable cover read before first write 3'b1xx: Enable cover read before first read FRV_COVER_READ==0 Disable all read cover properties
FRV_OUTADDR_EN	1	Out of address range cover properties 0: Disable out of address cover properties 1: Enable out of address cover properties

4.2 The AHB AIP Interface Ports

Table 4-2 describes all the interface signals of AHB Lite and AHB Full.

Table 4-2 The AHB AIP Interface Ports

	Signal Name	Signal Width	AHB Full/Lite/ARMv6/AMBA5	Description
1	hclk	1	All	The hclk signal is an input clock from the system.
2	hresetn	1	All	The hresetn a signal reset input from the system.
3	hbusreq	1	AHB Full	The hbusreq signal is an output port signal of the master.

Table 4-2 The AHB AIP Interface Ports

	Signal Name	Signal Width	AHB Full/Lite/ARMv6/AMBA5	Description
4	<code>hgrant</code>	1	AHB Full	The <code>hgrant</code> signal is an input grant from an arbiter to the master.
5	<code>hlock</code>	1	AHB Full	The <code>hlock</code> signal is an output signal of the master.
6	<code>htrans</code>	2	All	The <code>htrans</code> signal is an output signal of the master and input to the slave.
7	<code>haddr</code>	ADDR_WIDTH	All	The <code>haddr</code> signal is an output address bus of the master and input to the slave.
8	<code>hwrite</code>	1	All	The <code>hwrite</code> signal is an output signal of the master and input to the slave.
9	<code>hsize</code>	3	All	The <code>hsize</code> signal is an output signal of the master and input to the slave.
10	<code>hburst</code>	3	All	The <code>hburst</code> signal is an output signal of the master and input to the slave.
11	<code>hprot</code>	4 for AHB Full/Lite 6 for ARMv6 7 for AMBA5	All	The <code>hprot</code> signal is an output signal of the master and input to the slave.
12	<code>hwdata</code>	DATA_WIDTH	All	The <code>hwdata</code> signal is an output data bus of the master and input data bus to the slave.
13	<code>hready</code>	1	All	The <code>hready</code> signal is an input to both the master and the slave.
14	<code>hresp</code>	2 for AHB Full 1 for AHB Lite and AMBA5 3 for ARMv6	All	The <code>hresp</code> signal is an input to the master and output data from the slave.
15	<code>hrdata</code>	DATA_WIDTH	All	The <code>hrdata</code> signal is an input data bus to the master and output data from the slave.
16	<code>hmastlock</code>	1	All except for AHB Full MASTER	The <code>hmastlock</code> signal is an input signal to the slave.

Table 4-2 The AHB AIP Interface Ports

	Signal Name	Signal Width	AHB Full/Lite/ARMv6/AMBA5	Description
17	hreadyout	1	All for SLAVE N/A for MASTER	The hreadyout signal is an output signal to the slave.
18	hsplit	NUM_MASTER	AHB Full	The hsplit signal is an output signal from the slave.
19	hmaster	HMASTER_BIT	AHB Full and AMBA5	The hmaster signal is an input signal to the slave.
20	hsel	1	All for SLAVE N/A for MASTER	The hsel signal is an input signal to the slave.
21	hbstrb	STRB_WIDTH	ARMv6	The hbstrb signal is an output from the master, and denotes write byte strobe.
22	hunalign	1	ARMv6	The hunalign signal is an output signal from the master, and denotes unaligned address access.
23	hnonse	1	AMBA5	The hnonsec signal is output from the master, and denotes Non-secure access.
24	hexcl	1	AMBA5	The hexcl signal is output from the master, and denotes Exclusive access.
25	hexokay	1	AMBA5	The hexokay signal is output from the slave, and indicates Exclusive access is success or fail.
26	hauser	AUSER_WIDTH	AMBA5	The hauser signal is output from the master. Address user signal.
27	hwuser	WUSER_WIDTH	AMBA5	The hwuser signal is output from the master. Write data user signal.
28	hruser	RUSER_WIDTH	AMBA5	The hruser signal is output from the slave. Read data user signal.

4.3 The AHB AIP Properties

This Section describes the property name, type and behavior of the AHB AIP. The AHB Full properties have AHB_LITE_EN=0 parameter set. [Table 4-3](#) includes only assert and assume properties for Master, Slave and Arbiter. [Table 4-4](#) includes cover properties, which are independent of assert and assume properties.

Table 4-3: AHB AIP Properties

Property Name	Agent	Severity	Spec Reference	Property Description
ast_snps_ahb_hsplit_allowed	SLAVE	ERROR	Section 3.12 on page 3-35 (IHI 0011A)	When the slave can complete the transfer, it asserts the appropriate bit, according to the master number, on the HSPLITx[15:0] signals from the slave to the arbiter.
ast_snps_ahb_hsplit_pulse	SLAVE	ERROR	Section 3.12 on page 3-35 (IHI 0011A)	The arbiter samples the HSPLITx bus every cycle and therefore the slave only needs to assert the appropriate bit for a single cycle in order for the arbiter to recognize it.
ast_snps_ahb_hsplit_onehot0	SLAVE	ERROR	Section 3.12 on page 3-35 (IHI 0011A)	When the slave can complete the transfer, it asserts the appropriate bit, according to the master number, on the HSPLITx[15:0] signals from the slave to the arbiter.
ast_snps_ahb_busy_last_only_incr	MASTER	ERROR	Section 3.5 on page 3-9 (IHI 0011A), Section 3.2 on page 3-5 (IHI 0033A), Section 3.2 on page 3-30 (IHI0033B.b)	Only undefined length burst can have a BUSY transfer as the last cycle of burst. HTRANS should not be BUSY at the last cycle of burst except for HBURST is INCR.
ast_snps_ahb_seq_allowed	MASTER	ERROR	Section 3.5 Table 3-1 on page 3-9 (IHI 0011A), Section 3.2 Table 3-1 on page 3-5 (IHI 0033A), Section 3.2 Table 3-1 on page 3-30 (IHI0033B.b)	The remaining transfers in a burst are SEQUENTIAL. HTRANS should be SEQ until all beats of transfer completes.
ast_snps_ahb_busy_allowed	MASTER	ERROR	Section 3.5 Table 3-1 on page 3-9 (IHI 0011A), Section 3.2 Table 3-1 on page 3-5 (IHI 0033A), Section 3.2 Table 3-1 on page 3-30 (IHI0033B.b)	The BUSY transfer type enables masters to insert idle cycles in the middle of a burst. HTRANS can be BUSY only after HTRANS is NONSEQ or SEQ.
ast_snps_ahb_htrans_seq_length	MASTER	ERROR	Section 3.5 Table 3-1 on page 3-9 (IHI 0011A), Section 3.2 Table 3-1 on page 3-5 (IHI 0033A), Section 3.2 Table 3-1 on page 3-30 (IHI0033B.b)	Fixed length burst should complete with specified length, so SEQ will not last more than specified burst length.

Table 4-3: AHB AIP Properties

Property Name	Agent	Severity	Spec Reference	Property Description
ast_snps_ahb_single_burst_nonseq	MASTER	ERROR	Section 3.5 on page 3-9 (IHI 0011A), Section 3.2 on page 3-5 (IHI 0033A), Section 3.2 on page 3-30 (IHI0033B.b)	Single transfer type is NONSEQ. HTRANS can be SEQ or BUSY only when HBURST is not SINGLE.
ast_snps_ahb_htrans_wait_stable	MASTER	ERROR	Section 3.4 on page 3-7 (IHI 0011A), Section 3.6.1 on page 3-16 (IHI 0033A), Section 3.6.1 on page 3-39 (IHI0033B.b)	When the slave is requesting wait states, the master must not change the transfer type (HTRANS), except as described in: IDLE transfer, BUSY transfer, fixed length burst, BUSY transfer, undefined length burst.
ast_snps_ahb_hburst_wait_stable	MASTER	ERROR	Section 3.4 on page 3-7 (IHI 0011A), Section 3.6 on page 3-16 (IHI 0033A), Section 3.6.1 on page 3-39 (IHI0033B.b)	During a waited transfer, the master is restricted to change control signals. HBURST signal should not be changed while HREADY is 0.
ast_snps_ahb_hsize_wait_stable	MASTER	ERROR	Section 3.4 on page 3-7 (IHI 0011A), Section 3.6 on page 3-16 (IHI 0033A), Section 3.6.1 on page 3-39 (IHI0033B.b)	During a waited transfer, the master is restricted to change control signals. HSIZE signal should not be changed while HREADY is 0.
ast_snps_ahb_hwrite_wait_stable	MASTER	ERROR	Section 3.4 on page 3-7 (IHI 0011A), Section 3.6 on page 3-16 (IHI 0033A), Section 3.6.1 on page 3-39 (IHI0033B.b)	During a waited transfer, the master is restricted to change control signals. HWRITE signal should not be changed while HREADY is 0.
ast_snps_ahb_hprot_wait_stable	MASTER	ERROR	Section 3.4 on page 3-7 (IHI 0011A), Section 3.6 on page 3-16 (IHI 0033A), Section 3.6.1 on page 3-39 (IHI0033B.b)	During a waited transfer, the master is restricted to change control signals. HPROT signal should not be changed while HREADY is 0.
ast_snps_ahb_haddr_wait_stable	MASTER	ERROR	Section 3.4 on page 3-7 (IHI 0011A), Section 3.6.2 on page 3-19 (IHI 0033A), Section 3.6.2 on page 3-42 (IHI0033B.b)	When the slave is requesting wait states, the master must not change the address once, except as described in: during an IDLE transfer, after an ERROR response.
ast_snps_ahb_hwdata_wait_stable	MASTER	ERROR	Section 3.4 on page 3-7 (IHI 0011A), Section 3.1 on page 3-4 (IHI 0033A), Section 3.6.1 on page 3-39 (IHI0033B.b)	The master holds the data stable throughout the extended cycles. HWDATA signal should not be changed while HREADY is 0.

