

AXI Stream Assertion IP User Guide

Version 2020.12-SP1, March 2021





Copyright Notice and Proprietary Information

© 2021 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 <http://www.synopsys.com/Company/Pages/Trademarks.aspx>.

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	12
Chapter 2	
Installation and Setup	13
2.1 Verifying Hardware Requirements	13
2.2 Verifying Software Requirements	13
2.2.1 Platform/OS and Simulator Software	13
2.2.2 Synopsys Common Licensing (SCL) Software	13
2.2.3 Other Third Party Software	14
2.3 Preparing for Installation	15
Chapter 3	
The AXI STREAM AIP in a Formal Verification Environment	17
3.1 Introduction to the VC Formal Tool	17
3.2 The AXI STREAM AIP in a Formal Verification Environment	18
3.2.1 Instantiating the AXI STREAM AIP Using the bind Statement	18
3.2.2 Creating a Tcl File	18
3.2.3 Reading and Running a Tcl File	19
3.2.4 Commonly Used AXI STREAM AIP Configurations	22
3.2.5 The AXI STREAM AIP As a Master	23
3.2.6 The AXI STREAM AIP As a Slave	23
3.2.7 The AXI STREAM AIP As a Monitor	24
3.2.8 The AXI STREAM AIP In Constraint Mode	24
3.3 Clock and Reset Functionality	24

Chapter 4	27
The AXI STREAM AIP Configuration	27
4.1 The AXI STREAM AIP Configuration Parameters	27
4.2 The AXI STREAM AIP Interface Ports	29
4.3 The AXI STREAM AIP Properties	30
4.3.1 Properties (Assertions/ Assumptions)	30
4.3.2 The AXI STREAM AIP Cover Properties	32
4.4 Behavior of Properties	32
4.4.1 Properties In Assert Directives	32
4.4.2 Properties As Assume Directives	32
4.4.3 Properties In Cover Directives	33
Chapter 5	35
The AXI STREAM AIP Use Cases	35
5.1 The AXI STREAM AIP Examples	35
5.1.1 The AXI STREAM Master AIP With Slave DUT	35
5.1.2 The AXI STREAM Slave AIP With a Master DUT	37

Figures

Figure 3-1:	Proven and Falsified Properties	20
Figure 3-2:	Options to Debug Properties	21
Figure 3-3:	Signal Behavior in Waveforms	21
Figure 3-4:	The AXI STREAM AIP As a Master	23
Figure 3-5:	The AXI STREAM AIP As a Slave	24
Figure 5-1:	Slave DUT with AXI STREAM Master AIP	36
Figure 5-2:	AXI STREAM Master AIP – Slave DUT bind example	36
Figure 5-3:	Master DUT with AXI STREAM Slave AIP	37
Figure 5-4:	AXI STREAM Slave AIP – Master DUT bind example	38



Tables

Table 2-1:	AIP Licensing Key Features	14
Table 3-1:	Tcl File Example	19
Table 3-2:	Common Usage Models	22
Table 4-1:	The AXI STREAM AIP Configuration Parameters	27
Table 4-2:	The AXI STREAM AIP Interface Ports	29
Table 4-3:	AXI STREAM AIP Properties	30
Table 4-4:	The AXI STREAM AIP Cover Properties	32





Preface

This guide contains the installation, setup, and usage instructions for the AMBA AXI STREAM SystemVerilog Assertion IP(AIP), and is meant for design or verification engineers who want to verify RTL designs with an AMBA AXI STREAM interface.

This chapter includes the following sections:

- [Guide Organization](#)
- [Web Resources](#)
- [Customer Support](#)

Guide Organization

The chapters of this guide are organized as follows:

Chapter 1, “[Introduction](#)”, introduces the Synopsys AXI STREAM AIP and its features.

Chapter 2, “[Installation and Setup](#)”, describes system requirements and provides instructions on how to install, configure, and begin using the AXI STREAM AIP.

Chapter 3, “[The AXI STREAM AIP in a Formal Verification Environment](#)”, introduces the setup and usage of the AXI STREAM AIP with the “VC Formal” tool.

Chapter 4, “[The AXI STREAM AIP Configuration](#)”, presents the configuration parameters affecting the functionality of the AXI STREAM AIP.

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

Web Resources

The AMBA AXI STREAM AIP is compliant with the following specifications:

- AMBA specification ARM IHI 0051A (ID030610)
- AMBA FAQ document

Customer Support

For customer support, perform any of the following tasks:

- Enter a call through SolvNet:
 - Go to <http://onlinecase.synopsys.com/Support/OpenCase.aspx>
 - Provide the requested information, including:

- 
- **Product L1:** VC Formal
 - **Sub Product 1:** AIP
 - **Product Version:** 2020.12
 - Fill in the remaining fields according to your environment and issue.
 - Send an e-mail message to support_center@synopsys.com
 - Include product name, sub-product name, and product version (as noted above) in your e-mail, so it can be routed correctly.
 - Telephone your local support center:
 - North America:
Call 1-800-245-8005 from 7 AM to 5:30 PM Pacific time, Monday through Friday
 - All other countries:
<http://www.synopsys.com/Support/GlobalSupportCenters>

Introduction

This document describes the AXI STREAM protocol checkers available in the AXI STREAM Assertion IP (AIP). It also describes all parameters related to configurations, how to configure the AXI STREAM AIP, how to instantiate the AXI STREAM 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 attaching them to the target design, without modifying its RTL. Pre-built assertions from assertion IP are reusable and provide a powerful quality criteria for sign-off.

