

# **APB 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®/VCSi™ and VCS® MX/VCS® MXi™ 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/VCS MX 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.

Synopsys, Inc.  
690 E. Middlefield Road  
Mountain View, CA 94043  
[www.synopsys.com](http://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 .....	12
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 APB AIP in a Formal Verification Environment .....	17
3.1 Introduction to the VC Formal Tool .....	17
3.2 The APB AIP in a Formal Verification Environment .....	17
3.2.1 Instantiating the APB 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 Most Commonly Used Configurations .....	23
3.2.5 The APB AIP As a Master .....	23
3.2.6 The APB AIP As a Slave .....	24
3.2.7 The APB AIP As a Monitor .....	25
3.2.8 The APB AIP In Constraint Mode .....	25
3.3 Clock and Reset Functionality .....	25

Chapter 4 .....	
The APB AIP Configuration .....	27
4.1 The APB AIP Configuration Parameters .....	27
4.2 Interface Ports .....	31
4.3 The APB AIP Properties .....	31
4.4 Behavior of Properties .....	39
4.4.1 Properties In Assert Directives .....	39
4.4.2 Properties In Assume Directives .....	39
4.4.3 Properties In Cover Directives .....	39
Chapter 5 .....	
The APB AIP Use Cases .....	41
5.1 The APB AIP Examples .....	41
5.1.1 The APB AIP with a Master DUT .....	41
5.1.2 The APB AIP With a Slave DUT .....	43

# Figures

---

Figure 3-1:	VC Formal Tool showing Results of Properties .....	20
Figure 3-2:	VC Formal Tool Showing Options to Debug Failing Properties .....	21
Figure 3-3:	Waveforms Opened Through the GUI Mode .....	21
Figure 3-4:	Waveforms of Failing Property Opened in the Batch Mode .....	22
Figure 3-5:	The APB AIP As a Master .....	24
Figure 3-6:	The APB AIP As a Slave .....	25
Figure 5-1:	Master DUT With the Slave APB AIP .....	42
Figure 5-2:	Slave DUT With the Master APB AIP .....	44



# Tables

---

Table 2-1:	AIP Licensing Key Features .....	14
Table 3-1:	Tcl File Example .....	19
Table 3-2:	Common Usage Models .....	23
Table 4-1:	The APB AIP Configuration Parameters .....	27
Table 4-2:	The APB AIP Interface Ports .....	31
Table 4-3:	APB Properties .....	32
Table 4-4:	The APB AIP Cover Properties .....	38
Table 5-1:	Master DUT With Slave AIP Bind .....	43
Table 5-2:	Slave DUT With the Master APB AIP Bind .....	45







# Preface

---

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

## Guide Organization

The chapters of this guide are organized as follows:

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

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

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

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

## Web Resources


The APB AIP is compliant with the following specifications:

- AMBA specification (ARM IHI 0024B) (ARM IHI 0024C)
- 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 APB protocol checkers available in the APB Assertion IP (AIP). It also describes all parameters related to configurations, how to configure the APB AIP, how to instantiate the APB AIP, and so on.

The APB AIP 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 the APB AIP provides 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

Familiarize with APB protocol, SystemVerilog Assertion knowledge, and Verilog concepts.

## 1.2 References

The APB AIP is compliant with the following specifications:

- AMBA specification (ARM IHI 0024B) (ARM IHI 0024C)
- AMBA FAQ document  
(<http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.set.amba/index.html>)

### 1.3 Product Overview

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

The APB AIP consists of the following:

- Assertion properties
- Assume properties
- Cover properties
- Synthesizable modeling logic for the Properties

### 1.4 Language and Methodology Support

The APB AIP supports the following languages and methodology:

- SystemVerilog Assertions
- Verilog

### 1.5 Feature Support

#### 1.5.1 Protocol Features

The APB AIP currently supports the following protocol features:

- AMBA2, AMBA3, and AMBA4 APB support
- All data widths
- All address widths

#### 1.5.2 Verification Features

The APB AIP can be used to verify the RTL in configurations where:

- APB AIP acts as master
- APB AIP acts as slave
- APB AIP acts as multiple slaves
- APB AIP acts as monitor
- Protocol checks based on the configuration parameters, such as, CONFIG\_X\_CHECK, PSLVERR\_RCMND, and so on. For more information on configuration parameters, see [Tables 4-1](#).

### 1.6 Features Not Supported

- None

# Installation and Setup

---

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

The checklist consists of the following major steps:

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

## 2.1 Verifying Hardware Requirements

The APB 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 APB 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 APB AIP requires the following license feature:

AIP-xxx-SVA

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

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

To run the APB 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 APB AIP on VCS

To run the APB 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 APB 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 APB 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 13.





# The APB AIP in a Formal Verification Environment

---

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

- [“Introduction to the VC Formal Tool”](#) on page 17
- [“The APB AIP in a Formal Verification Environment”](#) on page 17
- [“Clock and Reset Functionality”](#) on page 25

## 3.1 Introduction to the VC Formal Tool

The VC Formal tool is used to verify assertion properties by examining all sequences of possible value combinations for the signals that it monitors. These signal values can be constrained either by the DUT that is driving them or by the assume properties in the APB AIP or by a combination of both. The VC Formal tool provides the following information:

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

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

- The path of the APB AIP source code
- The path of the RTL source code (if validating the RTL)
- Clock and reset information
- Timeout details

## 3.2 The APB AIP in a Formal Verification Environment

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

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

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

### 3.2.1 Instantiating the APB AIP Using the `bind` Statement

Create a bind file to bind the APB AIP with a design. Map modules and port names in the design with those of the APB AIP in the bind statement. Pass valid values to the configuration parameters of the APB AIP. The next step is to compile files.



#### Note

If a signal corresponding to the APB 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 `pslverr` signal, set with `1'b0`.

### 3.2.2 Creating a Tcl File

To compile the RTL files, the APB AIP files, and the bind file, create a Tcl file to set the path of the RTL directory, the APB AIP directory, and the bind file. The DUT clock and reset are initialized, as shown in [Table 3-1](#). In Tcl, you can mention the valid timeout as per the requirement. VC Formal can generate the logs on the basis of proven, falsified, vacuous, and inconclusive properties.

**Table 3-1 Tcl File Example**

<pre> set AIP_HOME \$::env(AIP_HOME) # Source code and tb and Tcl path SetTEST_DIR \$AIP_HOME/example set AIP_SRC_DIR \$AIP_HOME/esrc set TCL_DIR \$TEST_DIR/apb_test/tcl set TB_DIR \$TEST_DIR/apb_test/tb  if { [ file exists setup.tcl ] == 1 } {     source setup.tcl -echo -verbose } set hierarchy_delimiter "."  #Enable all Formal debug modes set_app_var fml_mode_on true  # Timeout settings set_fml_var fml_max_time 1H  #Initialization - reset sequence proc initd {} {     create_clock &lt;design clock&gt; -period 100     create_reset &lt;design reset&gt; -low     sim_config -rst_wave ON -replay_all_wave ON     sim_run -stable     sim_save_reset } </pre>	<pre> <b># Commands to run the files in VC Static</b> proc compd {} {     global AIP_HOME TB_DIR AIP_SRC_DIR TCL_DIR     read_file -sva -top dummy_dut_common -     format sverilog -vcs "         -sverilog         +incdir+\${AIP_SRC_DIR}         \${AIP_SRC_DIR}/snps_apb_aip_pkg.sv         \${AIP_SRC_DIR}/snps_apb_aip.sv         \${AIP_SRC_DIR}/snps_apb_aip_slave_mux.sv         \${TB_DIR}/dummy_dut_common.v         \${TB_DIR}/bind_lmls.sv         -parameters \$TCL_DIR/config1.param         -assert svaext "     }      compd     initd  <b># To run in a batch regression</b>     check_fv -block     report_fv      OR  <b># To run in an interactive debug mode</b>     check_fv -run_finish {     report_fv -list &gt; \$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 19
- “Reading and Running Tcl File in Batch Mode” on page 21

