

AXI ABVIP User Guide

Product Version 11.3

July 2022

© 2022 Cadence Design Systems, Inc. All rights reserved.
Printed in the United States of America.

Cadence Design Systems, Inc., 555 River Oaks Parkway, San Jose, CA 95134, USA

Trademarks: Trademarks and service marks of Cadence Design Systems, Inc. (Cadence) contained in this document are attributed to Cadence with the appropriate symbol. For queries regarding Cadence's trademarks, contact the corporate legal department at the address shown above or call 800.862.4522.

Open SystemC, Open SystemC Initiative, OSCI, SystemC, and SystemC Initiative are trademarks or registered trademarks of Open SystemC Initiative, Inc. in the United States and other countries and are used with permission.

All other trademarks are the property of their respective holders.

Restricted Print Permission: This publication is protected by copyright and any unauthorized use of this publication may violate copyright, trademark, and other laws. Except as specified in this permission statement, this publication may not be copied, reproduced, modified, published, uploaded, posted, transmitted, or distributed in any way, without prior written permission from Cadence. This statement grants you permission to print one (1) hard copy of this publication subject to the following conditions:

1. The publication may be used solely for personal, informational, and noncommercial purposes;
2. The publication may not be modified in any way;
3. Any copy of the publication or portion thereof must include all original copyright, trademark, and other proprietary notices and this permission statement; and
4. Cadence reserves the right to revoke this authorization at any time, and any such use shall be discontinued immediately upon written notice from Cadence.

Patents: Cadence Product AXI Assertion-Based VIP, described in this document, is protected by U.S. Patents 5,095,454, 5,418,931, 5,606,698, 6,487,704, 7,039,887, 7,055,116, 5,838,949, 6,263,301, 6,163,763, 6,301,578

Disclaimer: Information in this publication is subject to change without notice and does not represent a commitment on the part of Cadence. The information contained herein is the proprietary and confidential information of Cadence or its licensors, and is supplied subject to, and may be used only by Cadence's customer in accordance with, a written agreement between Cadence and its customer. Except as may be explicitly set forth in such agreement, Cadence does not make, and expressly disclaims, any representations or warranties as to the completeness, accuracy or usefulness of the information contained in this document. Cadence does not warrant that use of such information will not infringe any third party rights, nor does Cadence assume any liability for

damages or costs of any kind that may result from use of such information.

Restricted Rights: Use, duplication, or disclosure by the Government is subject to restrictions as set forth in FAR52.227-14 and DFAR252.227-7013 et seq. or its successor.

Contents

1	8
Introduction	8
Checking Version Compatibility	9
Usage	9
Getting Started	9
Setup Options	10
2	13
Configuring your ABVIP	13
Module Configuration	13
Master	13
Slave	14
Monitor	14
Configurable	15
Parameter Configuration	15
3	17
Methodology	17
Using ABVIP with Formal Analysis	17
Using Prove	19
Sanity of Assumptions	19
Waveform Groups	20
ABVIP Table Window	21
Liveness Properties in ABVIP	26
Using ABVIP with Simulation	28
Configuring ABVIP	29
Debugging ABVIP	29
4	30
Verification Plan	30
AXI3 vPlan	30
AXI4 vPlan	32
ACE vPlan	33
5	35

AXI3 Design Examples	35
Copying the Design Examples	35
Verifying the AXI3 Master using Formal Analysis	36
Running an Example	38
Verifying AXI3 Master Design	38
Verifying AXI Design	38
Verifying the AXI3 Slave using Formal Analysis	39
Running an Example	41
Verifying AXI3 Slave Design	42
Verifying the AXI3 Bridge using Formal Analysis	42
Running an Example	44
Verifying AXI3 Bridge Design	44
Verifying the AXI3 Slave using Simulation	44
Running an Example	47
Verifying AXI3 slave Design in Simulation	47
Verifying AXI Using Xcelium	47
6	49
AXI4 Design Examples	49
Copying the Design Examples	49
Verifying the AXI4 Master using Formal Analysis	50
Running an Example	52
Verifying AXI4 Master Design	52
Verifying AXI Design	53
Verifying the AXI4 Slave using Formal Analysis	53
Running an Example	55
Verifying AXI4 Slave Design	56
Verifying the AXI4 Bridge using Formal Analysis	56
Running an Example	58
Verifying AXI4 Bridge Design	58
Verifying the AXI4 Slave using Simulation	58
Running an Example	60
Verifying AXI4 slave Design in Simulation	61
Verifying AXI4 Using Xcelium	61
7	62
ACE Design Examples	62
Copying the Design Examples	62