This chapter consists of the following sections:

- [Prerequisites](#)
- [References](#)
- [Product Overview](#)
- [Language and Methodology Support](#)
- [Feature Support](#)
- [Features Not Supported](#)

1.1 Prerequisites

Readers are assumed to be familiar with the AMBA AXI STREAM protocol, SystemVerilog Assertions, and SystemVerilog language.

1.2 References

The AXI STREAM AIP is compliant with the following specifications:

- AMBA specification ARM IHI 0051A (ID030610)
- AMBA FAQ document
(<http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.set.amba/index.html>)

1.3 Product Overview

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

The AXI STREAM AIP consists of the following:

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

1.4 Language and Methodology Support

The AXI STREAM AIP suite supports the following languages and methodology:

- SystemVerilog Assertions
- SystemVerilog 2009 or later

1.5 Feature Support

1.5.1 Protocol Features

The AXI STREAM AIP currently supports the following protocol features:

- All data widths
- All address widths
- All transfer types

1.5.2 Verification Features

The AXI STREAM AIP currently supports the following verification features:

- Ability to configure as Master, Slave, Monitor, Constraint Provider, and Interconnect
- Protocol checks
- Selective inclusion or exclusion of multiple features like CONFIG_X_CHECK, ENABLE_ASSERT, ENABLE_ASSUME and ENABLE_COVER. For more information on these parameters, see [Table 4-1](#).

1.6 Features Not Supported

- None

Installation and Setup

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

The checklist consists of the following major steps:

- [“Verifying Hardware Requirements”](#) on page 13
- [“Verifying Software Requirements”](#) on page 13
- [“Preparing for Installation”](#) on page 15

2.1 Verifying Hardware Requirements

The AXI STREAM AIP suite requires a Linux workstation configured as follows:

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

2.2 Verifying Software Requirements

This section lists software that the AXI STREAM requires.

- VCS version R-2020.12 (Simulator)
- Verdi version R-2020.12 (Debugger)
- VCF version R-2020.12 (Formal)
- VC version R-2020.12 (Verification Compiler Platform)

2.2.1 Platform/OS and Simulator Software

- VC Formal tool is required

2.2.2 Synopsys Common Licensing (SCL) Software

The AXI STREAM AIP requires the following license feature:

AIP-AXI-SVA or AIP-ACE-SVA

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

2.2.2.1 Running the AXI STREAM AIP on the VC Formal Tool

To run the AXI STREAM 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 AXI STREAM AIP on VCS

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

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

Example,

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

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

or

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

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

Table 2-1 lists the AIP licensing key features:


Table 2-1 AIP Licensing Key Features

Package Name	Feature Keys	Included Titles
VC Formal AIP AMBA5 AXI	AIP-AXI5-SVA	AXI5
VC Formal AIP AMBA AXI	AIP-AXI-SVA	AXI4, AXI3, AXI Stream
VC Formal AIP AMBA ACE	AIP-ACE-SVA	ACE, ACE-Lite, AXI4, AXI3, AXI Stream
VC Formal AIP AMBA AHB	AIP-AHB-SVA	AHB5, AHB, AHB-Lite
VC Formal AIP AMBA APB	AIP-APB-SVA	APB4, APB3, APB2
VC Formal AIP AMBA5 CHI	AIP-CHI-SVA	CHI

2.2.3 Other Third Party Software

Adobe Acrobat: The AXI STREAM AIP documentation is available as Acrobat PDF files. You can get the Adobe Acrobat Reader for free from <http://www.adobe.com>.

HTML browser: You can view the coverage reports of the AXI STREAM 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 AXI STREAM AIP, see [“Synopsys Common Licensing \(SCL\) Software”](#) on page 13.



The AXI STREAM AIP in a Formal Verification Environment

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

- [“Introduction to the VC Formal Tool”](#) on page 17
- [“The AXI STREAM AIP in a Formal Verification Environment”](#) on page 18
- [“Clock and Reset Functionality”](#) on page 24

3.1 Introduction to the VC Formal Tool

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

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

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

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

3.2 The AXI STREAM AIP in a Formal Verification Environment

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

To use the AXI STREAM AIP in a formal environment, perform the following steps in a sequence:

- Instantiate the AXI STREAM AIP and connect it to the DUT using the `bind` statement
- Creating a Tcl file
- Reading and running a Tcl file
- Analyzing results

3.2.1 Instantiating the AXI STREAM AIP Using the `bind` Statement

Create the `bind` file to instantiate the AXI STREAM AIP and bind the AXI STREAM AIP to the design. Map module and port names in the design with those of the AIP in the `bind` statement. Pass valid values to the configuration parameters of the AIP. The next step is to compile files.



Note

If a signal corresponding to the AXI STREAM AIP port does not exist in the DUT, set an inactive value or the value expected by the DUT for the port. For example, if the DUT does not have the `trready` signal, set it to `1'b1`.