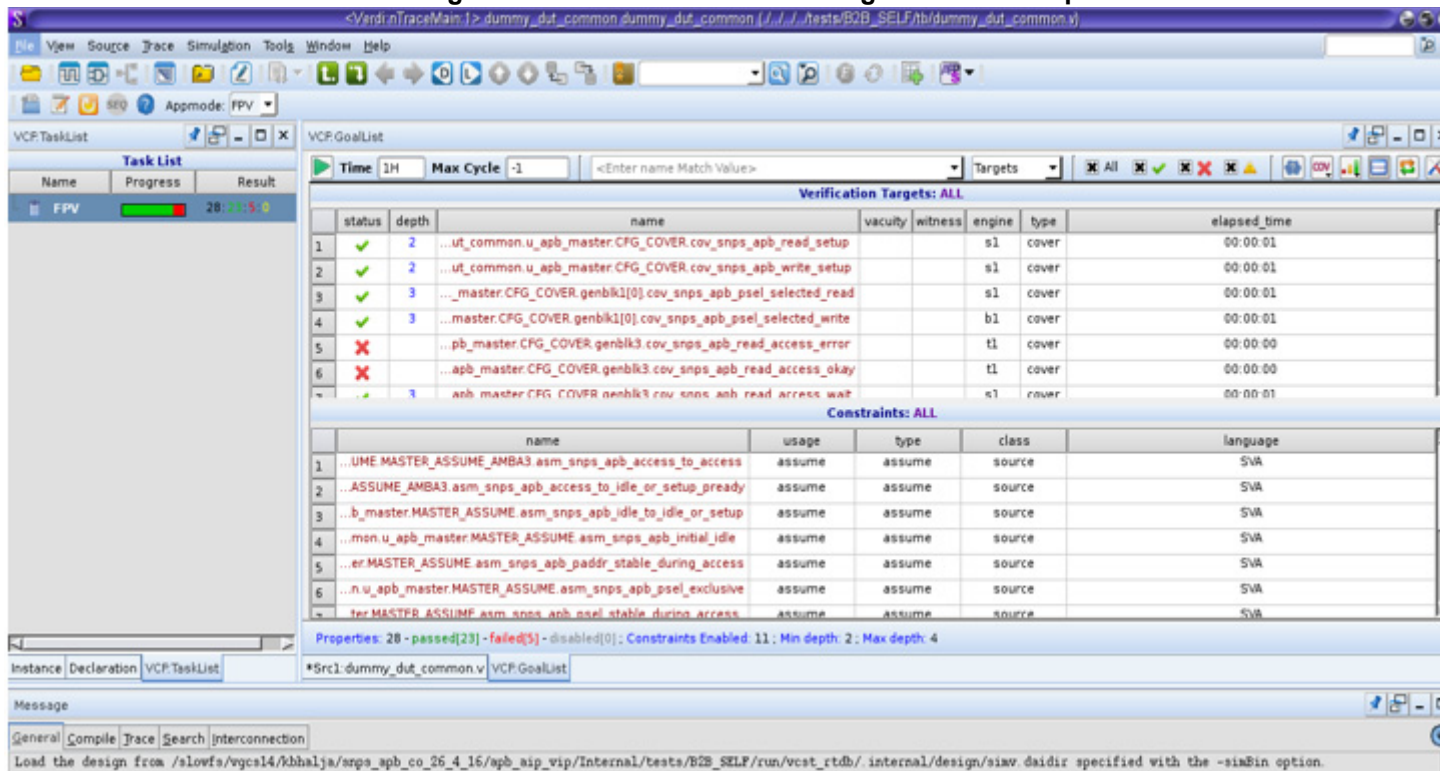
#### 3.2.3.1 GUI Mode

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



### 3.2.3.1.1 Analyzing Results for 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, 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 falsified properties and waveforms in the GUI mode respectively.

Figure 3-2 VC Formal Tool Showing Options to Debug Failing Properties

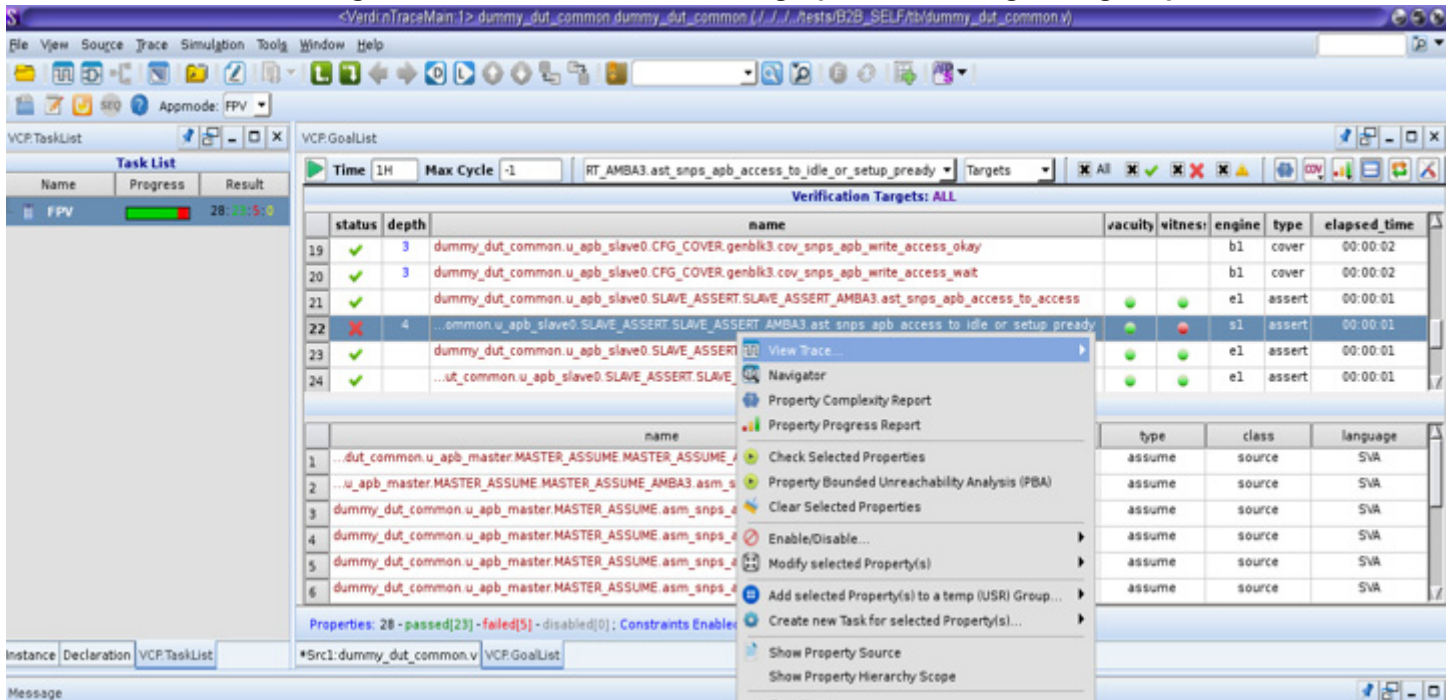
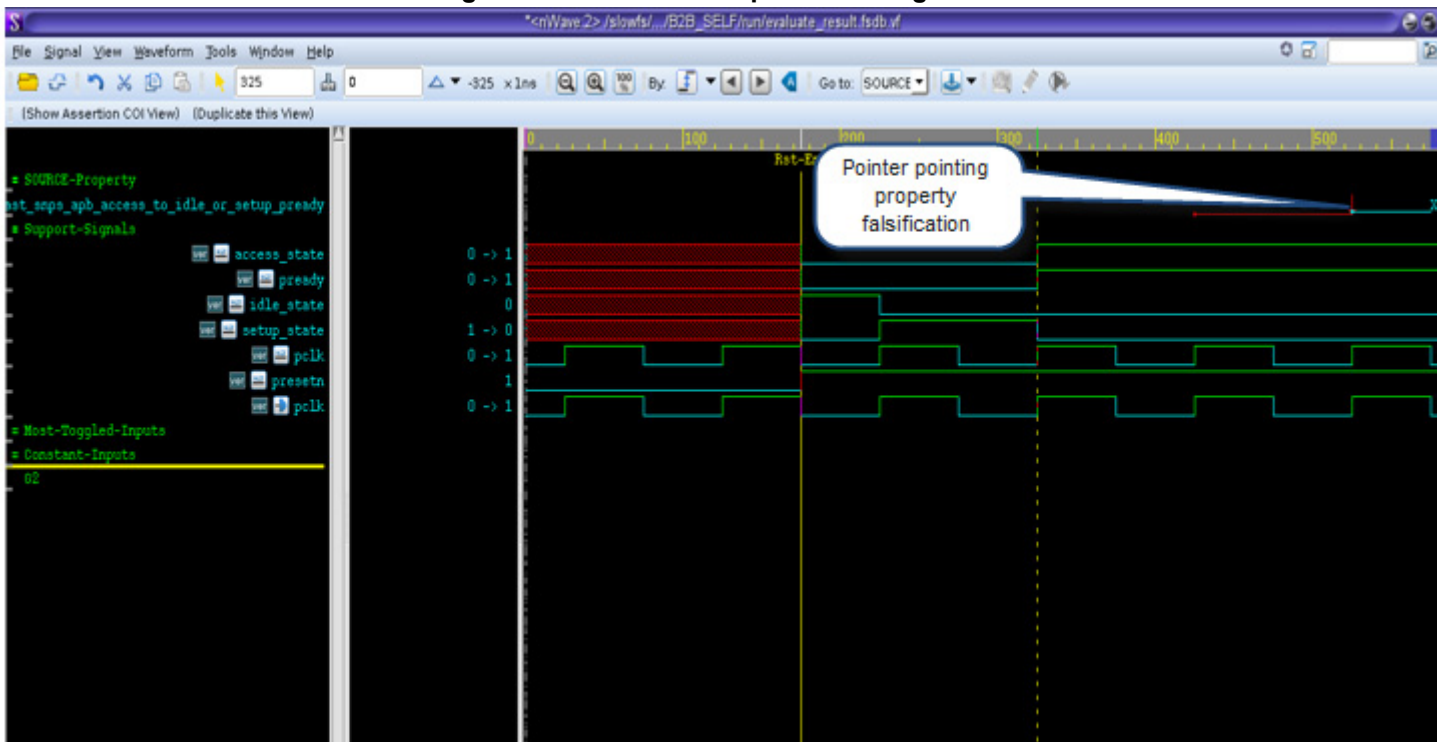