Verifying the ACE Master using Formal Analysis	63
Running an Example	65
Verifying ACE Master Design	65
Verifying ACE Design	65
Verifying the ACE Bridge using Formal Analysis	66
Running an Example	68
Verifying ACE Bridge Design	68
Verifying the ACE-LITE Master using Formal Analysis	69
Running an Example	71
Verifying ACE Master Design	71
ABVIP_SIM : Running Formal Setup in Simulation	71
Running using ABVIP_SIM option	72
Verifying AXI4_ACE Using Xcelium	72
8	74
AXI5 Design Examples	74
Copying the Design Examples	74
Verifying the AXI5 Slave using Formal Verification (Jasper)	75
Running an Example	75
Verifying AXI5 Master Design	75
Verifying the AXI5-G Slave using Formal Verification (Jasper)	75
Running an Example	76
Verifying AXI5-G Master Design	76
Verifying AXI5 Using Xcelium	76
9	77
ACE5 Design Examples	77
Copying the Design Examples	77
Verifying the ACE5 Slave using Formal Verification (Jasper)	78
Running an Example	78
Verifying ACE5 Master Design	78
Verifying the ACE5-G Slave using Formal Verification (Jasper)	78
Running an Example	79
Verifying ACE5-G Master Design	79
Verifying ACE5 Using Xcelium	79
Appendix A: AXI3 ABVIP Package Content	80
Package Contents	80
AXI3 Monitor Pin-Level Interface	82


AXI3 ABVIP Debug Signals	84
AXI3 Monitor Parameters	85
AXI3 Checks	91
Compliance Checks	91
Coverage Checks	113
Appendix B: AXI4 ABVIP Package Contents	122
Package Contents	122
AXI4 Monitor Pin-Level Interface	125
AXI4 ABVIP Debug Signals	127
AXI4 Monitor Parameters	128
AXI4 Checks	136
Compliance Checks	136
Coverage Checks	159
Appendix C: ACE ABVIP Package Contents	169
Package Contents	169
ACE Monitor Pin-Level Interface	171
ACE Monitor Parameters	173
ACE Checks	180
Compliance Checks	180
Coverage Checks	210
Appendix D: AXI5 ABVIP Package Contents	217
Package Contents	217
AXI5 ABVIP Pin-Level Interface	218
AXI5 ABVIP Parameters	223
AXI5 Checks	231
Compliance Checks	231
Coverage Checks	260
Appendix E: ACE5 ABVIP Package Contents	268
Package Contents	268
ACE5 ABVIP Pin-Level Interface	271
ACE5 Monitor Parameters	277
ACE5 Checks	289
Compliance Checks	289
Coverage Checks	351

Introduction

- [Checking Version Compatibility](#)
 - [Usage](#)
- [Getting Started](#)
 - [Setup Options](#)
 - [Instantiating ABVIP Components in a Top-Level Module](#)
 - [Connecting AXI ABVIP with DUV using bind Directive](#)

Verifying a protocol and analyzing its completeness is a key challenge for engineers these days. Cadence presents the Assertion Based VIP (ABVIP) for ARM AMBA AXI protocol verification. In this ABVIP, protocol specifications have been comprehensively captured to verify your design. At this point of time, AXI3, AXI4, ACE, AXI4 ACE-CACHE, and ACE-LITE ABVIPs are supported. The ABVIPs are a comprehensive set of checkers and RTL that check for protocol compliance of ARM AMBA AXI based systems and designs.