3.2.2 Creating a Tcl File

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

Table 3-1 Tcl File Example

<pre># variables setting set AIP_HOME \$::env(VC_STATIC_HOME)/packages/aip set AIP_SRC \$AIP_HOME/AXI_STREAM_AIP/src set TBH_DIR ../tb set TCL_DIR ../tcl set design axi_stream_dut # vcs option set vcs " \ +incdir+\${AIP_SRC} \ \${AIP_SRC}/snps_axi_stream_aip_pkg.sv \ \${AIP_SRC}/snps_axi_stream_aip.sv \ \${TBH_DIR}/snps_axi_stream_aip_tb.v \ -assert svaext "</pre>	<pre># Enable all Formal Debug Modes set_fml_appmode FPV # analyze and elaborate design and AIP read_file -sva -top \$design -format sverilog - vcs "\$vcs" # clock setting create_clock aclk -period 100 # reset setting create_reset aresetn -low sim_run -stable sim_save_reset # execute proof check_fv -run_finish { report_fv }</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 19
- “Reading and Running Tcl File in the Batch Mode” on page 21

3.2.3.1 GUI Mode

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

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

After all the properties are executed, Verdi displays the list of properties (see [Figure 3-1](#)). In [Figure 3-1](#), the red crosses indicate falsified properties and the green tick marks indicate proven properties.

Figure 3-1 Proven and Falsified Properties

status	depth	name	vacuity	witness	engine	type	clock	elapsed_time
✗	1	...sed.ast.snps_axi_stream_aip_tkeep_absent_rule			s1	assert	...am_slave.acik	00:00:04
✓		...ady_used.ast.snps_axi_stream_aip_tdata_stable	1		e2	assert	...am_slave.acik	00:00:05
✓		...ady_used.ast.snps_axi_stream_aip_ttest_stable	1		e2	assert	...am_slave.acik	00:00:05
✓		...tready_used.ast.snps_axi_stream_aip_tid_stable	1		e2	assert	...am_slave.acik	00:00:05
✗	8	...ady_used.ast.snps_axi_stream_aip_tkeep_stable	1		s1	assert	...am_slave.acik	00:00:04
✓		...eady_used.ast.snps_axi_stream_aip_tstrb_stable	1		e2	assert	...am_slave.acik	00:00:05

name	vacuity	witness	usage	type	class	language
1 constant_76			assume	constconstraint	script	
2 ...t.gen_assumes.asm.snps_axi_stream_aip_reserved_encodings_must_not_be_used	1		assume	assume	source	SVA
3 ...ect.gen_assumes.gen_x_checks.asm.snps_axi_stream_aip_tdata_never_unknown	1		assume	assume	source	SVA
4 ...ect.gen_assumes.gen_x_checks.asm.snps_axi_stream_aip_ttest_never_unknown	1		assume	assume	source	SVA
5 ...nnect.gen_assumes.gen_x_checks.asm.snps_axi_stream_aip_tid_never_unknown	1		assume	assume	source	SVA
6 ...ect.gen_assumes.gen_x_checks.asm.snps_axi_stream_aip_tkeep_never_unknown	1		assume	assume	source	SVA

Properties: 42 - passed[39] - failed[3] - disabled[0]; Constraints Enabled: 35; Min depth: 1; Max depth: 8; Run Time: 0:00:35

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 debugged to view the options, such as View Trace, Explore the Property, Report, and so on.
2. When you select the View Trace option, waveforms are opened, providing signal details and falsification depth. You can dump other signals required for debugging into 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 properties and signal behavior in waveforms respectively.