Figure 3-3 Waveforms Opened Through the GUI Mode



### 3.2.3.2 Reading and Running Tcl File in 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 Formal Tcl file.

### 3.2.3.2.1 Analyzing Results for 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`

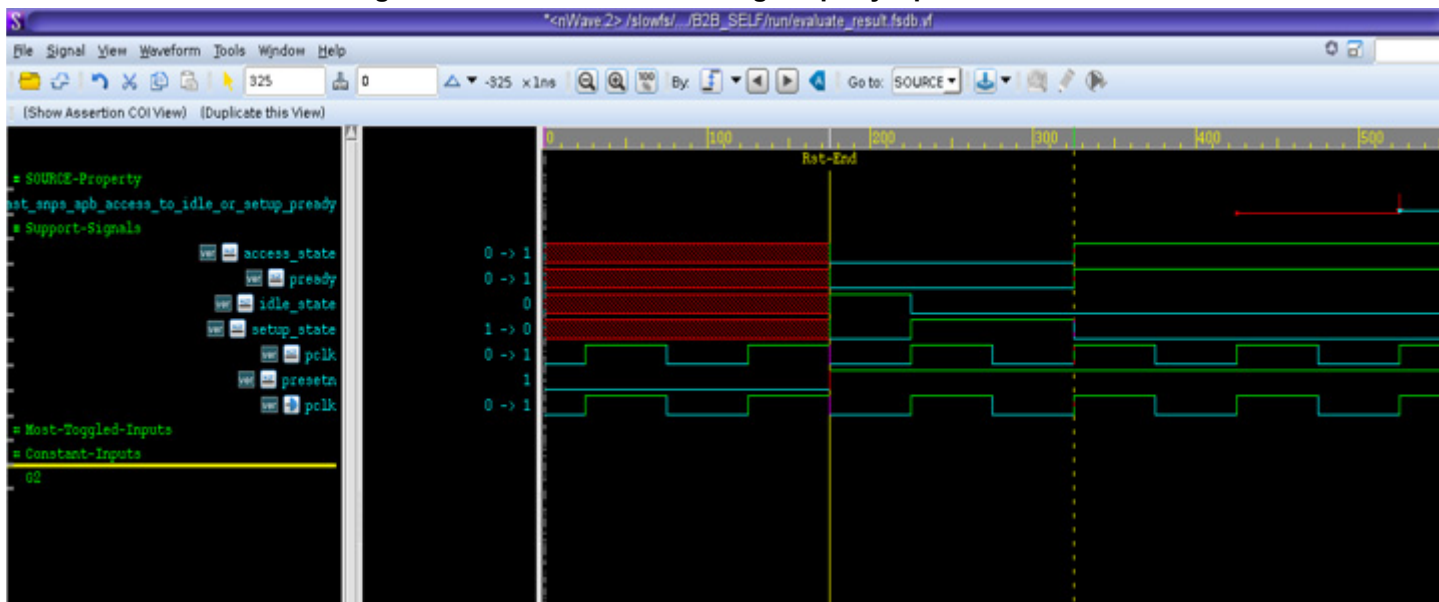
To display the number of falsified properties.

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.

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

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





### 3.2.4 Most Commonly Used Configurations

Table 3-2 Common Usage Models

1	Master	Instantiates the APB 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 APB 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 APB 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 APB 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 APB AIP As a Master

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

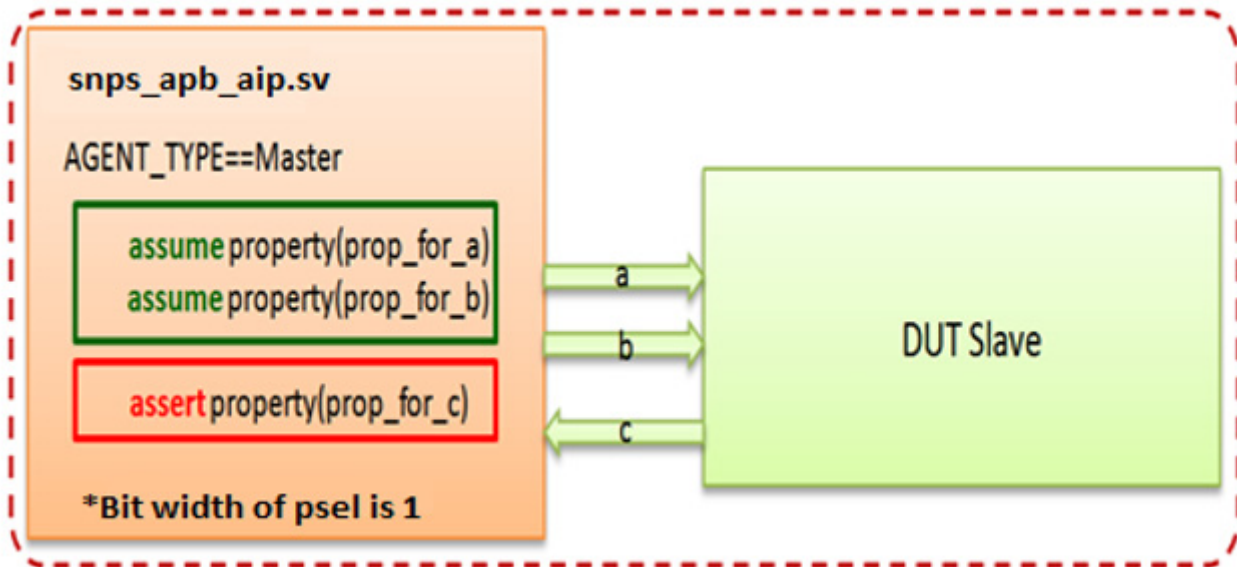
To disable all `assert`, `assume`, or `cover` properties, explicitly set the following parameters to value 0:

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

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

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

Figure 3-5 The APB AIP As a Master



### 3.2.6 The APB AIP As a Slave

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

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

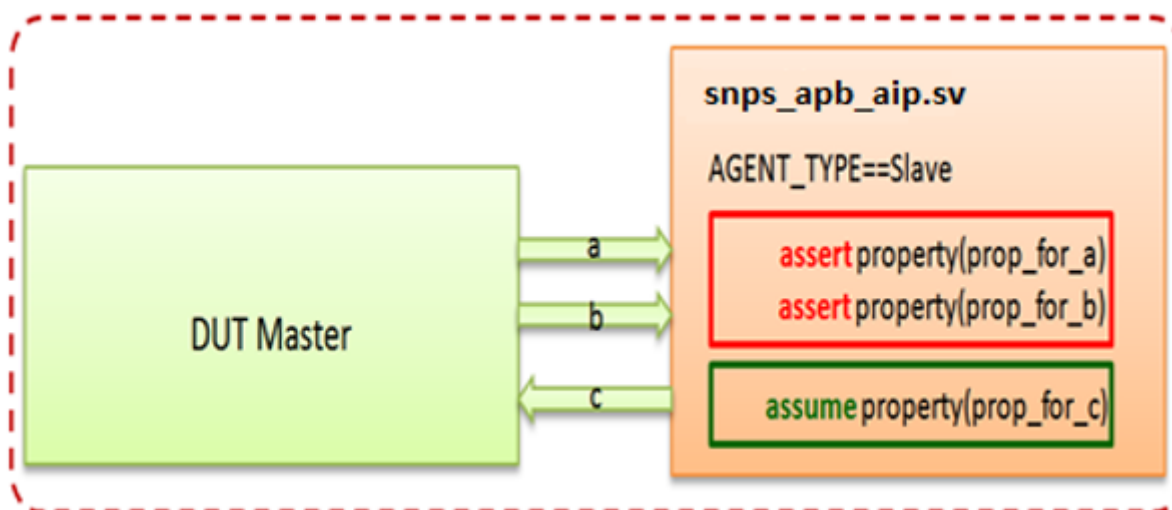
- `ENABLE_ASSERT`
- `ENABLE_ASSUME`
- `ENABLE_COVER`

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

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



Figure 3-6 The APB AIP As a Slave



### 3.2.7 The APB AIP As a Monitor

To verify the behavior of an APB master and slave DUT, set the `AGENT_TYPE` parameter to `MONITOR` during the APB AIP instantiation. When this parameter is set to `MONITOR`, all the APB AIP properties are declared as `assert`, that is, both inputs and outputs of the DUT are checked.

By default, `ENABLE_ASSERT=1` and `ENABLE_ASSUME=0`. This disables all assume properties.

### 3.2.8 The APB AIP In Constraint Mode

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

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

## 3.3 Clock and Reset Functionality

- To run the APB 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 APB 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 design.

The formal analysis of properties starts after the reset state.



# The APB AIP Configuration

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

- “The APB AIP Configuration Parameters” on page 27
- “Interface Ports” on page 31
- “The APB AIP Properties” on page 31
- “Behavior of Properties” on page 39

## 4.1 The APB AIP Configuration Parameters

Table 4-1 shows the user-defined parameters for setting the interface characteristics of the APB AIP. You can change these parameters to match your design specification.

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

Parameter	Description	Default Value
PROTOCOL_VER	AMBA2/AMBA3/AMBA4 protocol version selection.	AMBA3
AGENT_TYPE	<p>Specifies the agent type.</p> <p>MASTER:</p> <ul style="list-style-type: none"> <li>Instantiates the APB AIP as a master to check the master APB AIP input behavior with assert and constraint the master output.</li> <li>By default: ENABLE_ASSERT=1 and ENABLE_ASSUME=1</li> </ul> <p>SLAVE:</p> <ul style="list-style-type: none"> <li>Instantiates the APB AIP as a slave to check the slave APB AIP input behavior with assert and constraint the APB AIP output.</li> <li>By default: ENABLE_ASSERT=1 and ENABLE_ASSUME=1</li> </ul>	MASTER

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

Parameter	Description	Default Value
	<p>MONITOR:</p> <ul style="list-style-type: none"> <li>Instantiates the APB AIP as a monitor to check the DUT slave behavior and the DUT master behavior.</li> <li>By default: ENABLE_ASSERT=1 and ENABLE_ASSUME=0</li> </ul> <p>CONSTRAINT:</p> <ul style="list-style-type: none"> <li>Instantiates the APB AIP to constraint the DUT slave input and the DUT master input.</li> <li>By default: ENABLE_ASSERT=0 and ENABLE_ASSUME=1</li> </ul>	
ENABLE_ASSERT	Set this parameter to 1/0 to enable/disable assertions of the selected agent type	1
ENABLE_ASSUME	Set this to 1/0 to enable/disable assume properties of the selected agent type	1
ENABLE_COVER	Set this parameter to 1/0 to enable/disable cover properties of the selected agent type	1
PSEL_WIDTH	Denotes the number of slaves the APB AIP should configured as	1
ADDR_WIDTH	Denotes the width of the address bus	32
WDATA_WIDTH	Denotes the width of the write data bus	32
RDATA_WIDTH	Denotes the width of the read data bus	32
CHECK_FORMAL	Set this parameter to 1/0 for selecting instantiation of the formally optimized/unoptimized properties	1
CONFIG_LOWPOWER	Set this parameter to 1/0 to enable/disable group of properties supporting low power feature	0
CONFIG_X_CHECK	Set this parameter to 1/0 to enable/disable group of properties for checking the x value of the signal	0
PSLVERR_RCMND	Set this parameter to 1/0 to enable/disable group of properties supporting APB3	1
CHECK_MAX_WAITS	Set this parameter to 1/0 to enable/disable PREADY latency check	1
MAX_WAITS	Set the maximum latency for PREADY	16

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

Parameter	Description	Default Value
FRV	Enable or disable FRV interface 0: Disable FRV interface 1: Enable FRV interface	0
FRV_SLAVE	Indicates which <code>p_selx</code> bit corresponds to FRV target slave. If <code>FRV_SLAVE &lt; PSEL_WIDTH</code> : <code>p_selx[FRV_SLAVE]</code> is used for FRV If <code>FRV_SLAVE &gt;= PSEL_WIDTH</code> : all slaves, ored of <code>p_selx</code> , that is, <code>lp_selx</code> is used others: <code>p_selx[0]</code> is used	0
FRV_RSVD_MODE	Reserved mode 0: no check when <code>ACCESS_TYPE==ACS_RSVD</code> 1: 0 should be read when <code>ACCESS_TYPE==ACS_RSVD</code> and <code>CHECK_INIT==0</code> 2: 1 should be read when <code>ACCESS_TYPE==ACS_RSVD</code> and <code>CHECK_INIT==0</code>	1
FRV_WR_LATENCY	Default Write Latency bus write propagates to BackDoor signal.	1
FRV_RD_LATENCY	Default Read Latency from BackDoor signal to bus read.	1
FRV_LATENCY_MD	Latency handling 0: <code>WR/RD_LATENCY</code> parameters are used only in back door check 1: <code>WR/RD_LATENCY</code> parameters are used in all checks	0
FRV_OUTABLK_MD	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	2
FRV_RD_ACT_LATENCY	readAction Latency	1
FRV_RO_RD_CHK	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.	0

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

Parameter	Description	Default Value
FRV_COMPOSITE	Type of assertion per field for initial value and after write. 1: Generate 1 common assertion (better convergence). 0: Generate 2 individual assertions.	1
FRV_CHECK_FORMAL	Use (1) or do not use (0) Formal optimized property.	0
FRV_SNPS_ND_ABS	Synopsys Non-Deterministic Abstractions 0: check all possible scenarios 1: check only specific scenario in each step	0
FRV_COVER_WRITE	Enable/Disable cover properties for write transactions 0: Disable all write related cover properties 1: Enable write related cover properties	1
FRV_COVER_READ	Enable/Disable cover properties for read transactions 0: Disable all read related cover properties 1: Enable read related cover properties	1
FRV_COVER_TYPE	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	3'h7
FRV_OUTADDR_EN	Out of address range cover properties 0: Disable out of address cover properties 1: Enable out of address cover properties	1