Table 4-3: AHB AIP Properties

Property Name	Agent	Severity	Spec Reference	Property Description
ast_snps_ahb_hlock_after_ebt	MASTER	ERROR	Section 3.6.1 on page 3-12 (IHI 0011A)	If a bus master cannot complete a burst because it loses ownership of the bus, then it must rebuild the burst appropriately when it next gains access to the bus. HLOCK signal is different from transfer before Early Burst Termination.
ast_snps_ahb_hlock_split_retry	MASTER	ERROR	Section 3.9.5 on page 3-24 (IHI 0011A)	It should continue to request the bus and attempt the transfer until it has either completed successfully or been terminated with an RETRY or SPLIT response. HLOCK signal is not as expected after RETRY or SPLIT.
ast_snps_ahb_hmastlock_valid_level	MASTER	ERROR	Section 3.13 on page 3-40 (IHI 0011A), Section 7.1.2 on page 7-2 (IHI 0033A), Section 7.1.2 on page 7-68 (IHI0033B.b)	During reset all masters must ensure the address and control signals are at valid levels. A value of X/Z on HMASTLOCK is not permitted.
ast_snps_ahb_hsel_valid_level	DECODER	ERROR	Section 3.13 on page 3-40 (IHI 0011A), Section 7.1.2 on page 7-2 (IHI 0033A), Section 7.1.2 on page 7-68 (IHI0033B.b)	During reset all masters must ensure the address and control signals are at valid levels. A value of X/Z on HSEL is not permitted.
ast_snps_ahb_hsplit_valid_level	SLAVE	ERROR	Section 3.13 on page 3-40 (IHI 0011A)	During reset all slaves ensures signals are at valid levels. A value of X/Z on HSPLIT is not permitted.
ast_snps_ahb_nosel_okay_response	SLAVE	ERROR	Section 3.5 on page 3-9 (IHI 0011A), Section 3.2 on page 3-5 (IHI 0033A), Section 3.2 on page 3-30 (IHI 0033B.b)	Slave must always provide a zero wait state OKAY response when slave is not selected.
ast_snps_ahb_idle_okay_response	SLAVE	ERROR	Section 3.5 on page 3-9 (IHI 0011A), Section 3.2 on page 3-5 (IHI 0033A), Section 3.2 on page 3-30 (IHI 0033B.b)	Slave must always provide a zero wait state OKAY response to IDLE transfers.
ast_snps_ahb_busy_okay_response	SLAVE	ERROR	Section 3.5 on page 3-9 (IHI 0011A), Section 3.2 on page 3-5 (IHI 0033A), Section 3.2 on page 3-30 (IHI 0033B.b)	Slave must always provide a zero wait state OKAY response to BUSY transfers.

Table 4-3: AHB AIP Properties

Property Name	Agent	Severity	Spec Reference	Property Description
ast_snps_ahb_hreadyout_reset_high	SLAVE	ERROR	Section 3.13 on page 3-40 (IHI 0011A), Section 7.1.2 on page 7-2 (IHI 0033A), Section 7.1.2 on page 7-68 (IHI 0033B.b)	During reset all slaves must ensure that HREADYOUT is High.
ast_snps_ahb_error_response_allowed	SLAVE	ERROR	Section 3.9.3 on page 3-22 (IHI 0011A), Section 5.1 on page 5-2 to 5-4 (IHI 0033A), Section 5.1 on page 5-56 (IHI0033B.b)	An error has occurred during the transfer. The error condition must be signaled to the master so that it is aware the transfer has been unsuccessful. ERROR response should not be returned when there's no request.
ast_snps_ahb_error_response_1st_cycle	SLAVE	ERROR	Section 3.9.3 on page 3-22 (IHI 0011A), Section 5.1 on page 5-2 to 5-4 (IHI 0033A), Section 5.1 on page 5-56 (IHI0033B.b)	A two-cycle response is required for an ERROR with HREADYOUT being de-asserted in the first cycle. HREADY should be 0 at the first cycle of ERROR response.
ast_snps_ahb_error_response_2nd_cycle	SLAVE	ERROR	Section 3.9.3 on page 3-22 (IHI 0011A), Section 5.1 on page 5-2 to 5-4 (IHI 0033A), Section 5.1 on page 5-56 (IHI0033B.b)	A two-cycle response is required for an ERROR with HREADYOUT being asserted in the second cycle. HREADY should be 1 and HRESP should be ERROR at the second cycle of ERROR response.
ast_snps_ahb_error_response_2_cycles	SLAVE	ERROR	Section 3.9.3 on page 3-22 (IHI 0011A), Section 5.1 on page 5-2 to 5-4 (IHI 0033A), Section 5.1 on page 5-56 (IHI0033B.b)	A two-cycle response is required for an ERROR with HREADYOUT being asserted in the second cycle. (HREADY==1 && HRESP==ERROR) should not be seen at the next cycle of HREADY==1 && HRESP==ERROR.
ast_snps_ahb_hreadyout_write_max_waits	SLAVE	ERROR	Section 3.9.1 on page 3-20 (IHI 0011A), Section 5.1.2 on page 5-3 (IHI 0033A), Section 5.1.2 on page 5-57 (IHI0033B.b)	Every slave must have a predetermined maximum number of wait states. It is recommended that slaves do not insert more than %6d wait states.
ast_snps_ahb_hreadyout_read_max_waits	SLAVE	ERROR	Section 3.9.1 on page 3-20 (IHI 0011A), Section 5.1.2 on page 5-3 (IHI 0033A), Section 5.1.2 on page 5-57 (IHI0033B.b)	Every slave must have a predetermined maximum number of wait states. It is recommended that slaves do not insert more than %6d wait states.

Table 4-3: AHB AIP Properties

Property Name	Agent	Severity	Spec Reference	Property Description
ast_snps_ahb_hreadyout_write_eventually	SLAVE	ERROR	section 3.9.1 on page 3-20 (IHI 0011A), section 5.1.2 on page 5-3 (IHI 0033A), section 5.1.2 on page 5-57 (IHI0033B.b)	Every slave must have a predetermined maximum number of wait states.
ast_snps_ahb_hreadyout_read_eventually	SLAVE	ERROR	section 3.9.1 on page 3-20 (IHI 0011A), section 5.1.2 on page 5-3 (IHI 0033A), section 5.1.2 on page 5-57 (IHI0033B.b)	Every slave must have a predetermined maximum number of wait states.
ast_snps_ahb_hresp_valid_level	SLAVE	ERROR	Section 3.13 on page 3-40 (IHI 0011A), Section 7.1.2 on page 7-2 (IHI 0033A)	During reset all slaves ensures signals are at valid levels. A value of X/Z on HRESP is not permitted.
ast_snps_ahb_hready_valid_level	SLAVE	ERROR	Section 3.13 on page 3-40 (IHI 0011A), Section 7.1.2 on page 7-2 (IHI 0033A), Section 7.1.2 on page 7-68 (IHI0033B.b)	During reset all slaves ensures signals are at valid levels. A value of X/Z on HREADY is not permitted.
ast_snps_ahb_hrdata_valid_level	SLAVE	ERROR	Section 3.10.2 on page 3-25 (IHI 0011A), Section 6.1.2 on page 6-2 (IHI 0033A), Section 6.1.2 on page 6-60 (IHI0033B.b)	The slave only has to provide valid data in the final cycle of the transfer. A value of X/Z on HRDATA is not permitted.
ast_snps_ahb_retry_response_allowed	SLAVE	ERROR	Section 3.9.3 on page 3-22 (IHI 0011A)	RETRY response is allowed as 2 cycle responses during data phase. RETRY response should not be returned when there's no request.
ast_snps_ahb_retry_response_1st_cycle	SLAVE	ERROR	Section 3.9.3 on page 3-32 (IHI 0011A)	A two-cycle response is required for an RETRY with HREADY being de-asserted in the first cycle. HREADY should be 0 at the first cycle of RETRY response.
ast_snps_ahb_retry_response_2nd_cycle	SLAVE	ERROR	Section 3.9.3 on page 3-22 (IHI 0011A)	A two-cycle response is required for an RETRY with HREADY being asserted in the second cycle. HREADY should be 1 and HRESP should be RETRY at the second cycle of RETRY response.

Table 4-3: AHB AIP Properties

Property Name	Agent	Severity	Spec Reference	Property Description
ast_snps_ahb_retry_response_2_cycles	SLAVE	ERROR	Section 3.9.3 on page 3-22 (IHI 0011A)	A two-cycle response is required for an RETRY with HREADY being asserted in the second cycle. (HREADY==1 && HRESP==RETRY) should not be seen at the next cycle of HREADY==1 && HRESP==RETRY.
ast_snps_ahb_split_response_allowed	SLAVE	ERROR	Section 3.9.3 on page 3-22 (IHI 0011A)	SPLIT response is allowed as 2 cycle responses during data phase. SPLIT response should not be returned when there's no request.
ast_snps_ahb_split_response_1st_cycle	SLAVE	ERROR	Section 3.9.3 on page 3-22 (IHI 0011A)	A two-cycle response is required for an SPLIT with HREADY being de-asserted in the first cycle. HREADY should be 0 at the first cycle of SPLIT response.
ast_snps_ahb_split_response_2nd_cycle	SLAVE	ERROR	Section 3.9.3 on page 3-22 (IHI 0011A)	A two-cycle response is required for an SPLIT with HREADY being asserted in the second cycle. HREADY should be 1 and HRESP should be SPLIT at the second cycle of SPLIT response.
ast_snps_ahb_split_response_2_cycles	SLAVE	ERROR	Section 3.9.3 on page 3-22 (IHI 0011A)	A two-cycle response is required for an SPLIT with HREADY being asserted in the second cycle. (HREADY==1 && HRESP==SPLIT) should not be seen at the next cycle of HREADY==1 && HRESP==SPLIT.
ast_snps_ahb_hresp_no_split_retry	SLAVE	ERROR	Section 8.3.6 on page 8-15 (ARM DDI 0211K)	No HSPLIT and RETRY responses. HRESP should not indicate HSPLIT or RETRY.
ast_snps_ahb_xfail_valid_response	SLAVE	ERROR	Section 8.3.6 on page 8-15 (ARM DDI 0211K)	It is not possible to indicate a combination of either Error, Retry, or Split with Xfail. The values b101, b110, and b111 are not valid responses in HRESP.
ast_snps_ahb_xfail_response_allowed	SLAVE	ERROR	Section 8.3.6 on page 8-15 (ARM DDI 0211K)	Slave should not respond with Xfail response for non Exclusive access in HRESP.