Figure 3-2 Options to Debug Properties

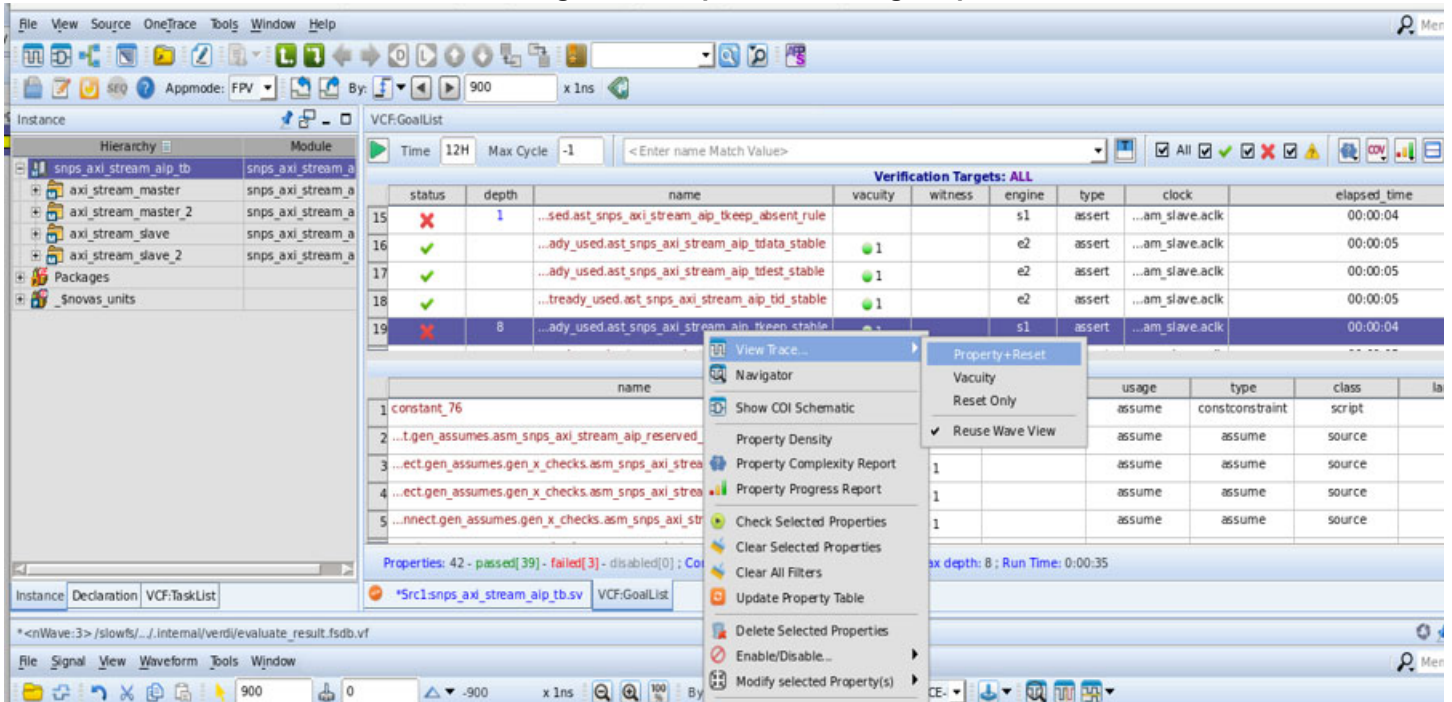
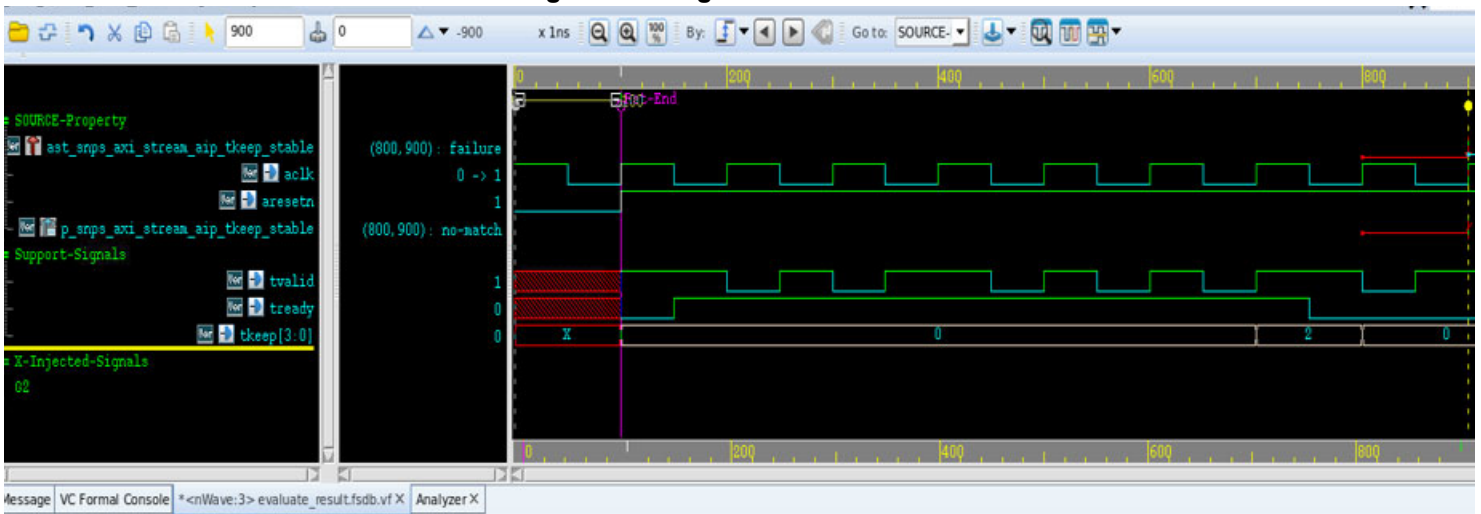


Figure 3-3 Signal Behavior in Waveforms



3.2.3.2 Reading and Running Tcl File in the Batch Mode

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

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

```
% which vcf
```

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

2. Run a Tcl file using the following command:

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

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

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

3.2.3.2.1 Analyzing Results for the Batch Mode Run

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

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

On the VC Formal window, execute the following commands:

1. `get_props -status falsified`

To display the number of falsified properties.

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

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

3.2.4 Commonly Used AXI STREAM AIP Configurations

Table 3-2 Common Usage Models

Master	Instantiates the AXI STREAM 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
Slave	Instantiates the AXI STREAM 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
Monitor	Instantiates the AXI STREAM 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
Constraint	Instantiates the AXI STREAM AIP to constraint the inputs of a DUT master and slave.
	AGENT_TYPE=CONSTRAINT
	By default, ENABLE_ASSERT=0 and ENABLE_ASSUME=1
Interconnect	Instantiates the AXI STREAM AIP to check the output behavior of an interconnect and constrain the slave.
	AGENT_TYPE=INTERCONNECT
	By default, ENABLE_ASSERT=1 and ENABLE_ASSUME=1