The AXI3, AXI4, ACE, AXI4 ACE-CACHE, and ACE-LITE ABVIPs consist of RTL written in SystemVerilog and assertions written in SVA. They also have a comprehensive set of covers for measuring functional coverage of verification. The AXI ABVIPs are used both in formal analysis and simulation environment. In formal analysis, the ABVIP can either be configured as a master and the slave device can be verified or the ABVIP can be configured as a slave and a master device can be verified. In case you want to verify a complete master-slave system, you can configure ABVIP as a monitor/checker. In simulation environment, this ABVIP can only be used as a checker.

 In accordance with AMBA Progressive Terminology Update v3.0, the terminology used in ABVIP is being updated to remove or replace offensive terminology.

In the source code of AXI and ACE ABVIP, except for AXI3, the 'Master' is changed to 'Manager' and 'Slave' is changed to 'Subordinate'. This change is done across parameter, property, and example names.

The ABVIP with progressive terminology is available at
`<VIPCAT_INSTALL>/progressive_terminology/<protocol>/rtl/`

You can continue to use this user guide as a reference for ABVIP with progressive terminology. While reading, infer terms, such as "master" and "slave" as "manager" and "subordinate" respectively.

Checking Version Compatibility

A version compatibility script, `abvip_ver_check.pl`, has been included in the release to check the compatibility of the INCISIV release in your path with a specified ABVIP. This script is located at:

`<VIPCAT_INSTALL_DIR>/tools/abvip/utls/abvip_ver_check.pl`

Usage

`abvip_ver_check.pl -abvip <ABVIP_protocol>`

This prints whether INCISIV install specified is compatible with the specified ABVIP. For example:

`abvip_ver_check.pl -abvip AXI3`

To print the help and valid options to this script, type `abvip_ver_check.pl -help`

Getting Started

The AXI3, AXI4, and ACE ABVIPs are designed to be highly configurable in order to control complexity during protocol verification. The ABVIP configuration is done through parameters while instantiating the ABVIP in a top level module or an external VUNIT file. These parameters then control complexity and an extent of verification. These parameters can also be used to configure the ABVIP according to the DUT being verified.

Setup Options

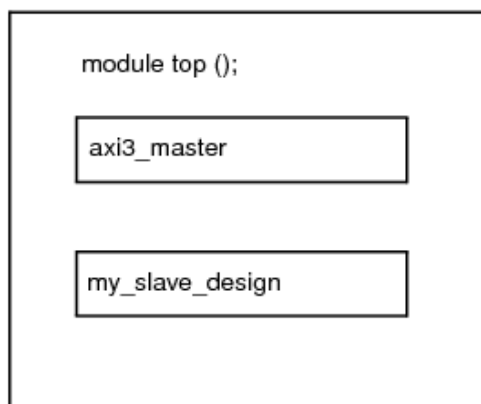
The various options of using an AXI Assertion-Based VIP with your design are:

- [Instantiating ABVIP Components in a Top-Level Module](#)
- [Instantiating AXI ABVIP in a Property File](#)
- [Connecting AXI ABVIP with DUV using bind Directive](#)

Instantiating ABVIP Components in a Top-Level Module

You can create a top-level module in which you can instantiate the AXI ABVIP as a master or a slave along with the design under verification(DUV). For example, if you are verifying an AXI3 slave design, instantiate the AXI3 ABVIP master within top module along with the slave DUV as shown in [Figure 1-1](#).

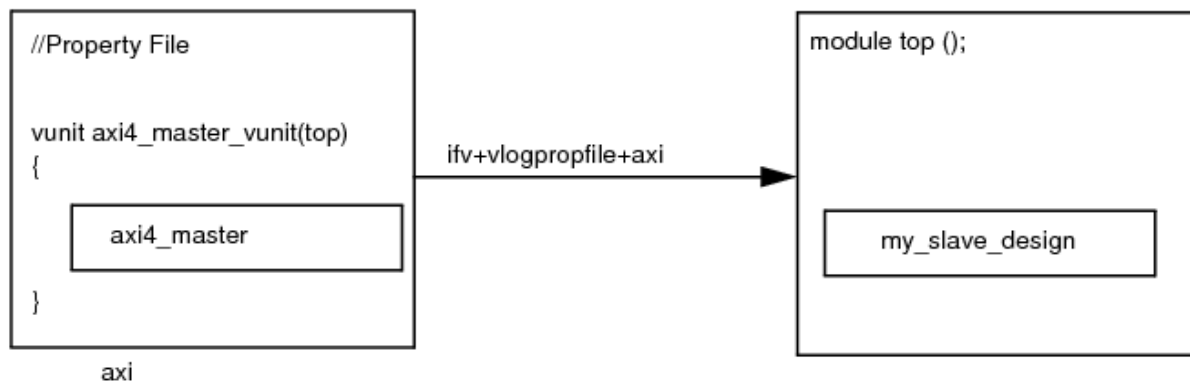
Figure 1-1 Instantiating the AXI ABVIP Master in a Top-Level Module