## 4.2 Interface Ports

Table 4-2 describes the interface signals of AMBA2 and AMBA3 APB AIP:

**Table 4-2 The APB AIP Interface Ports**

Serial Number	Signal Name	AMBA2/AMBA3/AMBA4	Description
1	pclk	AMBA2,AMBA3, AMBA4	Denotes the APB bus clock. The pclk signal is an output from the master and input to the slave.
2	presetn	AMBA2,AMBA3, AMBA4	Denotes an active low bus reset. The presetn signal is an output from the master and input to the slave.
3	pselx	AMBA2,AMBA3, AMBA4	Denotes a slave select bus. The pselx signal is an output from the master and input to the slave.
4	penable	AMBA2,AMBA3, AMBA4	Strobe denoting transfer of data. The penable signal is an output from the master and input to the slave.
5	pwrite	AMBA2,AMBA3, AMBA4	Qualifies whether transfer is read or write. The pwrite signal is an output from the master and input to the slave.
6	paddr	AMBA2,AMBA3, AMBA4	Denotes the address bus. The paddr signal is an output from master and input to the slave.
7	pwwdata	AMBA2,AMBA3, AMBA4	Denotes the write data bus. The pwwdata signal is an output from the master and input to the slave.
8	prdata	AMBA2,AMBA3, AMBA4	Denotes the read data bus. The prdata signal is an output from the slave and input to the master.
9	pready	AMBA3, AMBA4	Denotes the slave ready signal. The pready signal is an output from the slave and input to the master.
10	pslverr	AMBA3, AMBA4	Denotes the slave error signal. The pslverr signal is an output from the slave and input to the master.
11	pprot	AMBA4	Denotes Protection type.
12	pstrb	AMBA4	Denotes Write strobes.

## 4.3 The APB AIP Properties

This section describes the property name, type, and behavior of AMBA 2 and AMBA 3. Table 4-3 and Table 4-4 includes assert and assume properties of Master and Slave for AMBA2 and AMBA3.

**Table 4-3 APB Properties**

Property Name	Parameters	Error Kind	Spec Reference	Property Description
ast_snps_apb_psel_exclusive	((AGENT_TYPE==snps_apb_aip_pkg::MONITOR    (AGENT_TYPE==snps_apb_aip_pkg::SLAVE && PSEL_WIDTH>1)) && ENABLE_ASSERT==1)	ERROR	[ARM IHI 0011A] Section 5.4.2 on page 5-8	Only one select signal can be active during a transfer.
ast_snps_apb_initial_idle	((AGENT_TYPE==snps_apb_aip_pkg::MONITOR    (AGENT_TYPE==snps_apb_aip_pkg::SLAVE && PSEL_WIDTH>1)) && ENABLE_ASSERT==1)	ERROR	[ARM IHI 0011A] Section 5.2.1 on page 5-4	IDLE The default state for the peripheral bus.
ast_snps_apb_idle_to_idle_or_setup	((AGENT_TYPE==snps_apb_aip_pkg::SLAVE    AGENT_TYPE==snps_apb_aip_pkg::MONITOR) && ENABLE_ASSERT==1) && (APB_VERSION==snps_apb_aip_pkg::APB2)	ERROR	[ARM IHI 0011A] Section 5.2.1 Figure 5-2 on page 5-4	Improper state transition from idle state.
ast_snps_apb_setup_to_access	((AGENT_TYPE==snps_apb_aip_pkg::SLAVE    AGENT_TYPE==snps_apb_aip_pkg::MONITOR) && ENABLE_ASSERT==1) && (APB_VERSION==snps_apb_aip_pkg::APB2)	ERROR	[ARM IHI 0011A] Section 5.2.1 Figure 5-2 on page 5-4	Improper state transition from setup state.
ast_snps_apb_access_to_idle_or_setup	((AGENT_TYPE==snps_apb_aip_pkg::SLAVE    AGENT_TYPE==snps_apb_aip_pkg::MONITOR) && ENABLE_ASSERT==1) && (APB_VERSION==snps_apb_aip_pkg::APB2)	ERROR	[ARM IHI 0011A] Section 5.2.1 Figure 5-2 on page 5-4	Improper state transition from access state
ast_snps_apb_psel_stable_during_access	((AGENT_TYPE==snps_apb_aip_pkg::SLAVE    AGENT_TYPE==snps_apb_aip_pkg::MONITOR) && ENABLE_ASSERT==1) && (APB_VERSION==snps_apb_aip_pkg::APB2)	ERROR	[ARM IHI 0011A] Section 5.2.1 on page 5-5	The address, write, select, and write data signals must remain stable during the transition from the SETUP to ACCESS state.



**Table 4-3 APB Properties**

Property Name	Parameters	Error Kind	Spec Reference	Property Description
ast_snps_apb_pwrite_stable_during_access	((AGENT_TYPE==snps_apb_aip_pkg::SLAVE    AGENT_TYPE==snps_apb_aip_pkg::MONITOR) && ENABLE_ASSERT==1) && (APB_VERSION==snps_apb_aip_pkg::APB2)	ERROR	[ARM IHI 0011A] Section 5.2.1 on page 5-5	The address, write, select, and write data signals must remain stable during the transition from the SETUP to ACCESS state.
ast_snps_apb_paddr_stable_during_access	((AGENT_TYPE==snps_apb_aip_pkg::SLAVE    AGENT_TYPE==snps_apb_aip_pkg::MONITOR) && ENABLE_ASSERT==1) && (APB_VERSION==snps_apb_aip_pkg::APB2)	ERROR	[ARM IHI 0011A] Section 5.2.1 on page 5-5	The address, write, select, and write data signals must remain stable during the transition from the SETUP to ACCESS state.
ast_snps_apb_pwdata_stable_during_write_access	((AGENT_TYPE==snps_apb_aip_pkg::SLAVE    AGENT_TYPE==snps_apb_aip_pkg::MONITOR) && ENABLE_ASSERT==1) && (APB_VERSION==snps_apb_aip_pkg::APB2)	ERROR	[ARM IHI 0011A] Section 5.2.1 on page 5-5	The address, write, select, and write data signals must remain stable during the transition from the SETUP to ACCESS state.
ast_snps_apb_idle_to_idle_or_setup	((AGENT_TYPE==snps_apb_aip_pkg::SLAVE    AGENT_TYPE==snps_apb_aip_pkg::MONITOR) && ENABLE_ASSERT==1) && (APB_VERSION!=snps_apb_aip_pkg::APB2)	ERROR	[ARM IHI 0024B] Section 3.1 Figure 3-1 on page 3-2	Improper state transition from idle state.
ast_snps_apb_setup_to_access	((AGENT_TYPE==snps_apb_aip_pkg::SLAVE    AGENT_TYPE==snps_apb_aip_pkg::MONITOR) && ENABLE_ASSERT==1) && (APB_VERSION!=snps_apb_aip_pkg::APB2)	ERROR	[ARM IHI 0024B] Section 3.1 Figure 3-1 on page 3-2	Improper state transition from idle state.
ast_snps_apb_access_to_idle_or_setup_ready	((AGENT_TYPE==snps_apb_aip_pkg::SLAVE    AGENT_TYPE==snps_apb_aip_pkg::MONITOR) && ENABLE_ASSERT==1) && (APB_VERSION!=snps_apb_aip_pkg::APB2)	ERROR	[ARM IHI 0024B] Section 3.1 Figure 3-1 on page 3-2	Improper state transition from idle state.