3.2.5 The AXI STREAM AIP As a Master

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

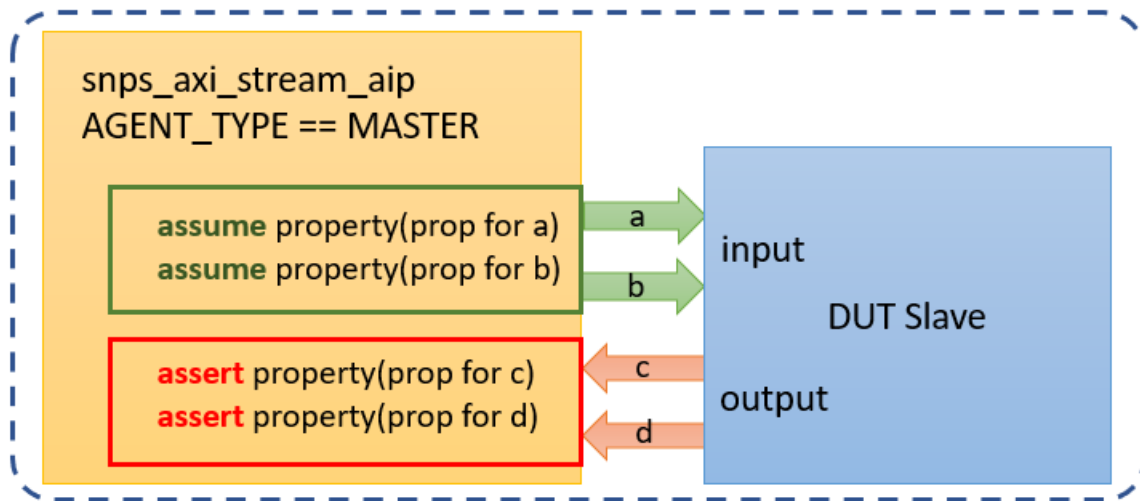
To disable all assert, assume, or cover properties, explicitly set the following parameters to value 0:

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

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

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

Figure 3-4 The AXI STREAM AIP As a Master



3.2.6 The AXI STREAM AIP As a Slave

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

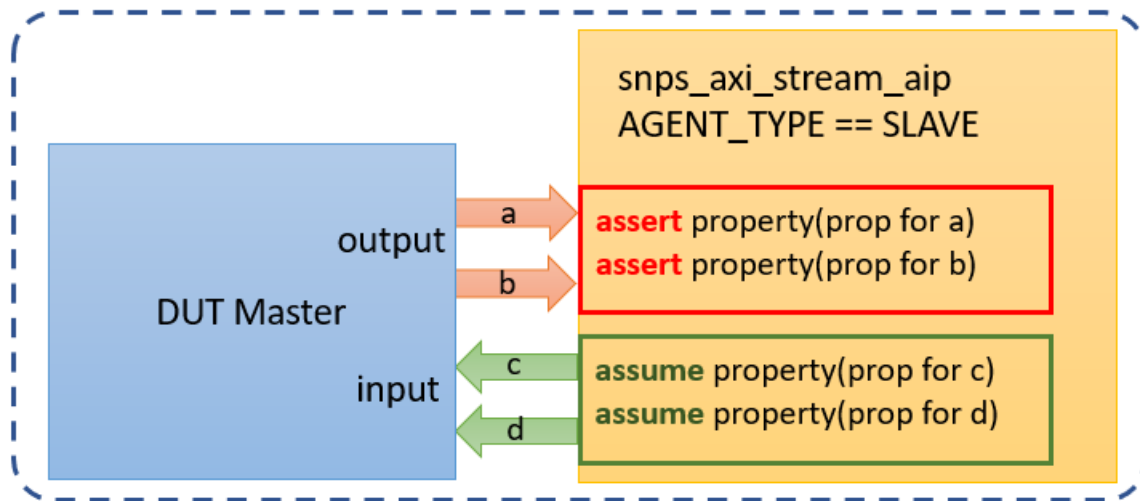
To disable all assert, assume, or cover properties, explicitly set the following parameters to value 0:

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

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

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

Figure 3-5 The AXI STREAM AIP As a Slave



3.2.7 The AXI STREAM AIP As a Monitor

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

3.2.7.1 AXI STREAM AIP Use Case as A Monitor

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

3.2.8 The AXI STREAM AIP In Constraint Mode

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


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

3.2.8.1 AXI STREAM AIP Use Case as A Constraint

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

3.3 Clock and Reset Functionality

- To run the AXI STREAM 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 AXI STREAM 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 AXI STREAM AIP Configuration

This chapter describes about configuration of the AXI STREAM AIP in the following sections:

- [“The AXI STREAM AIP Configuration Parameters”](#) on page 27
- [“The AXI STREAM AIP Interface Ports”](#) on page 29
- [“The AXI STREAM AIP Properties”](#) on page 30
- [“Behavior of Properties”](#) on page 32

4.1 The AXI STREAM AIP Configuration Parameters

[Table 4-1](#) shows the AXI STREAM AIP Configuration Parameters.