Table 4-3: AHB AIP Properties

Property Name	Agent	Severity	Spec Reference	Property Description
ast_snps_ahb_xfail_response_1st_cycle	SLAVE	ERROR	Section 8.3.8 Figure 8-4 on page 8-20 (ARM DDI 0211K)	A two-cycle response is required for an XFAIL response with HREADY being de-asserted in the first cycle. HREADY should be 0 at the first cycle of HRESP==XFAIL.
ast_snps_ahb_xfail_response_2nd_cycle	SLAVE	ERROR	Section 8.3.8 Figure 8-4 on page 8-20 (ARM DDI 0211K)	A two-cycle response is required for an XFAIL response with HREADY being asserted in the second cycle. HREADY should be 1 at the second cycle of HRESP==XFAIL.
ast_snps_ahb_xfail_response_2_cycles	SLAVE	ERROR	Section 8.3.8 Figure 8-4 on page 8-20 (ARM DDI 0211K)	A two-cycle response is required for an XFAIL response. (HREADY==1 && HRESP==XFAIL) should not be seen at the next cycle of HREADY==1 && HRESP==XFAIL.
ast_snps_ahb_hruser_valid_level	SLAVE	ERROR	Section 10.1 on page 10-80 (IHI0033B.b)	HRUSER signal has the same timing and validity requirements as the associated read data channel. A value of X/Z on HRUSER is not permitted.
ast_snps_ahb_excl_hexokay_allows	SLAVE	ERROR	Section 8.4 on page 8-73 (IHI0033B.b)	Slave should not assert HEXOKAY for Non Exclusive access.
ast_snps_ahb_excl_write_exfail	SLAVE	ERROR	Section 8.4 on page 8-73 (IHI0033B.b)	It is permitted for a master to issue an Exclusive Write transfer, which has not been preceded by an Exclusive Read transfer in the same Exclusive access sequence. In this case, the Exclusive Write transfer must fail and the HEXOKAY response signal must be de-asserted.
ast_snps_ahb_excl_hexokay_not_supported	SLAVE	ERROR	Section 8.3.1 on page 8-72 (IHI0033B.b)	When de-asserted HEXOKAY indicates that the Exclusive Transfer has failed. This can be because an Exclusive Transfer has been attempted to an address location that does not support Exclusive Transfers.

Table 4-3: AHB AIP Properties

Property Name	Agent	Severity	Spec Reference	Property Description
ast_snps_ahb_hresp_haddr_out_of_range	SLAVE	ERROR	Section 3.8 on page 3-19 (IHI 0011A), Section 4.1 on page 4-2 (IHI 0033A), Section 4.2.1 on page 4-53 (IHI0033B.b)	If a NONSEQUENTIAL or SEQUENTIAL transfer is attempted to a nonexistent address location then the default slave provides an ERROR response.
ast_snps_ahb_hburst_burst_constant	MASTER	ERROR	Section 3.7 on page 3-17 (IHI 0011A), Section 3.2 on page 3-5 (IHI 0033A), Section 3.5 on page 3-34 (IHI0033B.b)	The control information is identical to the previous transfer during SEQ. HBURST signal must remain constant throughout a burst transfer.
ast_snps_ahb_hsize_burst_constant	MASTER	ERROR	Section 3.7 on page 3-17 (IHI 0011A), Section 3.4 on page 3-8 (IHI 0033A), Section 3.5 on page 3-34 (IHI0033B.b)	The control information is identical to the previous transfer during SEQ. HSIZE signal must remain constant throughout a burst transfer.
ast_snps_ahb_hwrite_burst_constant	MASTER	ERROR	Section 3.7 on page 3-17 (IHI 0011A), Section 3.2 on page 3-5 (IHI 0033A), Section 3.5 on page 3-34 (IHI0033B.b)	The control information is identical to the previous transfer during SEQ. HWRITE signal must remain constant throughout a burst transfer.
ast_snps_ahb_hprot_burst_constant	MASTER	ERROR	Section 3.7 on page 3-17 (IHI 0011A), Section 3.2 on page 3-5 (IHI 0033A), Section 3.5 on page 3-34 (IHI0033B.b)	The control information is identical to the previous transfer during SEQ. HPROT signal must remain constant throughout a burst transfer.
ast_snps_ahb_hsize_max	MASTER	ERROR	Section 3.16.1 on page 3-43 (IHI 0011A), Section 6.2.3 on page 6-6 (IHI 0033A), Section 6.3.3 on page 6-66 (IHI0033B.b)	The master must never attempt a transfer where the width, as indicated by HSIZE, is wider than the data bus that it connects to.
ast_snps_ahb_haddr_aligned	MASTER	ERROR	Section 3.5 on page 3-9 (IHI 0011A), Section 3.5 on page 3-10 (IHI 0033A), Section 3.5 on page 3-34 (IHI0033B.b)	All transfers in a burst must be aligned to the address boundary equal to the size of the transfer. HADDR is not aligned to the address boundary.
ast_snps_ahb_haddr_seq_burst	MASTER	ERROR	Section 3.5 on page 3-9 (IHI 0011A), Section 3.2 on page 3-5 (IHI 0033A), Section 3.2 on page 3-30 (IHI0033B.b)	The remaining transfers in a burst are SEQUENTIAL and the address is related to the previous transfer. Wrong HADDR is seen during a burst.

Table 4-3: AHB AIP Properties

Property Name	Agent	Severity	Spec Reference	Property Description
ast_snps_ahb_haddr_incr_address	MASTER	ERROR	Section 3.5 on page 3-9 (IHI 0011A), Section 3.2 on page 3-5 (IHI 0033A), Section 3.2 on page 3-30 (IHI0033B.b)	During undefined length burst (HBURST==INCR), HADDR should not cross 1KB address boundary.
ast_snps_ahb_haddr_1kb_boundary	MASTER	ERROR	Section 3.6 on page 3-11 (IHI 0011A), Section 3.5 on page 3-9 (IHI 0033A), Section 3.5 on page 3-34 (IHI0033B.b)	Masters must not attempt to start an incrementing burst that crosses a 1KB address boundary. HADDR should not cross 1KB address boundary during a burst.
ast_snps_ahb_fixlen_burst_seq_or_busy	MASTER	ERROR	Section 3.6.1 on page 3-12 (IHI 0011A), Section 3.5.1 on page 3-10 (IHI 0033A), Section 3.5 on page 3-34 (IHI0033B.b)	Fixed burst length burst continues with SEQ or BUSY.
ast_snps_ahb_fixlen_burst_terminates_seq	MASTER	ERROR	Section 3.6.1 on page 3-12 (IHI 0011A), Section 3.5.1 on page 3-10 (IHI 0033A), Section 3.5.1 on page 3-35 (IHI0033B.b)	Fixed length burst type must terminate with a SEQ transfer.
ast_snps_ahb_single_followed_by_idle_or_nonseq	MASTER	ERROR	Section 3.6.1 on page 3-12 (IHI 0011A), Section 3.5.1 on page 3-10 (IHI 0033A), Section 3.5.1 on page 3-35 (IHI0033B.b)	SINGLE bursts must be followed by and IDLE transfer or a NONSEQ transfer.
ast_snps_ahb_htrans_reset_idle	MASTER	ERROR	Section 3.13 on page 3-40 (IHI 0011A), Section 7.1.2 on page 7-2 (IHI 0033A), Section 7.1.2 on page 7-2 (IHI0033B.b)	During reset, all masters must ensure that HTRANS indicates IDLE.
ast_snps_ahb_htrans_valid_level	MASTER	ERROR	Section 3.13 on page 3-40 (IHI 0011A), Section 7.1.2 on page 7-2 (IHI 0033A), Section 7.1.2 on page 7-68 (IHI0033B.b)	During reset, all masters must ensure the address and control signals are at valid levels. A value of X/Z on HTRANS is not permitted.
ast_snps_ahb_hburst_valid_level	MASTER	ERROR	Section 3.13 on page 3-40 (IHI 0011A), Section 7.1.2 on page 7-2 (IHI 0033A), Section 7.1.2 on page 7-68 (IHI0033B.b)	During reset, all masters must ensure the address and control signals are at valid levels. A value of X/Z on HBURST is not permitted.