**Table 4-3 APB Properties**

Property Name	Parameters	Error Kind	Spec Reference	Property Description
ast_snps_apb_access_to_access	((AGENT_TYPE==snps_apb_aip_pkg::SLAVE    AGENT_TYPE==snps_apb_aip_pkg::MONITOR) && ENABLE_ASSERT==1) && (APB_VERSION!=snps_apb_aip_pkg::APB2)	ERROR	[ARM IHI 0024B] Section 3.1 Figure 3-1 on page 3-2	Improper state transition from idle state.
ast_snps_apb_psel_stable_during_access	((AGENT_TYPE==snps_apb_aip_pkg::SLAVE    AGENT_TYPE==snps_apb_aip_pkg::MONITOR) && ENABLE_ASSERT==1) && (APB_VERSION!=snps_apb_aip_pkg::APB2)	ERROR	[ARM IHI 0024B] Section 3.1 on page 3-2	The address, write, select, and write data signals must remain stable during the transition from the SETUP to ACCESS state.
ast_snps_apb_pwrite_stable_during_access	((AGENT_TYPE==snps_apb_aip_pkg::SLAVE    AGENT_TYPE==snps_apb_aip_pkg::MONITOR) && ENABLE_ASSERT==1) && (APB_VERSION!=snps_apb_aip_pkg::APB2)	ERROR	[ARM IHI 0024B] Section 3.1 on page 3-2	The address, write, select, and write data signals must remain stable during the transition from the SETUP to ACCESS state.
ast_snps_apb_paddr_stable_during_access	((AGENT_TYPE==snps_apb_aip_pkg::SLAVE    AGENT_TYPE==snps_apb_aip_pkg::MONITOR) && ENABLE_ASSERT==1) && (APB_VERSION!=snps_apb_aip_pkg::APB2)	ERROR	[ARM IHI 0024B] Section 3.1 on page 3-2	The address, write, select, and write data signals must remain stable during the transition from the SETUP to ACCESS state.
ast_snps_apb_pwdata_stable_during_write_access	((AGENT_TYPE==snps_apb_aip_pkg::SLAVE    AGENT_TYPE==snps_apb_aip_pkg::MONITOR) && ENABLE_ASSERT==1) && (APB_VERSION!=snps_apb_aip_pkg::APB2)	ERROR	[ARM IHI 0024B] Section 3.1 on page 3-2	The address, write, select, and write data signals must remain stable during the transition from the SETUP to ACCESS state.
ast_snps_apb_pprot_stable_during_access	((AGENT_TYPE==snps_apb_aip_pkg::SLAVE    AGENT_TYPE==snps_apb_aip_pkg::MONITOR) && ENABLE_ASSERT==1) && (APB_VERSION==snps_apb_aip_pkg::APB4)	ERROR	[ARM IHI 0024C] Section 4.1 on page 4-2	The address, write, select, and write data signals must remain stable during the transition from the SETUP to ACCESS state.

**Table 4-3 APB Properties**

Property Name	Parameters	Error Kind	Spec Reference	Property Description
ast_snps_apb_pstrb_stable_during_write_access	((AGENT_TYPE==snps_apb_aip_pkg::SLAVE    AGENT_TYPE==snps_apb_aip_pkg::MONITOR) && ENABLE_ASSERT==1) && (APB_VERSION==snps_apb_aip_pkg::APB4)	ERROR	[ARM IHI 0024C] Section 4.1 on page 4-2	The address, write, select, and write data signals must remain stable during the transition from the SETUP to ACCESS state.
ast_snps_apb_pstrb_zero_during_read_access	((AGENT_TYPE==snps_apb_aip_pkg::SLAVE    AGENT_TYPE==snps_apb_aip_pkg::MONITOR) && ENABLE_ASSERT==1) && (APB_VERSION==snps_apb_aip_pkg::APB4)	ERROR	[ARM IHI 0024C] Section 2.1 on page 2-2	Write strobes must not be active during a read transfer.
ast_snps_apb_paddr_idle_stable	((AGENT_TYPE==snps_apb_aip_pkg::SLAVE    AGENT_TYPE==snps_apb_aip_pkg::MONITOR) && ENABLE_ASSERT==1) && (CONFIG_LOWPOWER==1 && PSEL_WIDTH>1)	WARNING	[ARM IHI 0024C] Section 3.1.2 on page 3-3	It is recommended that the address and write signals are not changed immediately after a transfer, but remain stable until another access occurs. This reduces power consumption.
ast_snps_apb_pwrite_idle_stable	((AGENT_TYPE==snps_apb_aip_pkg::SLAVE    AGENT_TYPE==snps_apb_aip_pkg::MONITOR) && ENABLE_ASSERT==1) && (CONFIG_LOWPOWER==1 && PSEL_WIDTH>1)	WARNING	[ARM IHI 0024C] Section 3.1.2 on page 3-3	It is recommended that the address and write signals are not changed immediately after a transfer, but remain stable until another access occurs. This reduces power consumption.
ast_snps_apb_psel_not_unknown	((AGENT_TYPE==snps_apb_aip_pkg::SLAVE    AGENT_TYPE==snps_apb_aip_pkg::MONITOR) && ENABLE_ASSERT==1) && (CONFIG_X_CHECK==1)	ERROR	AMBA Specification	PSEL signal should not include unknown bit.
ast_snps_apb_penable_not_unknown	((AGENT_TYPE==snps_apb_aip_pkg::SLAVE    AGENT_TYPE==snps_apb_aip_pkg::MONITOR) && ENABLE_ASSERT==1) && (CONFIG_X_CHECK==1)	ERROR	AMBA Specification	PENABLE signal should not be unknown.

**Table 4-3 APB Properties**

Property Name	Parameters	Error Kind	Spec Reference	Property Description
ast_snps_apb_pwrite_not_unknown	((AGENT_TYPE==snps_apb_aip_pkg::SLAVE    AGENT_TYPE==snps_apb_aip_pkg::MONITOR) && ENABLE_ASSERT==1) && (CONFIG_X_CHECK==1)	ERROR	AMBA Specification	PWRITE signal should not be unknown.
ast_snps_apb_paddr_not_unknown	((AGENT_TYPE==snps_apb_aip_pkg::SLAVE    AGENT_TYPE==snps_apb_aip_pkg::MONITOR) && ENABLE_ASSERT==1) && (CONFIG_X_CHECK==1)	ERROR	AMBA Specification	PADDR signal should not include unknown bit.
ast_snps_apb_pwdata_not_unknown	((AGENT_TYPE==snps_apb_aip_pkg::SLAVE    AGENT_TYPE==snps_apb_aip_pkg::MONITOR) && ENABLE_ASSERT==1) && (CONFIG_X_CHECK==1)	ERROR	AMBA Specification	PWDATA signal should not include unknown bit.
ast_snps_apb_pprot_not_unknown	((AGENT_TYPE==snps_apb_aip_pkg::SLAVE    AGENT_TYPE==snps_apb_aip_pkg::MONITOR) && ENABLE_ASSERT==1) && (CONFIG_X_CHECK==1) && (APB_VERSION==snps_apb_aip_pkg::APB4)	ERROR	AMBA Specification	PPROT signal should not include unknown bit.
ast_snps_apb_pstrb_not_unknown	((AGENT_TYPE==snps_apb_aip_pkg::SLAVE    AGENT_TYPE==snps_apb_aip_pkg::MONITOR) && ENABLE_ASSERT==1) && (CONFIG_X_CHECK==1) && (APB_VERSION==snps_apb_aip_pkg::APB4)	ERROR	AMBA Specification	PSTRB signal should not include unknown bit.
ast_snps_apb_pslverr_allowed	((AGENT_TYPE==snps_apb_aip_pkg::MASTER    AGENT_TYPE==snps_apb_aip_pkg::MONITOR) && ENABLE_ASSERT==1) && (APB_VERSION!=snps_apb_aip_pkg::APB2 && PSLVERR_RCMND==1)	WARNING	[ARM IHI 0024C] Section 3.4 on page 3-6	It is recommended, but not mandatory, that you drive PSLVERR LOW when it is not being sampled. That is, when any of PSEL, PENABLE, or PREADY are LOW.