Table 4-1 The AXI STREAM AIP Configuration Parameters

Parameter Name	Description	Default value
AGENT_TYPE	Agent Type: either one of MASTER, SLAVE, MONITOR, CONSTRAINT, or INTERCONNECT	MASTER
ENABLE_ASSERT	1: Enable 0: Disable Assertions	1
ENABLE_ASSUME	1: Enable 0: Disable Assumptions	1
ENABLE_COVER	1: Enable 0: Disable Cover Properties	1
CHECK_FORMAL	1: Use Formal 0: Use Simulation/Emulation	1
CONFIG_X_CHECK	Indicates check if X signal properties or not (1: check 0: no check)	1
CONFIG_RECOMMEND	Indicates check if ARM recommendation properties or not (1: check 0: no check)	0
TDATA_NUM_BYTES	The number of bytes present in the TDATA bus	4
TID_BIT_WIDTH	The width of the TID signal in bits	8
TDEST_BIT_WIDTH	The width of the TUSER signal in bits	4
TUSER_BIT_WIDTH	The width of the TUSER signal in bits	TDATA_NUM_BYTES

Table 4-1 The AXI STREAM AIP Configuration Parameters

Parameter Name	Description	Default value
OPTIONAL_TREADY_USED	1: TREADY used by DUT 0: TREADY not used by DUT	1
OPTIONAL_TDATA_USED	1: TDATA used by DUT 0: TDATA not used by DUT	1
OPTIONAL_TLAST_USED	1: TLAST used by DUT 0: TLAST not used by DUT	1
OPTIONAL_TSTRB_USED	1: TSTRB used by DUT 0: TSTRB not used by DUT	1
OPTIONAL_TKEEP_USED	1: TKEEP used by DUT 0: TKEEP not used by DUT	1
OPTIONAL_TID_USED	1: TID used by DUT 0: TID not used by DUT	1
OPTIONAL_TDEST_USED	1: TDEST used by DUT 0: TDEST not used by DUT	1
OPTIONAL_TUSER_USED	1: TUSER used by DUT 0: TUSER not used by DUT	1
SLAVE_SUPPORTS_POSITION_BYTES	1: The Slave can accept position bytes 0: The Slave cannot accept position bytes, and none must be seen in the interface.	1
SLAVE_SUPPORTS_NULL_BYTES	1: The Slave can accept null bytes 0: The Slave cannot accept null bytes, and none must be seen in the stream.	1
SLAVE_WAITS_FOR_TVALID	1: The Slave waits until the Master asserts TVALID 0: The slave do not wait until the Master asserts TVALID.	0

4.2 The AXI STREAM AIP Interface Ports

Table 4-2 describes interface signals of the AXI STREAM AIP:

Table 4-2 The AXI STREAM AIP Interface Ports

Signal Name	Signal width	Description	
		Source	Destination
acclk	1	Input clock from the system	
aresetn	1	Reset input from the system	
tvalid	ID_WIDTH	Master	Slave
tready	ADDR_WIDTH	Slave	Master
tdata	(8*TDATA_NUM_BYTES)	Master	Slave
tstrb	TDATA_NUM_BYTES	Master	Slave
tkeep	TDATA_NUM_BYTES	Master	Slave
tlast	1	Master	Slave
tid	TID_BIT_WIDTH	Master	Slave
tdest	TDEST_BIT_WIDTH	Master	Slave
tuser	TDEST_BIT_WIDTH	Master	Slave

4.3 The AXI STREAM AIP Properties

4.3.1 Properties (Assertions/Assumptions)

Table 4-3 shows the AXI STREAM AIP properties.

Table 4-3 AXI STREAM AIP Properties

Property Name	Master/Slave	Error Kind	Spec Reference	Property Description
ast_snps_axi_stream_aip_tkeep_absent_rule	Master	ERROR	[ARM IHI 0051A] Section 3.1.2 on page 3-2	When TKEEP is absent, TKEEP defaults to all bits HIGH.
ast_p_snps_axi_stream_aip_tstrb_absent_rule	Master	ERROR	[ARM IHI 0051A] Section 3.1.2 on page 3-2	When TSTRB is absent, TSTRB = TKEEP.
ast_snps_axi_stream_aip_tkeep_and_tstrb_absent_rule	Master	ERROR	[ARM IHI 0051A] Section 3.1.2 on page 3-2	When TSTRB and TKEEP are absent, TSTRB and TKEEP default to all bits HIGH.
ast_snps_axi_stream_aip_tvalid_steady	Master	ERROR	[ARM IHI 0051A] Section 2.2.1 on page 2-3	Once TVALID is asserted, it must remain asserted until the handshake occurs.
ast_snps_axi_stream_aip_tdata_stable	Master	ERROR	[ARM IHI 0051A] Section 2.2.1 on page 2-3	Once the master has asserted TVALID, the data or control information from the master must remain unchanged until the slave drives the TREADY signal HIGH.
ast_snps_axi_stream_aip_tstrb_stable	Master	ERROR	[ARM IHI 0051A] Section 2.2.1 on page 2-3	Once the master has asserted TVALID, the data or control information from the master must remain unchanged until the slave drives the TREADY signal HIGH.
ast_snps_axi_stream_aip_tkeep_stable	Master	ERROR	[ARM IHI 0051A] Section 2.2.1 on page 2-3	Once the master has asserted TVALID, the data or control information from the master must remain unchanged until the slave drives the TREADY signal HIGH.
ast_snps_axi_stream_aip_tid_stable	Master	ERROR	[ARM IHI 0051A] Section 2.2.1 on page 2-3	Once the master has asserted TVALID, the data or control information from the master must remain unchanged until the slave drives the TREADY signal HIGH.
ast_snps_axi_stream_aip_tdest_stable	Master	ERROR	[ARM IHI 0051A] Section 2.2.1 on page 2-3	Once the master has asserted TVALID, the data or control information from the master must remain unchanged until the slave drives the TREADY signal HIGH.
ast_snps_axi_stream_aip_tuser_stable	Master	ERROR	[ARM IHI 0051A] Section 2.2.1 on page 2-3	Once the master has asserted TVALID, the data or control information from the master must remain unchanged until the slave drives the TREADY signal HIGH.