Table 4-3: AHB AIP Properties

Property Name	Agent	Severity	Spec Reference	Property Description
ast_snps_ahb_hsize_valid_level	MASTER	ERROR	Section 3.13 on page 3-40 (IHI 0011A), Section 7.1.2 on page 7-2 (IHI 0033A), Section 7.1.2 on page 7-68 (IHI0033B.b)	During reset, all masters must ensure the address and control signals are at valid levels. A value of X/Z on HSIZE is not permitted.
ast_snps_ahb_hwrite_valid_level	MASTER	ERROR	Section 3.13 on page 3-40 (IHI 0011A), Section 7.1.2 on page 7-2 (IHI 0033A), Section 7.1.2 on page 7-68 (IHI0033B.b)	During reset, all masters must ensure the address and control signals are at valid levels. A value of X/Z on HWRITE is not permitted.
ast_snps_ahb_hprot_valid_level	MASTER	ERROR	Section 3.13 on page 3-40 (IHI 0011A), Section 7.1.2 on page 7-2 (IHI 0033A), Section 7.1.2 on page 7-68 (IHI0033B.b)	During reset, all masters must ensure the address and control signals are at valid levels. A value of X/Z on HPROT is not permitted.
ast_snps_ahb_haddr_valid_level	MASTER	ERROR	Section 3.13 on page 3-40 (IHI 0011A), Section 7.1.2 on page 7-2 (IHI 0033A), Section 7.1.2 on page 7-68 (IHI0033B.b)	During reset, all masters must ensure the address and control signals are at valid levels. A value of X/Z on HADDR is not permitted.
ast_snps_ahb_hwdata_valid_level	MASTER	ERROR	Section 3.10.1 on page 3-25 (IHI 0011A), Section 6.1.1 on page 6-2 (IHI 0033A), Section 7.1.2 on page 7-68 (IHI0033B.b)	The bus master must hold the data valid until the transfer completes. A value of X/Z on HWDATA is not permitted.
ast_snps_ahb_hmaster_wait_stable	MASTER	ERROR	Section 3.4 on page 3-7 (IHI 0011A), Section 3.6 on page 3-16 (IHI 0033A)	During a waited transfer, the master is restricted to change control signals. HMASTER signal should not be changed while HREADY is 0.
ast_snps_ahb_hmaster_burst_constant	MASTER	ERROR	Section 3.7 on page 3-17 (IHI 0011A), Section 3.2 on page 3-5 (IHI 0033A)	The control information is identical to the previous transfer during SEQ. HMASTER signal must remain constant throughout a burst transfer.
ast_snps_ahb_hmaster_valid_level	MASTER	ERROR	Section 3.13 on page 3-40 (IHI 0011A), Section 7.1.2 on page 7-2 (IHI 0033A), Section 7.1.2 on page 7-68 (IHI0033B.b)	During reset, all masters must ensure the address and control signals are at valid levels. A value of X/Z on HMASTER is not permitted.

Table 4-3: AHB AIP Properties

Property Name	Agent	Severity	Spec Reference	Property Description
ast_snps_ahb_hburst_after_ebt	MASTER	ERROR	Section 3.6.1 on page 3-12 (IHI 0011A)	If a bus master cannot complete a burst because it loses ownership of the bus, then it must rebuild the burst appropriately when it next gains access to the bus. HBURST signal is different from transfer before Early Burst Termination.
ast_snps_ahb_haddr_after_ebt	MASTER	ERROR	Section 3.6.1 on page 3-12 (IHI 0011A)	If a bus master cannot complete a burst because it loses ownership of the bus, then it must rebuild the burst appropriately when it next gains access to the bus. HADDR signal is different from transfer before Early Burst Termination.
ast_snps_ahb_hsize_after_ebt	MASTER	ERROR	Section 3.6.1 on page 3-12 (IHI 0011A)	If a bus master cannot complete a burst because it loses ownership of the bus, then it must rebuild the burst appropriately when it next gains access to the bus. HSIZE signal is different from transfer before Early Burst Termination.
ast_snps_ahb_hwrite_after_ebt	MASTER	ERROR	Section 3.6.1 on page 3-12 (IHI 0011A)	If a bus master cannot complete a burst because it loses ownership of the bus, then it must rebuild the burst appropriately when it next gains access to the bus. HWRITE signal is different from transfer before Early Burst Termination.
ast_snps_ahb_hprot_after_ebt	MASTER	ERROR	Section 3.6.1 on page 3-12 (IHI 0011A)	If a bus master cannot complete a burst because it loses ownership of the bus, then it must rebuild the burst appropriately when it next gains access to the bus. HPROT signal is different from transfer before Early Burst Termination.
ast_snps_ahb_hburst_split_retry	MASTER	ERROR	Section 3.9.5 on page 3-24 (IHI 0011A)	It should continue to request the bus and attempt the transfer until it has either completed successfully or been terminated with an RETRY or SPLIT response. HBURST signal is not as expected after RETRY or SPLIT.

Table 4-3: AHB AIP Properties

Property Name	Agent	Severity	Spec Reference	Property Description
ast_snps_ahb_haddr_split_retry	MASTER	ERROR	Section 3.9.5 on page 3-24 (IHI 0011A)	It should continue to request the bus and attempt the transfer until it has either completed successfully or been terminated with an RETRY or SPLIT response. HADDR signal is not as expected after RETRY or SPLIT.
ast_snps_ahb_hsize_split_retry	MASTER	ERROR	Section 3.9.5 on page 3-24 (IHI 0011A)	It should continue to request the bus and attempt the transfer until it has either completed successfully or been terminated with an RETRY or SPLIT response. HSIZE signal is not as expected after RETRY or SPLIT.
ast_snps_ahb_hwrite_split_retry	MASTER	ERROR	Section 3.9.5 on page 3-24 (IHI 0011A)	It should continue to request the bus and attempt the transfer until it has either completed successfully or been terminated with an RETRY or SPLIT response. HWRITE signal is not as expected after RETRY or SPLIT.
ast_snps_ahb_hprot_split_retry	MASTER	ERROR	Section 3.9.5 on page 3-24 (IHI 0011A)	It should continue to request the bus and attempt the transfer until it has either completed successfully or been terminated with an RETRY or SPLIT response. HPROT signal is not as expected after RETRY or SPLIT.
ast_snps_ahb_hunalign_wait_stable	MASTER	ERROR	Section 8.3.7 on page 8-17 (ARM DDI 0211K)	During a waited transfer, the master is restricted to change control signals. HUNALIGN signal should not be changed while HREADY is 0.
ast_snps_ahb_hbstrb_wait_stable	MASTER	ERROR	Section 8.3.7 on page 8-17 (ARM DDI 0211K)	HBSTRB has the same timing and validity requirements as Write data channel. HBSTRB signal should not be changed while HREADY is 0.
ast_snps_ahb_hbstrb_aligned	MASTER	ERROR	Section 8.3.7 Table 8-8 on page 8-18 (ARM DDI 0211K)	HBSTRB is aligned during data phase when HUNALIGN is 0 during address phase.
ast_snps_ahb_hbstrb_unaligned	MASTER	ERROR	Section 8.3.7 Table 8-8 on page 8-18 (ARM DDI 0211K)	HBSTRB is unaligned during data phase when HUNALIGN is 1 during address phase.

Table 4-3: AHB AIP Properties

Property Name	Agent	Severity	Spec Reference	Property Description
ast_snps_ahb_hbstrb_legal	MASTER	ERROR	Section 8.3.7 Table 8-8 on page 8-18 (ARM DDI 0211K)	Active bits in HBSTRB should be calculated based on HADDR and HSIZE.
ast_snps_ahb_hprot_valid_allocate	MASTER	ERROR	Section 8.3.5 on page 8-12 (ARM DDI 0211K)	When the transfer is Non-cacheable (HPROT[3] is LOW) then the Allocate bit is not used and must also be driven LOW by a master.
ast_snps_ahb_hprot_432_reserved	MASTER	ERROR	Section 8.3.5 Table 8-5 on page 8-12 (ARM DDI 0211K)	HPROT[4:2]==3'h4 and HPROT[4:2]==3'h5 are not defined and reserved.
ast_snps_ahb_hunalign_valid_level	MASTER	ERROR	Section 8.3.7 on page 8-16 (ARM DDI 0211K)	During reset, all masters must ensure the address and control signals are at valid levels. A value of X/Z on HUNALIGN is not permitted.
ast_snps_ahb_hbstrb_valid_level	MASTER	ERROR	Section 8.3.7 on page 8-16 (ARM DDI 0211K)	During reset, all masters must ensure the address and control signals are at valid levels. A value of X/Z on HBSTRB is not permitted.
ast_snps_ahb_hnonsec_wait_stable	MASTER	ERROR	Section 3.6 on page 3-39 (IHI0033B.b)	During a waited transfer, the master is restricted to change control signals. HNONSEC signal should not be changed while HREADY is 0.
ast_snps_ahb_hexcl_wait_stable	MASTER	ERROR	Section 3.6 on page 3-39 (IHI0033B.b)	During a waited transfer, the master is restricted to change control signals. HEXCL signal should not be changed while HREADY is 0.
ast_snps_ahb_hmaster_wait_stable	MASTER	ERROR	Section 3.6 on page 3-39 (IHI0033B.b)	During a waited transfer, the master is restricted to change control signals. HMASTER signal should not be changed while HREADY is 0.
ast_snps_ahb_hnonsec_burst_constant	MASTER	ERROR	Section 3.5 on page 3-34 (IHI0033B.b)	The control information is identical to the previous transfer during SEQ. HNONSEC signal must remain constant throughout a burst transfer.