**Table 4-3 APB Properties**

Property Name	Parameters	Error Kind	Spec Reference	Property Description
ast_snps_apb_prdata_not_unknown	((AGENT_TYPE==snps_apb_aip_pkg::MASTER    AGENT_TYPE==snps_apb_aip_pkg::MONITOR) && ENABLE_ASSERT==1) && (CONFIG_X_CHECK==1)	ERROR	AMBA Specification	PRDATA signal should not include unknown bit.
ast_snps_apb_pready_not_unknown	((AGENT_TYPE==snps_apb_aip_pkg::MASTER    AGENT_TYPE==snps_apb_aip_pkg::MONITOR) && ENABLE_ASSERT==1) && (CONFIG_X_CHECK==1) && (APB_VERSION!=snps_apb_aip_pkg::APB2)	ERROR	AMBA Specification	PREADY signal should not be unknown.
ast_snps_apb_pslverr_not_unknown	((AGENT_TYPE==snps_apb_aip_pkg::MASTER    AGENT_TYPE==snps_apb_aip_pkg::MONITOR) && ENABLE_ASSERT==1) && (CONFIG_X_CHECK==1) && (APB_VERSION!=snps_apb_aip_pkg::APB2)	ERROR	AMBA Specification	PSLVERR signal should not be unknown.
ast_snps_apb_pready_max_waits	((AGENT_TYPE==snps_apb_aip_pkg::MASTER    AGENT_TYPE==snps_apb_aip_pkg::MONITOR) && ENABLE_ASSERT==1 && CHECK_MAX_WAITS==1 && APB_VERSION!=snps_apb_aip_pkg::APB2)	ERROR	AMBA Specification	PREADY should be returned within specified cycles.

**Table 4-4 The APB AIP Cover Properties**

Property Name	Parameter
cov_snps_apb_psel_selected_read	(ENABLE_COVER==1)
cov_snps_apb_psel_selected_write	(ENABLE_COVER==1)
cov_snps_apb_read_setup	(ENABLE_COVER==1)
cov_snps_apb_write_setup	(ENABLE_COVER==1)
cov_snps_apb_read_access	(ENABLE_COVER==1) && (APB_VERSION==snps_apb_aip_pkg::APB2)
cov_snps_apb_write_access	(ENABLE_COVER==1) && (APB_VERSION==snps_apb_aip_pkg::APB2)
cov_snps_apb_read_access_wait	(ENABLE_COVER==1) && (APB_VERSION==snps_apb_aip_pkg::APB3)
cov_snps_apb_read_access_okay	(ENABLE_COVER==1) && (APB_VERSION==snps_apb_aip_pkg::APB3)
cov_snps_apb_read_access_error	(ENABLE_COVER==1) && (APB_VERSION==snps_apb_aip_pkg::APB3)
cov_snps_apb_write_access_wait	(ENABLE_COVER==1) && (APB_VERSION==snps_apb_aip_pkg::APB3)
cov_snps_apb_write_access_okay	(ENABLE_COVER==1) && (APB_VERSION==snps_apb_aip_pkg::APB3)
cov_snps_apb_write_access_error	(ENABLE_COVER==1) && (APB_VERSION==snps_apb_aip_pkg::APB3)
cov_snps_apb_pprot_normal_access	(ENABLE_COVER==1) && (APB_VERSION==snps_apb_aip_pkg::APB4)
cov_snps_apb_pprot_privileged_access	(ENABLE_COVER==1) && (APB_VERSION==snps_apb_aip_pkg::APB4)
cov_snps_apb_pprot_secure_access	(ENABLE_COVER==1) && (APB_VERSION==snps_apb_aip_pkg::APB4)
cov_snps_apb_pprot_non_secure_access	(ENABLE_COVER==1) && (APB_VERSION==snps_apb_aip_pkg::APB4)
cov_snps_apb_pprot_data_access	(ENABLE_COVER==1) && (APB_VERSION==snps_apb_aip_pkg::APB4)
cov_snps_apb_pprot_instruction_access	(ENABLE_COVER==1) && (APB_VERSION==snps_apb_aip_pkg::APB4)

## 4.4 Behavior of Properties

This section describes the behavior of all properties of the APB AIP, when acting as assert, assume, or cover.

### 4.4.1 Properties In Assert Directives

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

### 4.4.2 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 VC Formal tool generates random stimulus for the input signals.
- When AGENT\_TYPE is SLAVE, the checkers mentioned as 'Master' in Master/Slave column of [Table 4-3](#) are declared as assert, and checkers mentioned as 'Slave' in Master/Slave column are declared as assume.

### 4.4.3 Properties In Cover Directives

- Cover properties indicate what kind of transactions or scenarios are exercised during a formal run.
- Cover properties are useful in checking whether the environment is over-constraint and if a design issues all possible transactions.
- To enable cover properties, set ENABLE\_COVER to 1.
- Note that the cover properties are independent from the properties used as assert/assume.





# The APB AIP Use Cases

This chapter discusses about the setup of AMBA2 and AMBA3 APB AIP in an AIP-DUT environment.

## 5.1 The APB AIP Examples

This section describes the setup of the APB AIP where the APB AIP is connected with the DUT.

- For connecting the ports of the APB DUT with the APB AIP, create a bind file to include the instance of the APB AIP into the top level module of the DUT.
- In the APB AIP, there is no limitation on the number of slaves that can be configured. You can configure the PSEL\_WIDTH parameter as per the requirement.
- Connect the APB AIP input/output signal to the DUT input and output signal such that the input to the DUT is constraint by the assume property and output from the DUT is checked by the assert property. The assume and assert properties can be enabled using `ENABLE_ASSUME` and `ENABLE_ASSERT`.



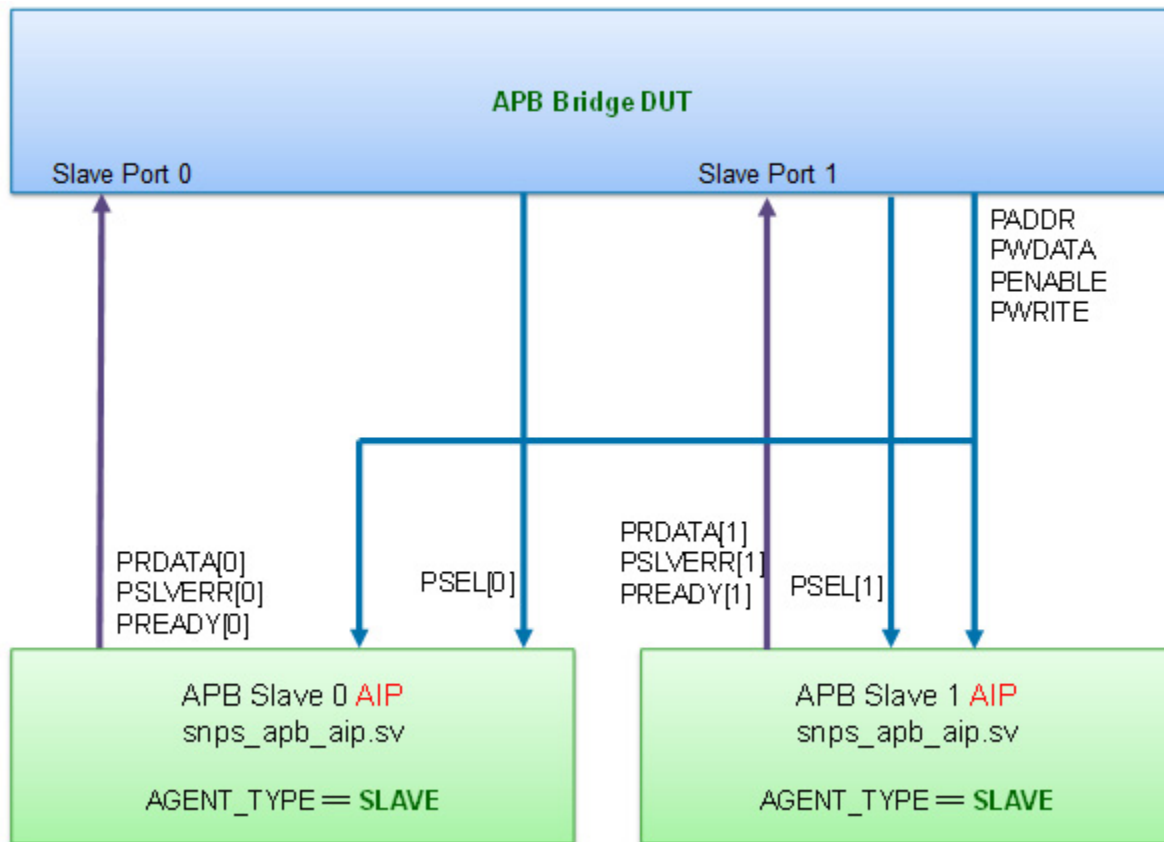
### Note

If a signal corresponding to the APB 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 `pslverr` signal, set with `1'b0`.