Table 4-3 AXI STREAM AIP Properties

Property Name	Master/Slave	Error Kind	Spec Reference	Property Description
ast_snps_axi_stream_aip_reserved_encodings_must_not_be_used	Master	ERROR	[ARM IHI 0051A] Section 2.4.3 on page 2-9	Reserved (data type) - Must not be used.
ast_snps_axi_stream_aip_tdata_never_unknown	Master	ERROR	Not In the Specification	When valid, TDATA should be a known value.
ast_snps_axi_stream_aip_tstrb_never_unknown	Master	ERROR	Not In the Specification	When valid, TSTRB should be a known value.
ast_snps_axi_stream_aip_tkeep_never_unknown	Master	ERROR	Not In the Specification	When valid, TKEEP should be a known value.
ast_snps_axi_stream_aip_tid_never_unknown	Master	ERROR	Not In the Specification	When valid, TID should be a known value.
ast_snps_axi_stream_aip_tdest_never_unknown	Master	ERROR	Not In the Specification	When valid, TDEST should be a known value.
ast_snps_axi_stream_aip_tuser_never_unknown	Master	ERROR	Not In the Specification	When valid, TUSER should be a known value.
ast_snps_axi_stream_aip_tvalid_not_unknown_out_of_reset	Master	ERROR	Not in the Specification	TVALID should have a known value out of reset.
ast_snps_axi_stream_aip_tvalid_eventually_gets_tready	Slave	ERROR	Not in the Specification	The handshake process is expected to remain live.
ast_snps_axi_stream_aip_optional_tid_zero_if_not_used	Slave	ERROR	[ARM IHI 0051A] Section 3.1.4 on page 3-3	A slave with additional TID inputs must have all bits fixed LOW.
ast_snps_axi_stream_aip_optional_tdest_zero_if_not_used	Slave	ERROR	[ARM IHI 0051A] Section 3.1.4 on page 3-3	A slave with additional TDEST inputs must have all bits fixed LOW.
ast_snps_axi_stream_aip_optional_tuser_zero_if_not_used	Slave	ERROR	[ARM IHI 0051A] Section 3.1.4 on page 3-3	A slave with additional TUSER inputs must have all bits fixed LOW.
ast_snps_axi_stream_aip_slave_gets_pos_bytes_when_not_supported	Slave	ERROR	[ARM IHI 0051A] Section 3.2.2 on page 3-5	If a slave does not support position bytes, all [position] bytes must be converted to data bytes.
ast_snps_axi_stream_aip_slave_gets_null_bytes_when_not_supported	Slave	ERROR	[ARM IHI 0051A] Section 3.2.2 on page 3-5	If a slave does not support null bytes, then a component that performs packing is used to remove null bytes from the stream.

4.3.2 The AXI STREAM AIP Cover Properties

Table 4-4 shows the AXI STREAM AIP cover properties.

Table 4-4 The AXI STREAM AIP Cover Properties

Cover Property Name	Property Description
cov_snps_axi_stream_aip_tid_width_recm	The Width of TID is within recommended limits
cov_snps_axi_stream_aip_tdest_width_recm	The width of TDEST is within recommended limits
cov_snps_axi_stream_aip_master_asserts_tvalid_without_tready	Spec Section 2.2.1 statement: "A master is not permitted to wait until TREADY is asserted before asserting TVALID"
cov_snps_axi_stream_aip_tready_seen_dropping_without_tvalid	Spec Section 2.2.1 statement: "If a slave asserts TREADY, it is permitted to deassert TREADY before TVALID is asserted"

4.4 Behavior of Properties

Properties are grouped on the basis of categories, which depends on the configuration parameter values. Categories can be like `x_check` properties, configure command properties and max waits properties.

Parameters have some default value, refer to [The AXI STREAM AIP Configuration Parameters](#) section to know default values.

- To instantiate `x_check` properties:

Set `CONFIG_X_CHECK=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.

4.4.1 Properties In Assert Directives

Assert properties have the following features:

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

4.4.2 Properties As Assume Directives

Assume properties have the following features:

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

- When AGENT_TYPE is SLAVE, the checkers mentioned as 'Slave' in Master/Slave column are declared as assume, and checkers mentioned as 'Master' in Master/Slave column are declared as assert.

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



The AXI STREAM AIP Use Cases

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

5.1 The AXI STREAM AIP Examples

This section describes the setup of the AXI STREAM AIP where AIP is connected with RTL through a bind file. The bind file example is shown in [Figure 5-1](#), [Figure 5-2](#), [Figure 5-3](#) and [Figure 5-4](#), where both master and slave DUT signals are connected to AIP master and slave signals.



Note

If a signal corresponding to the AXI STREAM AIP port does not exist in the DUT, set an inactive value or the value expected by the DUT for the port. For example, if the DUT does not have the `awakeup` signal, set with `1'b0`.

5.1.1 The AXI STREAM Master AIP With Slave DUT

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

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



Note

Use `snps_axi_stream_aip.sv` file for AXI STREAM AIP and configure `AGENT_TYPE` parameter with `MASTER`.