Table 4-3: AHB AIP Properties

Property Name	Agent	Severity	Spec Reference	Property Description
ast_snps_ahb_hexcl_burst_constant	MASTER	ERROR	Section 3.5 on page 3-34 (IHI0033B.b)	The control information is identical to the previous transfer during SEQ. HEXCL signal must remain constant throughout a burst transfer.
ast_snps_ahb_hmaster_burst_constant	MASTER	ERROR	Section 3.5 on page 3-34 (IHI0033B.b)	The control information is identical to the previous transfer during SEQ. HMASTER signal must remain constant throughout a burst transfer.
ast_snps_ahb_hauser_wait_stable	MASTER	ERROR	Section 10.1 on page 10-80 (IHI0033B.b)	HAUSER has the same timing and validity requirements as Address channel. HAUSER signal should not be changed while HREADY is 0.
ast_snps_ahb_hwuser_wait_stable	MASTER	ERROR	Section 10.1 on page 10-80 (IHI0033B.b)	HWUSER has the same timing and validity requirements as Write data channel. HWUSER signal should not be changed while HREADY is 0.
ast_snps_ahb_hnonsec_valid_level	MASTER	ERROR	Section 7.1.2 on page 7-68 (IHI0033B.b)	During reset all masters must ensure the address and control signals are at valid levels. A value of X/Z on HNONSEC is not permitted.
ast_snps_ahb_hexcl_valid_level	MASTER	ERROR	Section 7.1.2 on page 7-68 (IHI0033B.b)	During reset, all masters must ensure the address and control signals are at valid levels. A value of X/Z on HEXCL is not permitted.
ast_snps_ahb_hmaster_valid_level	MASTER	ERROR	Section 7.1.2 on page 7-68 (IHI0033B.b)	During reset, all masters must ensure the address and control signals are at valid levels. A value of X/Z on HMASTER is not permitted.
ast_snps_ahb_excl_single_or_incr	MASTER	ERROR	Section 8.4 on page 8-73 (IHI0033B.b)	Exclusive transfer must have a single data transfer and must be included as burst type SINGLE or INCR.
ast_snps_ahb_excl_single_not_busy	MASTER	ERROR	Section 8.4 on page 8-73 (IHI0033B.b)	Exclusive transfer must not include a BUSY transfer.
ast_snps_ahb_excl_no_two_outstandings	MASTER	ERROR	Section 8.4 on page 8-73 (IHI0033B.b)	A master must not have two Exclusive Transfers outstanding at the same point in time.

Table 4-3: AHB AIP Properties

Property Name	Agent	Severity	Spec Reference	Property Description
ast_snps_ahb_excl_same_haddr	MASTER	ERROR	Section 8.4 on page 8-73 (IHI0033B.b)	For an Exclusive Read and an Exclusive Write to be considered part of the same Exclusive access sequence, and HADDR signals must be the same for both transfers.
ast_snps_ahb_excl_same_hsize	MASTER	ERROR	Section 8.4 on page 8-73 (IHI0033B.b)	For an Exclusive Read and an Exclusive Write to be considered part of the same Exclusive access sequence, and HSIZE signals must be the same for both transfers.
ast_snps_ahb_excl_same_hprot	MASTER	ERROR	Section 8.4 on page 8-73 (IHI0033B.b)	For an Exclusive Read and an Exclusive Write to be considered part of the same Exclusive access sequence, and HPROT signals must be the same for both transfers.
ast_snps_ahb_excl_same_hnonsec	MASTER	ERROR	Section 8.4 on page 8-73 (IHI0033B.b)	For an Exclusive Read and an Exclusive Write to be considered part of the same Exclusive access sequence, and HNONSEC signals must be the same for both transfers.
ast_snps_ahb_htrans_with_hgrant	MASTER	ERROR	3.11.2 Section on page 3-29 (IHI 0011A)	A bus master uses the HBUSREQx signal to request access to the bus.
ast_snps_ahb_htrans_nonseq_hbusreq	MASTER	ERROR	Section 3.11.2 on page 3-29 (IHI 0011A)	HBUSREQ should be asserted at the previous cycle when master initiates new burst transaction.
ast_snps_ahb_seq_incr_hbusreq	MASTER	ERROR	Section 3.11.2 on page 3-29 (IHI 0011A)	HBUSREQ should be asserted at the previous cycle when master issues INCR burst.
ast_snps_ahb_hlock_request_nonseq	MASTER	ERROR	Section 3.11.2 on page 3-29 (IHI 0011A)	If the master initiates locked accesses then it must also assert the HLOCKx signal along with HBUSREQ signal.
ast_snps_ahb_hlock_request_seq	MASTER	ERROR	Section 3.11.2 on page 3-29 (IHI 0011A)	If the master continues locked burst accesses with SEQ then it must also assert the HLOCKx signal.
ast_snps_ahb_htrans_retry_split_idle	MASTER	ERROR	Section 3.12.4 on page 3-39 (IHI 0011A)	The protocol requires that a master performs an IDLE transfer immediately after receiving a SPLIT or RETRY response.

Table 4-3: AHB AIP Properties

Property Name	Agent	Severity	Spec Reference	Property Description
ast_snps_ahb_hlock_retry_split	MASTER	ERROR	Section 3.9.5 on page 3-24 (IHI 0011A)	Master should issue consistent hlock with original transfer after RETRY or SPLIT.
ast_snps_ahb_hbusreq_valid_level	MASTER	ERROR	Section 3.1 on page 3-4 (IHI 0011A)	During reset, bus request signal should be at valid level. A value of X/Z on HBUSREQ is not permitted.
ast_snps_ahb_hgrant_valid_level	ARBITER	ERROR	Section 3.1 on page 3-4 (IHI 0011A)	During reset, bus grant signal should be at valid level. A value of X/Z on HGRANT is not permitted.
ast_snps_ahb_hlock_valid_level	MASTER	ERROR	Section 3.1 on page 3-4 (IHI 0011A)	During reset, bus lock signal should be at valid level. A value of X/Z on HLOCK is not permitted.
ast_snps_ahb_htrans_idle_after_locked	MASTER	RECOMMEND	Section 3.11.5 on page 3-34 (IHI 0011A), Section 3.3 on page 3-7 (IHI 0033A), Section 3.3 on page 3-32 (IHI0032B.b)	After a locked transfer, it is recommended that the master inserts an IDLE transfer.
ast_snps_ahb_haddr_locked_1kb_boundary	MASTER	RECOMMEND	AMBA Design Kit Reference Manual	Master should ensure that it does not perform a locked sequence of transfers over a 1KB boundary.
ast_snps_ahb_htrans_error_idle	MASTER	RECOMMEND	Section 3.5.2 on page 3-10 (IHI 0033A), Section 3.5.2 on page 3-35 (IHI0033B.b)	If a slave provides an ERROR response then the master can cancel the remaining transfers in the burst. However this is not a strict requirement and it is acceptable for the master to continue the remaining transfers in the burst.
ast_snps_ahb_hreadyout_write_min_waits	SLAVE	WARNING		Slave returned HREADYOUT for write transaction before predetermined minimum number of wait states.
ast_snps_ahb_hreadyout_read_min_waits	SLAVE	WARNING		Slave returned HREADYOUT for read transaction before predetermined minimum number of wait states.
ast_snps_ahb_htrans_busy_max	MASTER	WARNING		Master should not continue HTRANS==BUSY more than BUSY_MAX consecutive cycles.

Table 4-3: AHB AIP Properties

Property Name	Agent	Severity	Spec Reference	Property Description
ast_snps_ahb_hburst_incr_max	MASTER	WARNING		Master should not continue unspecified length INCR burst more than INCR_MAX length.
ast_snps_ahb_hwddata_error_wait_stable	MASTER	WARNING	Section 5.1 (IHI0033B.b), ARM FAQ	It is recommended, but not required, that a master keeps HWDATA stable for all cycles of a write transfer that receives an ERROR response.
ast_snps_ahb_excl_exwr_after_exrd	MASTER	WARNING	Section 8.4 on page 8-73 (IHI0033B.b)	Exclusive Write occurs without Exclusive Read.
ast_snps_ahb_hsel_sel	DECODER	ERROR	Section 3.8 on page 3-19 (IHI 0011A)	A Slave should be selected by decoding address, wrong Slave is selected in HSELx.
ast_snps_ahb_hsel_nosel	DECODER	ERROR	Section 3.8 on page 3-19 (IHI 0011A)	HSELx should not be asserted when address indicates non existent locations.
ast_snps_ahb_hready_sel	DECODER	ERROR	Section 3.18 on page 3-45 (IHI 0011A)	HREADY signal should be HREADYOUT signal from Slave selected by HSELx, wrong HREADYOUT is driven.
ast_snps_ahb_hrdata_sel	DECODER	ERROR	Section 3.18 on page 3-45 (IHI 0011A)	HRDATA signal should be HRDATA signal from Slave selected by HSELx, wrong HRDATA is driven.
ast_snps_ahb_hresp_sel	DECODER	ERROR	Section 3.18 on page 3-45 (IHI 0011A)	HRESP signal should be HRESP signal from Slave selected by HSELx, wrong HRESP is driven.
ast_snps_ahb_hexokay_sel	DECODER	ERROR	Section 2.5 on page 2-25 (IHI 0033B.b)	HEXOKAY should be HEXOKAY signal from Slave selected by HSELx, wrong HEXOKAY is driven.
ast_snps_ahb_hexokay_def	DECODER	ERROR	Section 2.5 on page 2-25 (IHI 0033B.b)	Default value of HEXOKAY should be 0.
ast_snps_ahb_hruser_sel	DECODER	ERROR	Section 2.5 on page 2-25 (IHI 0033B.b)	HRUSER signal should be HRUSER signal from Slave selected by HSELx, wrong HRUSER is driven.
ast_snps_ahb_hsel_onehot	DECODER	ERROR	Section 3.8 on page 3-19 (IHI 0011A)	A central address decoder is used to provide a select signal, HSELx, for each slave on the bus. Multiple slave selection is not allowed.