### 5.1.1 The APB AIP with a Master DUT

[Figure 5-1](#) shows the example setup for multiple APB AIPs in SLAVE mode connected with the APB bridge DUT. The APB AIP bridge DUT is connected with the two APB AIPs in the SLAVE mode. [Figure 5-1](#) shows the AIP-DUT setup. [Table 5-1](#) shows the bind file for the required setup.

Figure 5-1 Master DUT With the Slave APB AIP



**Table 5-1 Master DUT With Slave AIP Bind**

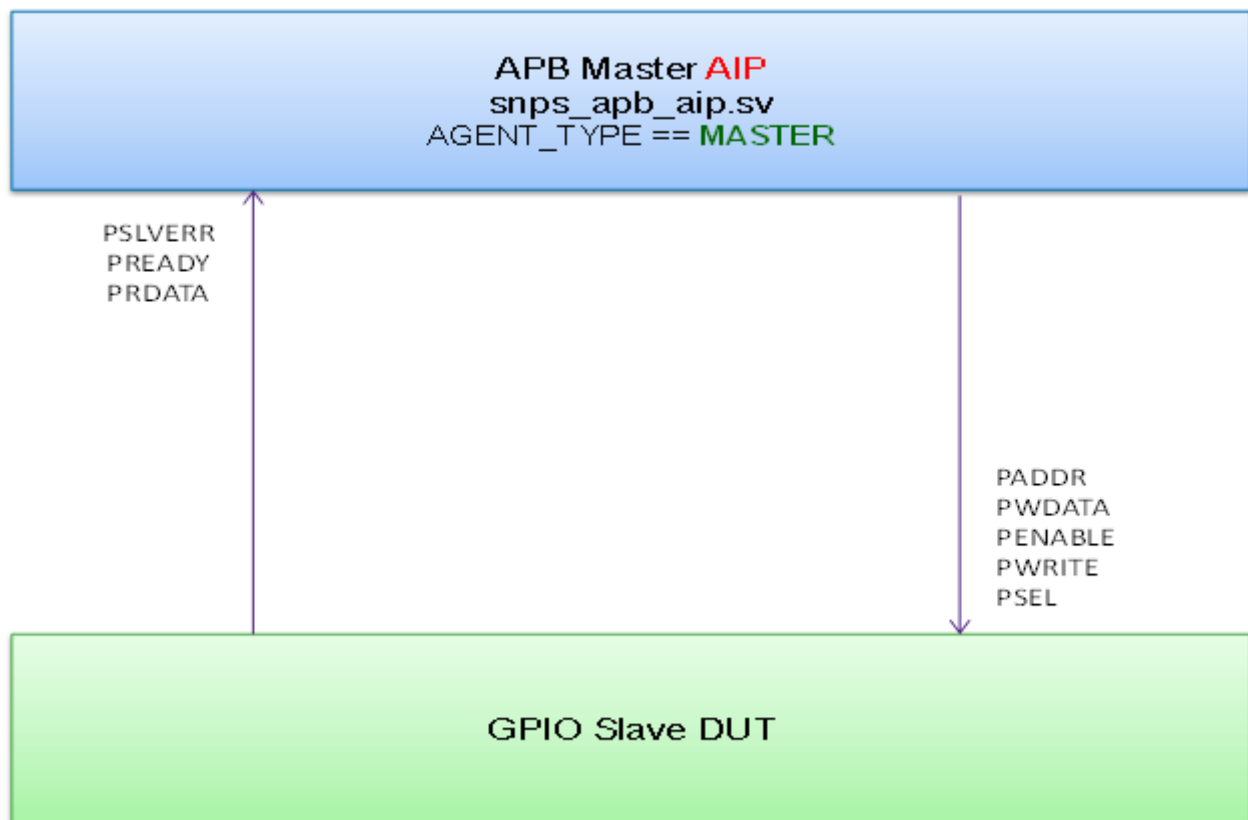
<pre> -----BIND SLAVE 0----- bind DW_apb  snps_apb_aip //AIP parameters #( .PROTOCOL_VER      (AMBA2)   , .AGENT_TYPE        (SLAVE)   , .ENABLE_ASSERT     (1)   , .ENABLE_ASSUME     (1)   , .ENABLE_COVER      (1)   , .PSEL_WIDTH         (1)   , .ADDR_WIDTH         (32)   , .WDATA_WIDTH        (32)   , .RDATA_WIDTH        (32)   , .CHECK_FORMAL       (1)   , .CONFIG_LOWPOWER    (1)   , .CONFIG_X_CHECK     (0)   , .PSLVERR_RCMND      (0)) u_snps_apb_aip0 //AIP-DUT Port Connections ( .pclk      (pclk_en)   , .presetn (hresetn)   , .pselx    (psel_s0)   , .penable  (penable)   , .pwrite   (pwrite)   , .paddr    (paddr)   , .pwwdata  (pwwdata)   , .prdata   (prdata_s0)   , .pready   (1)   , .pslverr  (0) ); </pre>	<pre> -----BIND SLAVE 1----- bind DW_apb  snps_apb_aip //AIP parameters #( .PROTOCOL_VER      (AMBA2)   , .AGENT_TYPE        (SLAVE)   , .ENABLE_ASSERT     (1)   , .ENABLE_ASSUME     (1)   , .ENABLE_COVER      (1)   , .PSEL_WIDTH         (1)   , .ADDR_WIDTH         (32)   , .WDATA_WIDTH        (32)   , .RDATA_WIDTH        (32)   , .CHECK_FORMAL       (1)   , .CONFIG_LOWPOWER    (1)   , .CONFIG_X_CHECK     (0)   , .PSLVERR_RCMND      (0)) u_snps_apb_aip1 //AIP-DUT Port Connections ( .pclk      (pclk_en)   , .presetn (hresetn)   , .pselx    (psel_s1)   , .penable  (penable)   , .pwrite   (pwrite)   , .paddr    (paddr)   , .pwwdata  (pwwdata)   , .prdata   (prdata_s1)   , .pready   (1)   , .pslverr  (0) ); </pre>
---	---

### 5.1.2 The APB AIP With a Slave DUT

Figure 5-2 shows the example for connecting the APB AIP MASTER with the GPIO slave DUT. The GPIO slave DUT is connected with the APB AIP in MASTER mode. Figure 5-2 shows the AIP-DUT setup.

Table 5-2 shows the bind file for the required setup.

Figure 5-2 Slave DUT With the Master APB AIP



**Table 5-2 Slave DUT With the Master APB AIP Bind**

```
---MASTER AIP WITH  SLAVE GPIO DUT BIND---

bind dw_apb_gpio  snps_apb_aip
//AIP parameters
#( .PROTOCOL_VER      (AMBA2)
  , .AGENT_TYPE       (MASTER)
  , .ENABLE_ASSERT    (1)
  , .ENABLE_ASSUME    (1)
  , .ENABLE_COVER     (1)
  , .PSEL_WIDTH       (1)
  , .ADDR_WIDTH       (32)
  , .WDATA_WIDTH      (32)
  , .RDATA_WIDTH      (32)
  , .CHECK_FORMAL     (1)
  , .CONFIG_LOWPOWER  (1)
  , .CONFIG_X_CHECK   (0)
  , .PSLVERR_RCMND    (0))
u_snps_apb_aip
//AIP-DUT Port Connections
( .pclk      (pclk)
  , .presetn (presetn)
  , .pselx   (psel)
  , .penable (penable)
  , .pwrite  (pwrite)
  , .paddr   (paddr)
  , .pwwdata (pwwdata)
  , .prdata  (prdata)
  , .pready  (1)
  , .pslverr (0)
);
```