Figure 5-1 Slave DUT with AXI STREAM Master AIP

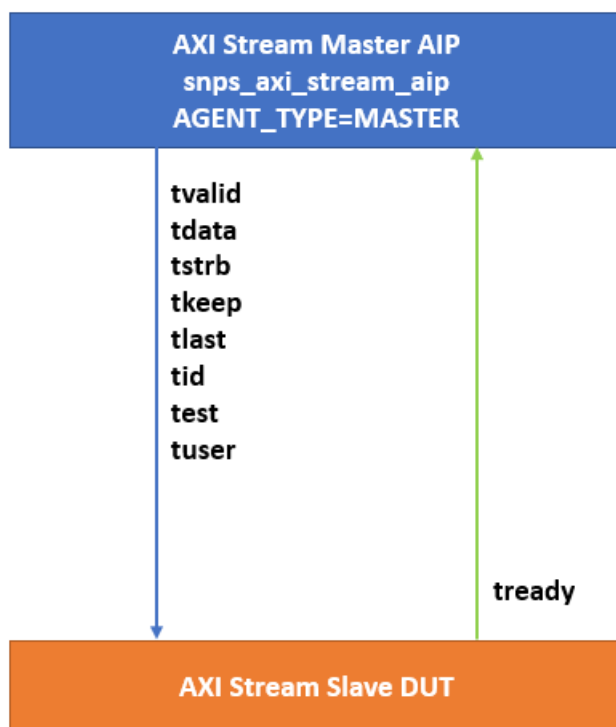
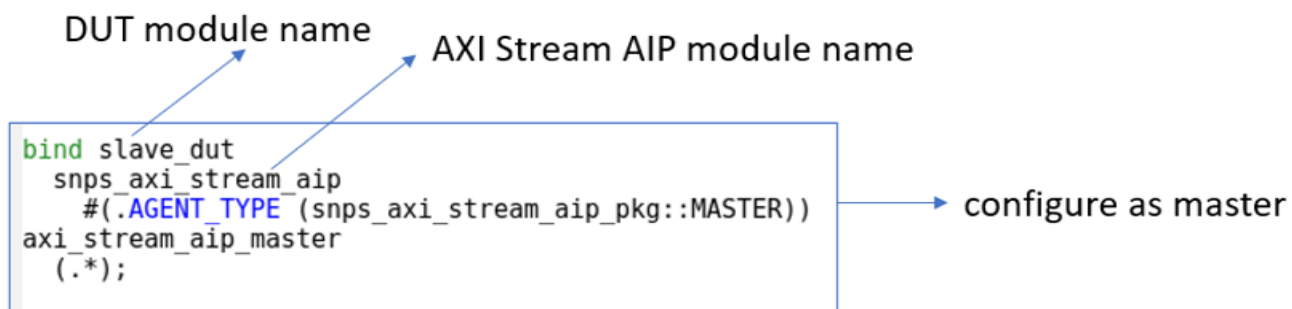


Figure 5-2 AXI STREAM Master AIP – Slave DUT bind example



5.1.2 The AXI STREAM Slave AIP With a Master DUT

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

- In this setup, the AXI STREAM Slave AIP is connected with the AXI STREAM Real Master DUT.
- For connecting the ports of the AXI STREAM DUT with the AXI STREAM AIP, create a bind file to include the instance of the AXI STREAM AIP into the top level module of the DUT.
- Instantiate the AXI STREAM Slave AIP (`snps_axi_stream_aip`) and connect with the master DUT signal.
- [Figure 5-3](#) shows a Master DUT connected with the AXI STREAM Slave AIP.



Note

Use `snps_axi_stream_aip.sv` file for AXI STREAM AIP and configure `AGENT_TYPE` parameter with `SLAVE`.

Figure 5-3 Master DUT with AXI STREAM Slave AIP

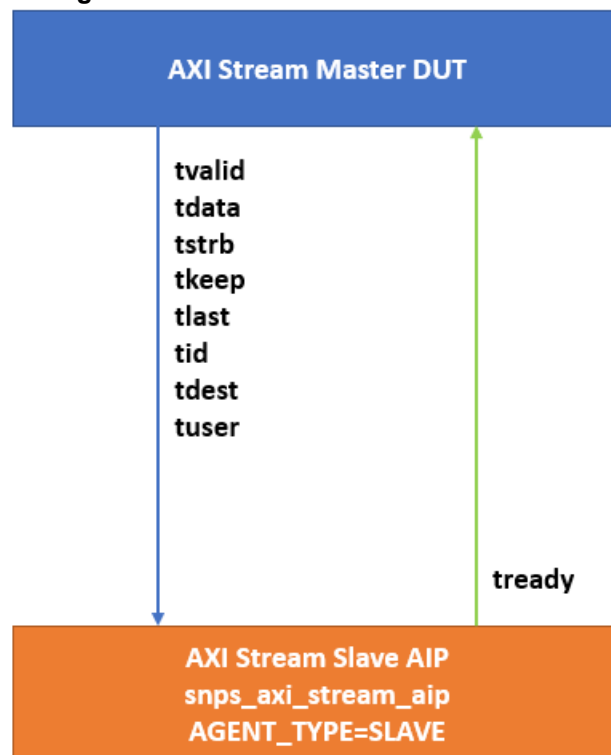


Figure 5-4 AXI STREAM Slave AIP – Master DUT bind example