Table 4-3: AHB AIP Properties

Property Name	Agent	Severity	Spec Reference	Property Description
ast_snps_ahb_hsel_onehot0	DECODER	ERROR	Section 3.8 on page 3-19 (IHI 0011A)	A central address decoder is used to provide a select signal, HSELx, for each slave on the bus. Multiple slave selection is not allowed.
ast_snps_ahb_def_slave_idle	DECODER	ERROR	Section 3.8 on page 3-19 (IHI 0011A)	IDLE or BUSY transfers to non-existent locations should result in a zero wait state OKAY response.
ast_snps_ahb_def_slave_error_1st	DECODER	ERROR	Section 3.8 on page 3-19 (IHI 0011A)	If a NONSEQUENTIAL or SEQUENTIAL transfer is attempted to a nonexistent address location then the default slave should provide an ERROR response.
ast_snps_ahb_def_slave_error_2nd	DECODER	ERROR	Section 3.8 on page 3-19 (IHI 0011A)	If a NONSEQUENTIAL or SEQUENTIAL transfer is attempted to a nonexistent address location then the default slave should provide an ERROR response.
ast_snps_ahb_hgrant_not_change	ARBITER	ERROR	Section 3.11.4 on page 3-33	Normally the arbiter will not hand over the bus to a new master until the end of a burst of transfers.
ast_snps_ahb_hgrant_only_one_master	ARBITER	ERROR	Section 3.11.1 on page 3-28	Arbiter must give grant to only one Master.
ast_snps_ahb_hgrant_default_master	ARBITER	ERROR	Section 3.11.2 on page 3-29	When no masters are requesting the bus and the arbiter grants access to a default master.
ast_snps_ahb_hgrant_requested_master	ARBITER	ERROR	Section 3.11.3 on page 3-30	Arbiter must give grant only to the master requesting the bus.
ast_snps_ahb_hgrant_split_retry	ARBITER	ERROR	Section 3.11.3 on page 3-30	Arbiter negates grant for the master gets RETRY or SPLIT response.
ast_snps_ahb_hgrant_active	ARBITER	ERROR	Section 3.11.3 on page 3-30	Arbiter gives grant when there's any master requesting the bus.
ast_snps_ahb_hmaster_granted	ARBITER	ERROR	Section 3.11.3 on page 3-30	Arbiter indicates master currently granted.
ast_snps_ahb_hmastlock_allow	ARBITER	ERROR	Section 3.11.5 on page 3-34	Arbiter indicates locked transfer when master requesting locked transfer.
ast_snps_ahb_wait_split_not_granted	ARBITER	ERROR	Section 3.12.1 on page 3-36	Master should not get grant while waiting SPLIT.

Table 4-3: AHB AIP Properties

Property Name	Agent	Severity	Spec Reference	Property Description
ast_snps_ahb_hbusreqx_not_unknown	ARBITER	ERROR	Section 3.1 on page 3-4 (IHI 0011A)	During reset, bus request signal should be at valid level. A value of X/Z on HBUSREQ is not permitted.
ast_snps_ahb_hlockx_not_unknown	ARBITER	ERROR	Section 3.1 on page 3-4 (IHI 0011A)	During reset, bus lock signal should be at valid level. A value of X/Z on HLOCK is not permitted.

4.4 The AHB AIP Cover Properties

Table 4-4 lists the AHB AIP cover properties.

Table 4-4: AHB AIP Cover Properties

Property Name	Description	Note
cov_snps_ahb_htrans_idle	Seen burst with HTRANS = IDLE	
cov_snps_ahb_htrans_busy	Seen burst with HTRANS = BUSY	
cov_snps_ahb_htrans_nonseq	Seen burst with HTRANS = NONSEQ	
cov_snps_ahb_htrans_seq	Seen burst with HTRANS = SEQ	
cov_snps_ahb_hburst_single	Seen burst with HBURST = SINGLE	
cov_snps_ahb_hburst_incr	Seen burst with HBURST = INCR	
cov_snps_ahb_hburst_incr4	Seen burst with HBURST = INCR4	
cov_snps_ahb_hburst_incr8	Seen burst with HBURST = INCR8	
cov_snps_ahb_hburst_incr16	Seen burst with HBURST = INCR16	
cov_snps_ahb_hburst_wrap4	Seen burst with HBURST = WRAP4	
cov_snps_ahb_hburst_wrap8	Seen burst with HBURST = WRAP8	
cov_snps_ahb_hburst_wrap16	Seen burst with HBURST = WRAP16	
cov_snps_ahb_hsize_byte	Seen burst with HSIZE = Byte (8 bits)	
cov_snps_ahb_hsize_halfword	Seen burst with HSIZE = Halfword (16 bits)	
cov_snps_ahb_hsize_word	Seen burst with HSIZE = Word (32 bits)	
cov_snps_ahb_hsize_doubleword	Seen burst with HSIZE = Double word (64 bits)	
cov_snps_ahb_hsize_4word	Seen burst with HSIZE = 4 Words (128 bits)	
cov_snps_ahb_hsize_8word	Seen burst with HSIZE = 8 Words (256 bits)	
cov_snps_ahb_hsize_16word	Seen burst with HSIZE = 16 Words (512 bits)	

Table 4-4: AHB AIP Cover Properties

Property Name	Description	Note
cov_snps_ahb_hsize_32word	Seen burst with HSIZE = 32 Words (1024 bits)	
cov_snps_ahb_hprot_opcode_fetch	Seen burst with HPROT = Opcode fetch	
cov_snps_ahb_hprot_data_access	Seen burst with HPROT = Data access	
cov_snps_ahb_hprot_user_access	Seen burst with HPROT = User access	
cov_snps_ahb_hprot_privileged_access	Seen burst with HPROT = Privileged access	
cov_snps_ahb_hprot_non_bufferable	Seen burst with HPROT = Not bufferable	
cov_snps_ahb_hprot_bufferable	Seen burst with HPROT = Bufferable	
cov_snps_ahb_hprot_non_cacheable	Seen burst with HPROT = Not cacheable	
cov_snps_ahb_hprot_cacheable	Seen burst with HPROT = Cacheable	
cov_snps_ahb_hmastlock_normal	Seen burst with HMASTLOCK = Normal	
cov_snps_ahb_hmastlock_locked	Seen burst with HMASTLOCK = Locked	
cov_snps_ahb_hresp_okay	Seen response with HRESP = OKAY	
cov_snps_ahb_hresp_error_1st	Seen response with HRESP = ERROR at the first cycle	
cov_snps_ahb_hresp_error_2nd	Seen response with HRESP = ERROR at the second cycle	
cov_snps_ahb_hresp_retry_1st	Seen response with HRESP = RETRY at the first cycle	AHB Full only
cov_snps_ahb_hresp_retry_2nd	Seen response with HRESP = RETRY at the second cycle	AHB Full only
cov_snps_ahb_request_after_retry	seen re-request after RETRY response	AHB Full only
cov_snps_ahb_hresp_split_1st	Seen response with HRESP = SPLIT at the first cycle	AHB Full only
cov_snps_ahb_hresp_split_2nd	Seen response with HRESP = SPLIT at the second cycle	AHB Full only
cov_snps_ahb_request_after_split	seen re-request after SPLIT response	AHB Full only
cov_snps_ahb_ebt_occur	seen Early Burst Termination	AHB Full only
cov_snps_ahb_master_lock_grant	Seen Master requested bus with Locked and granted	AHB Full only
cov_snps_ahb_master_nolock_grant	Seen Master requested bus with Normal and granted	AHB Full only

Table 4-4: AHB AIP Cover Properties

Property Name	Description	Note
cov_snps_ahb_hmaster_nonseq	Indicate which Master issues transaction with HTRANS = NONSEQ	AHB Full SLAVE only
cov_snps_ahb_hlock_normal	Seen Master issued Normal request with HTRANS = NONSEQ	AHB Full MASTER only
cov_snps_ahb_hlock_locked	Seen Master issued Locked request with HTRANS = NONSEQ	AHB Full MASTER only
cov_snps_ahb_hexcl_non_exclusive	Seen burst with HEXCL = Non Exclusive	AHB5 only
cov_snps_ahb_hexcl_exclusive	Seen burst with HEXCL = Exclusive	AHB5 only
cov_snps_ahb_hexokay_exfail_for_read	Seen response with HEXOKAY = Failure of Exclusive Read	AHB5 only
cov_snps_ahb_hexokay_exokay_for_read	Seen response with HEXOKAY = Success of Exclusive Read	AHB5 only
cov_snps_ahb_hexokay_exfail_for_write	Seen response with HEXOKAY = Failure of Exclusive Write	AHB5 only
cov_snps_ahb_hexokay_exokay_for_write	Seen response with HEXOKAY = Success of Exclusive Write	AHB5 only
cov_snps_ahb_hmaster_non_exclusive	Indicate which Master issues transaction with HEXCL = Non Exclusive	AHB5 only
cov_snps_ahb_hmaster_exclusive	Indicate which Master issues transaction with HEXCL = Exclusive	AHB5 only
cov_snps_ahb_hprot_allocate_0	Seen burst with HPROT indicates Allocated in the cache	ARMv6 only
cov_snps_ahb_hprot_allocate_1	Seen burst with HPROT indicates Not allocated in the cache	ARMv6 only
cov_snps_ahb_hprot_strongly_ordered	Seen burst with HPROT indicates Strongly Ordered	ARMv6 only
cov_snps_ahb_hprot_device	Seen burst with HPROT indicates Device access	ARMv6 only
cov_snps_ahb_hprot_cacheable_noallocate	Seen burst with HPROT indicates Cacheable and No Allocate	ARMv6 only
cov_snps_ahb_hprot_cacheable_writethrough_allocate_readonly	Seen burst with HPROT indicates Cacheable, write-through and Allocate read only	ARMv6 only
cov_snps_ahb_hprot_cacheable_writeback_allocate_readwrite	Seen burst with HPROT indicates Cacheable, write-back and Allocate read/write	ARMv6 only

Table 4-4: AHB AIP Cover Properties

Property Name	Description	Note
cov_snps_ahb_hprot_cacheable_writeback_allocate_readonly	Seen burst with HPROT indicates Cacheable, write-back and Allocate read only	ARMv6 only
cov_snps_ahb_granted_master	Indicate which Master issues transaction with HTRANS = NONSEQ	ARBITER only
cov_snps_ahb_master_locked	Indicate which Master issues transaction with HMASTLOCK = Locked	ARBITER only
cov_snps_ahb_master_normal	Indicate which Master issues transaction with HMASTLOCK = Normal	ARBITER only
cov_snps_ahb_hbusreqx_asserted	Indicate which Master asserts HBUSREQ signal	ARBITER only
cov_snps_ahb_hbusreqx_deasserted	Indicate which Master de-asserts HBUSREQ signal	ARBITER only
cov_snps_ahb_hgrantx_asserted	Indicate which Master gets granted	ARBITER only
cov_snps_ahb_hgrantx_deasserted	Indicate which Master loses granted	ARBITER only
cov_snps_ahb_hsplix_asserted	Indicate which Master gets HSPLIT request	ARBITER only

4.5 Description of Properties

Properties are grouped on the basis of categories, which depends on the configuration parameter values. Categories can be such as `x_check` properties, `error_idle` properties, `lockidle_recommend` properties, and `max_waits` properties. Each parameter has a default value (see [Table 4-1](#) for the same).

- To instantiate `x_check` properties:
Set `CONFIG_X_CHECK=1`
- To check for Write errors in IDLE state:
Set `ERROR_IDLE=1`
- To enable/disable MAXIMUM LATENCY of expected behavior (WAIT TIME for control/data signals) for both AHB Lite and AHB Full:
Set `CHECK_MAX_WAITS=1`

Similarly, to enable all assert or assume properties, the `ENABLE_ASSERT` and `ENABLE_ASSUME` parameters must be set to 1. See [Table 4-3](#) for details on each property.

4.6 Behavior of Properties

This Section describes the behavior of all properties of AHB Lite and AHB Full when acting as assert, assume, or cover.

4.6.1 Properties In Assert Directives

Assertion 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 4-3](#) 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 4-3](#) are declared as assume, and checkers mentioned as 'Master' in Master/Slave column are declared as assert.

4.6.2 Properties In Assume Directives

Assume properties have the following features:

- Assume properties act as a constraint for generating controlled stimulus as per a protocol because the formal tool treats the inputs as free variables.
- When AGENT_TYPE is SLAVE, the checkers mentioned as 'Master' in Master/Slave checkers column of [Table 4-3](#) are declared as assume, and checkers mentioned as 'Slave' in Master/Slave column are declared as assert.
- When AGENT_TYPE is SLAVE, the checkers mentioned as 'Slave' in Master/Slave column of [Table 4-3](#) are declared as assume, and checkers mentioned as 'Master' in Master/Slave column are declared as assert.

4.6.3 Properties In Cover Directives

Cover properties have the following features:

- Cover properties specify the number of assertions executed or covered when the stimulus is generated. This number reflects the AHB AIP coverage (that is, number of properties actually verified in verification run or the number of properties, checks, or constraints that got triggered). Also, cover properties indicate what kind of transactions or scenarios are exercised during proof. Therefore, cover properties are useful in checking if there are no over-constraints in the environment, and if a design issues all possible transactions.
- To enable cover properties, set `ENABLE_COVER=1`.



The AHB AIP Use Cases

This chapter discusses the AHB Lite and AHB Full Assertion IP in different environments used for validation.

5.1 The AHB AIP Examples

This section describes the setup of the AHB AIP (both for Lite and Full modes) where the AHB AIP is connected with RTL through a bind file. The bind file example is shown in the [“The AHB AIP with Slave DUT for AHB Lite”](#) and [“AHB AIP with Master DUT for AHB Lite”](#) topics, where both Master and Slave DUT signals are connected to AIP master and slave signals in case of AHB Lite and AHB Full. The arbiter signals of the DUT are connected to the AHB AIP arbiter in case of AHB Full.



Note

If a signal corresponding to the AHB 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 `hprot` signal, set with `4'h0`.

5.1.1 The AHB AIP with Slave DUT for AHB Lite

In this setup, the AHB Lite AIP is connected with the AHB Lite Slave DUT.

1. For connecting the ports of the AHB Lite Slave DUT with the AHB AIP, create a bind file to include the instance of the AHB AIP into the top level module of the DUT.
2. In this setup, both the slave AHB AIP and slave DUT are connected with the master AHB AIP, such that the decoder and multiplexer selects either of the slave AHB AIP or DUT.
3. [Figure 5-1](#) shows the AHB Lite AIP master connected to the slave DUT. Here, two slaves are connected, one is the slave DUT and other is the slave AHB AIP. You can replace the slave DUT with the slave AHB AIP or any other slave DUT as per the requirement.

Figure 5-1 Slave DUT: 1 Master 2 Slave (1 DUT and 1 AHB AIP)

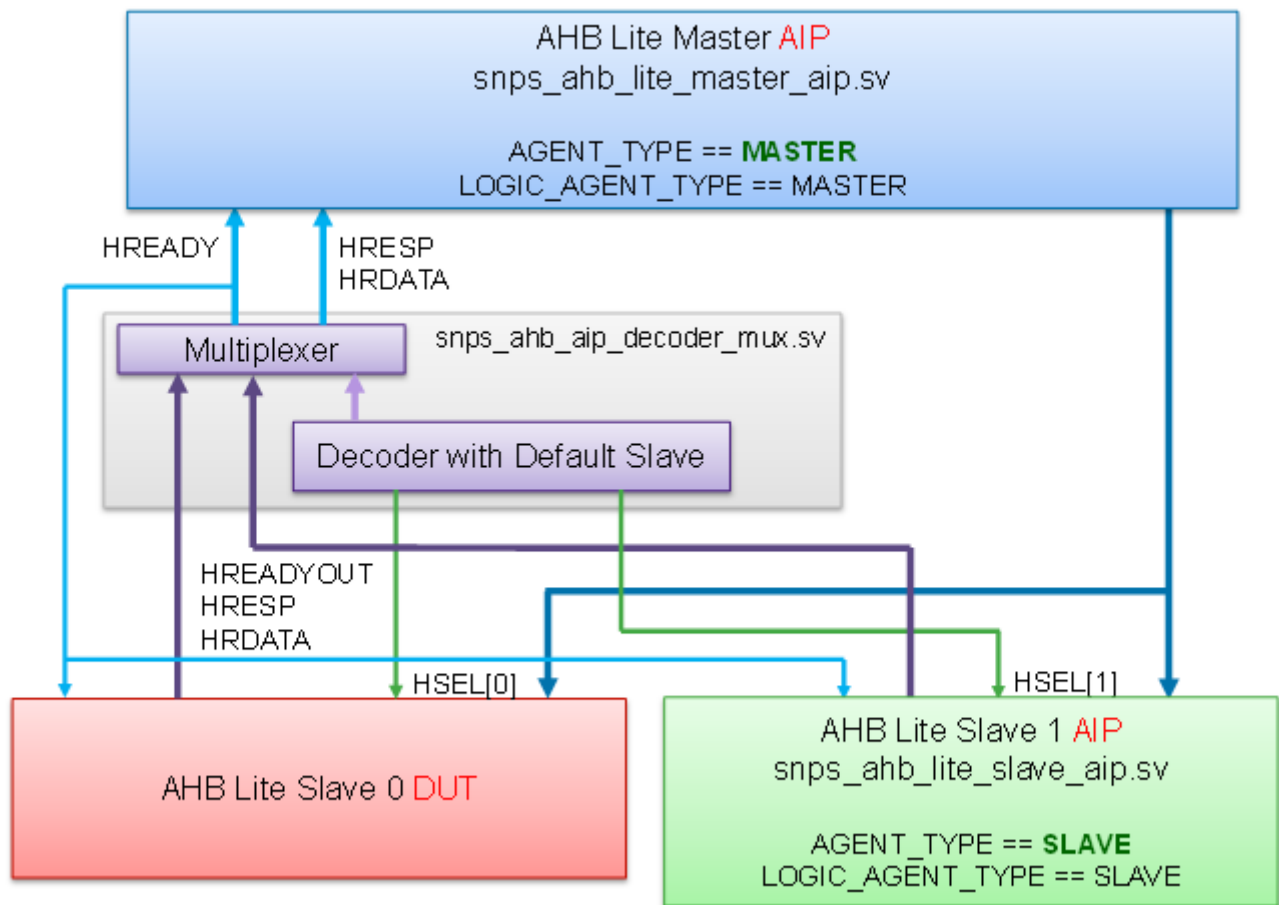


Table 5-1 Bind Example: Slave DUT Connected to the AHB AIP

<pre> module DW_ahb (hclk, hresetn, haddr_m1, hburst_m1, hlock_m1, hprot_m1, hsize_m1, htrans_m1, hwdma_m1, hwrite_m1, hsel_s1, hready_resp_s1, hresp_s1, hrdata_s1, hsel_s2, hready_resp_s2, hresp_s2, hrdata_s2, hsel_s3, hready_resp_s3, hresp_s3, hrdata_s3, haddr, hburst, hprot, hsize, htrans, hwdma, hwrite, hready, hresp, hrdata, hmaster, hmaster_data, hmastlock); DW_ahb_dcdcr #(`HADDR_WIDTH, INT_R1_N_SA_1, INT_R1_N_EA_1, INT_R1_N_SA_2, INT_R1_N_EA_2) U_dcdcr (.haddr(haddr), .hsel(hsel)); endmodule </pre>	<pre> bind DW_ahb snps_ahb_lite_master_aip #(.ENABLE_ASSERT (1), .ENABLE_ASSUME (1), .ENABLE_COVER (1), .CONFIG_X_CHECK (1), .LOCKIDLE_RCMND (0), .CHECK_FORMAL (1), .ADDR_WIDTH (`HADDR_WIDTH), .DATA_WIDTH (`AHB_DATA_WIDTH), .ERROR_IDLE (1), .CHECK_MAX_WAITS (0), .MAX_WAITS (16), .ADDR_RANGE (2), .MIN_ADDR ({32'h2000000, 32'ha000000}), .MAX_ADDR ({32'h200ffff, 32'ha00ffff})) u_ahb_lite_mst_aip (.hclk (hclk), .hresetn (hresetn), .htrans (htrans_m1), .haddr (haddr_m1), .hwrite (hwrite_m1), .hsize (hsize_m1), .hburst (hburst_m1), .hprot (hprot_m1), .hmastlock (hmastlock), .hwdma (hwdma), .hready (hready), .hresp (hresp[0]), .hrdata (hrdata), .*); bind DW_ahb snps_ahb_lite_slave_aip #(.ENABLE_ASSERT (1), .ENABLE_ASSUME (1), .ENABLE_COVER (1), .CONFIG_X_CHECK (1), .LOCKIDLE_RCMND (0), .CHECK_FORMAL (1), .ADDR_WIDTH (`HADDR_WIDTH), .DATA_WIDTH (`AHB_DATA_WIDTH), .ERROR_IDLE (1), .CHECK_MAX_WAITS (0), .MAX_WAITS (16)) u_ahb_lite_slv_aip_0 (.hclk (hclk), .hresetn (hresetn), .hsel (hsel_s1), .htrans (htrans), .haddr (haddr), .hwrite (hwrite), .hsize (hsize), .hburst (hburst), .hprot (hprot), .hmastlock (hmastlock), .hwdma (hwdma), .hready (hready), .hreadyout (hready_resp_s1), .hresp (hresp_s1[0]), .hrdata (hrdata_s1), .*); </pre>
--	---

5.1.2 AHB AIP with Master DUT for AHB Lite

In this setup, the AHB Lite AIP is connected with the AHB Lite Master DUT.

1. For connecting the ports of the AHB Lite Master DUT with the AHB AIP, create a bind file to include the instance of the AIP into the top level module of the DUT.
2. In this setup, both slave AHB AIPs are connected with the master DUT.

Figure 5-2 DUT is Master: 1 Master 2 Slave (Both AHB AIP)

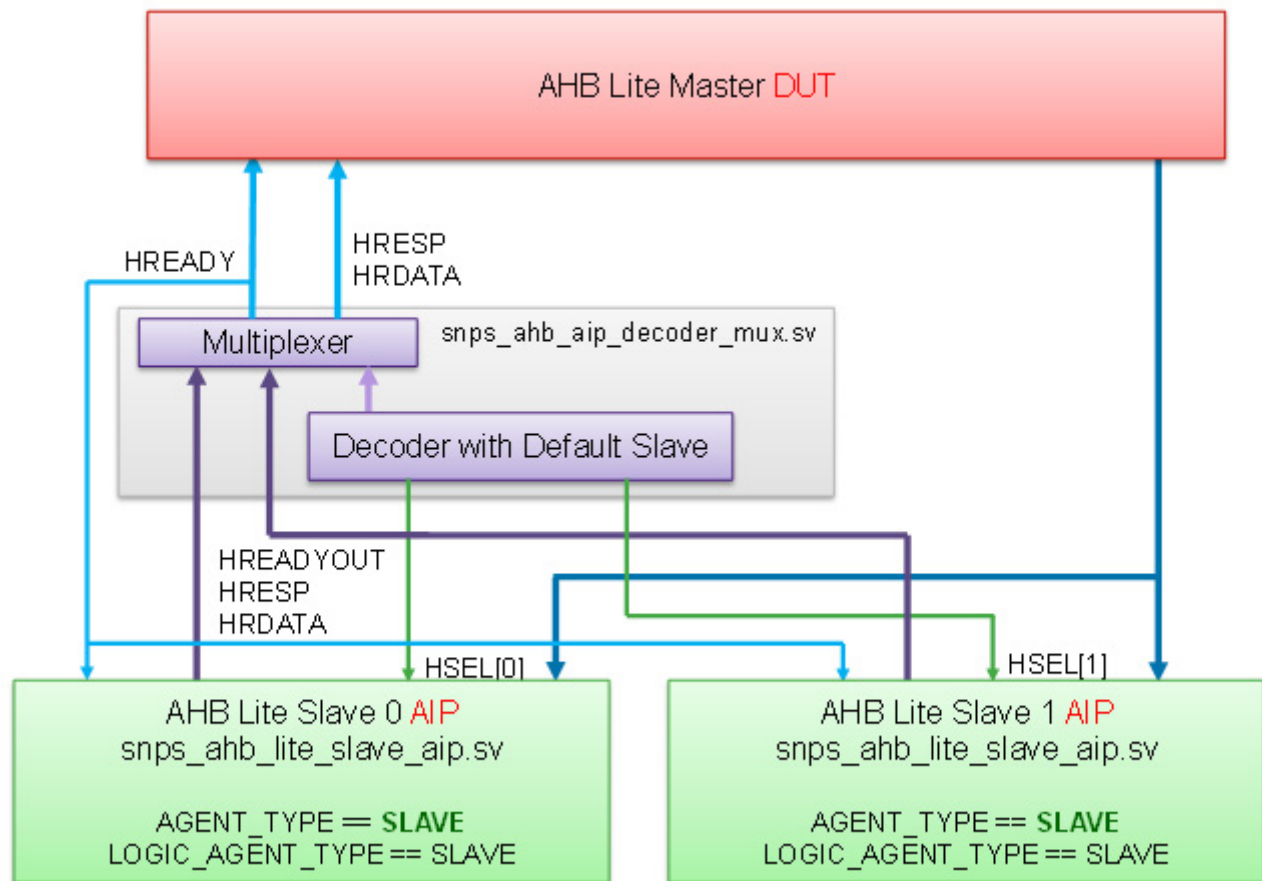


Table 5-2 Bind Example: Master DUT with AHB AIP

```

module DW_ahb (
    hclk,
    hresetn,
    haddr_m1,
    hburst_m1,
    hlock_m1,
    hprot_m1,
    hsize_m1,
    htrans_m1,
    hwdma_m1,
    hwrite_m1,
    hsel_s1,
    hready_resp_s1,
    hresp_s1,
    hrdata_s1,
    hsel_s2,
    hready_resp_s2,
    hresp_s2,
    hrdata_s2,
    hsel_s3,
    hready_resp_s3,
    hresp_s3,
    hrdata_s3,
    haddr,
    hburst,
    hprot,
    hsize,
    htrans,
    hwdma,
    hwrite,
    hready,
    hresp,
    hrdata,
    hmaster,
    hmaster_data,
    hmastlock
);

DW_ahb_dcdcr
#(`HADDR_WIDTH,
  INT_R1_N_SA_1,
  INT_R1_N_EA_1,
  INT_R1_N_SA_2,
  INT_R1_N_EA_2)
U_dcdcr (
    .haddr(haddr),
    .hsel(hsel) );

DW_ahb_arblite
U_arblite (
    //leda NTL_CON10B on
    //leda NTL_CON10 on
    .hclk(hclk),
    .hresetn(hresetn),
    .hlock_m1(hlock_m1),
    .hready(hready),
    .hmaster(hmaster),
    .hmastlock(hmastlock)
);

bind DW_ahb snps_ahb_lite_master_aip
#(.ENABLE_ASSERT (1),
  .ENABLE_ASSUME (1),
  .ENABLE_COVER (1),
  .CONFIG_X_CHECK (1),
  .LOCKIDLE_RCMND (0),
  .CHECK_FORMAL (1),
  .ADDR_WIDTH (`HADDR_WIDTH),
  .DATA_WIDTH (`AHB_DATA_WIDTH),
  .ERROR_IDLE (1),
  .ADDR_RANGE (2),
  .MIN_ADDR ({32'h2000000,32'ha000000}),
  .MAX_ADDR ({32'h200ffff,32'ha00ffff}))

u_ahb_lite_mst_aip
(.hclk(hclk),
 .hresetn(hresetn),
 .htrans(htrans_m1),
 .haddr(haddr_m1),
 .hwrite(hwrite_m1),
 .hsize(hsize_m1),
 .hburst(hburst_m1),
 .hprot(hprot_m1),
 .hmastlock(hmastlock),
 .hwdma(hwdma),
 .hready(hready),
 .hresp(hresp[0]),
 .hrdata(hrdata),
.* );

bind DW_ahb snps_ahb_lite_slave_aip
#(.ENABLE_ASSERT (1),
  .ENABLE_ASSUME (1),
  .ENABLE_COVER (1),
  .CONFIG_X_CHECK (1),
  .LOCKIDLE_RCMND (0),
  .CHECK_FORMAL (1),
  .ADDR_WIDTH (`HADDR_WIDTH),
  .DATA_WIDTH (`AHB_DATA_WIDTH),
  .ERROR_IDLE (1),
  .CHECK_MAX_WAITS (0),
  .MAX_WAITS (16))

u_ahb_lite_slv_aip_0
(.hclk(hclk),
 .hresetn(hresetn),
 .hsel(hsel_s1),
 .htrans(htrans),
 .haddr(haddr),
 .hwrite(hwrite),
 .hsize(hsize),
 .hburst(hburst),
 .hprot(hprot),
 .hmastlock(hmastlock),
 .hwdma(hwdma),
 .hready(hready),
 .hreadyout(hready_resp_s1),
 .hresp(hresp_s1[0]),
 .hrdata(hrdata_s1),
.* );

bind DW_ahb snps_ahb_lite_slave_aip
#(.ENABLE_ASSERT (1),
  .ENABLE_ASSUME (1),
  .ENABLE_COVER (1),
  .CONFIG_X_CHECK (1),
  .LOCKIDLE_RCMND (0),
  .CHECK_FORMAL (1),
  .ADDR_WIDTH (`HADDR_WIDTH),
  .DATA_WIDTH (`AHB_DATA_WIDTH),
  .ERROR_IDLE (1),
  .CHECK_MAX_WAITS (0),
  .MAX_WAITS (16))

u_ahb_lite_slv_aip_1
(.hclk(hclk),
 .hresetn(hresetn),
 .hsel(hsel_s2),
 .htrans(htrans),
 .haddr(haddr),
 .hwrite(hwrite),
 .hsize(hsize),
 .hburst(hburst),
 .hprot(hprot),
 .hmastlock(hmastlock),
 .hwdma(hwdma),
 .hready(hready),
 .hreadyout(hready_resp_s2),
 .hresp(hresp_s2[0]),
 .hrdata(hrdata_s2),
.* );

```