Instantiating AXI ABVIP in a Property File

You can instantiate an AXI ABVIP as a master or slave in a property file and associate the property file with your top level module. You can specify this property file at compile time, as illustrated in [Figure 1-2](#).

Figure 1-2 Instantiating AXI ABVIP Master in a Property File



There is no need to edit your source code when using this method. However, you need to create verification units for the AXI ABVIP instances.

Connecting AXI ABVIP with DUV using bind Directive

You can create an additional top entity or module that will bind the AXI ABVIP with design top level module, as shown in an example given below:

```
module bind_info ();
bind top cdn_abvip_axi3_slave #
(
    .ADDR_WIDTH (ADDR_WIDTH),
    .DATA_WIDTH (DATA_WIDTH),
    .ID_WIDTH (ID_WIDTH),
    .LEN_WIDTH (LEN_WIDTH),
    .MAX_PENDING (MAX_PENDING),
    .READ_INTERLEAVE_ON (0),
    .BYTE_STROBE_ON (0),
    .EXCL_ACCESS_ON (0),
    .LOCKED_ACCESS_ON (0),
    .MAX_WAIT_CYCLES_ON (0)

)
axi_slave
(
    .aclk (aclk_m),
    .aresetn (aresetn_m),
    .awid (awid_m),
    .awaddr (awaddr_m),
    -----
    -----
    .csysreq (csysreq),
    .csysack (csysack),
    .cactive (cactive)
);

endmodule
```

In the above example, the AXI3 ABVIP module `cdn_abvip_axi3_slave` is bound to design top level module top using a bind directive.

To run the design, explicitly specify the top module having bind information using `-bind_top` option as:

```
irun -ifv -f irun.f bind_info.v -bind_top bind_info
```

The extra top modules specified with `-bind_top` should only have bind directives. For more information on binding properties using bind directive, refer to Chapter 12 of *Formal Verifier User Guide*.

Configuring your ABVIP

Module Configuration

AXI ABVIP can be configured in master, slave or monitor modes during instantiation. Each mode automatically infers a valid set of assertions and constraints. Each of these modes is explained in the following sub-sections.

- [Module Configuration](#)
 - [Master](#)
 - [Slave](#)
 - [Monitor](#)
 - [Configurable](#)
- [Parameter Configuration](#)

Master

This mode is only applicable to formal analysis. In this mode, all properties related to signals/ports driven by the master are inferred as constraints. All properties related to signals/ports driven by the slave are inferred as assertions. The addition or deletion of assertions and constraints is not required for verification. To configure the ABVIP in master mode, you have to choose one of the following modules as applicable for instantiation in the testbench or vunit:

- `cdn_abvip_axi3_master`
- `cdn_abvip_axi4_master`
- `cdn_abvip_axi5_master`
- `cdn_abvip_axi4_ace_master`
- `cdn_abvip_ace5_master`

- `cdn_abvip_axi4_ace_lite_master`
- `cdn_abvip_axi4_ace_lite_dvm_master`
- `cdn_abvip_ace5_lite_master`
- `cdn_abvip_ace5_lite_dvm_master`

Slave

This mode is also applicable only to formal analysis. In this mode, all properties related to signals/ports driven by the slave are inferred as constraints. All properties related to signals/ports driven by the master are inferred as assertions. The addition or deletion of assertions and constraints is not required for verification. To configure the ABVIP in slave mode, you have to choose one of the following modules as applicable for instantiation in the top level module or vunit:

- `cdn_abvip_axi3_slave`
- `cdn_abvip_axi4_slave`
- `cdn_abvip_axi5_slave`
- `cdn_abvip_axi4_ace_slave`
- `cdn_abvip_ace5_slave`
- `cdn_abvip_axi4_ace_lite_slave`
- `cdn_abvip_axi4_ace_lite_dvm_slave`
- `cdn_abvip_ace5_lite_slave`
- `cdn_abvip_ace5_lite_dvm_slave`

Monitor

This mode is applicable to both formal analysis and simulation. In this mode, the ABVIP works as a passive checker. All relevant properties are inferred as assertions. To configure the ABVIP in monitor mode, you have to choose one of the following modules as applicable for instantiation in the top level module or vunit:

- `cdn_abvip_axi3_monitor`

- `cdn_abvip_axi4_monitor`
- `cdn_abvip_axi4_lite_monitor`
- `cdn_abvip_axi5_monitor`
- `cdn_abvip_axi5_lite_monitor`
- `cdn_abvip_axi4_ace_monitor`
- `cdn_abvip_ace5_monitor`
- `cdn_abvip_axi4_ace_lite_monitor`
- `cdn_abvip_axi4_ace_lite_dvm_monitor`
- `cdn_abvip_ace5_lite_monitor`
- `cdn_abvip_ace5_lite_dvm_monitor`

Configurable

With the following abvip modules you can configure ABVIP in any of the modes above. This usage is applicable to both formal analysis and simulation depending upon the mode you configure. For configuration the following modules have additional parameter named `VERIFY_BLK`.

- `cdn_abvip_axi3`
- `cdn_abvip_axi4`
- `cdn_abvip_axi4_ace`
- `cdn_abvip_ace5`

Parameter Configuration

Configuring parameters enables you to:

- Model DUT behavior to be verified
- Control complexity of verification
- Check comprehensiveness of completed verification

Modeling DUT behavior to be verified is one of the key reasons for parameterization. For example,

if your DUT does not support out-of-order write response then there is a parameter to configure the ABVIP in that mode. AXI specification by default allows out-of-order write response and the ABVIP will follow the default specification unless configured otherwise.

Controlling complexity is another aspect of configuration. For example, verifying byte strobes for write data before write control is an expensive activity for the ABVIP and Formal Verification. Complexity hot spots in verification like these can be partitioned into use-cases for quickly finding bugs in other areas, before performing this use-case.

Comprehensiveness of completed verification is another aspect of configurability that drives parameterization of the ABVIP. For example, you can choose to extract functional coverage through covers specified in the ABVIP by switching on a parameter. You can choose to do it per instance or generically throughout all instances by passing a defparam option to both formal analysis and simulation.

A detailed description and purpose of the parameters for AXI3, AXI4, AXI5, ACE4 and ACE5 ABVIP have been elaborated in individual package contents sections.

Methodology

- [Using ABVIP with Formal Analysis](#)
 - [Using Prove](#)
 - [Sanity of Assumptions](#)
 - [Waveform Groups](#)
 - [ABVIP Table Window](#)
 - [ABVIP Table Columns](#)
 - [ABVIP Table Operations](#)
- [Liveness Properties in ABVIP](#)
- [Using ABVIP with Simulation](#)
 - [Configuring ABVIP](#)
 - [Debugging ABVIP](#)

AXI ABVIP is a verification IP package that checks for compliance to the AMBA AXI protocol on your design under verification (DUV). ABVIP contains complete and optimized constraint and assertion models in the form of pre-verified set of SVA properties and cover checks. They are used for interface monitoring in simulation and for exhaustive formal analysis of protocol compliance with formal analysis tools.

Using ABVIP with Formal Analysis

When implementing or designing AXI master or slave devices, you can use AXI ABVIP to verify protocol compliance of the master and slave either separately or together. The VIP provides constraints on the inputs to the device, and assertions check that the output is correct. Before you start verifying the design using formal, configure ABVIP according to the DUT behavior. You can review the list of parameters provided by the ABVIP and set them according to the behavior of the DUT.

To verify your DUT with the ABVIP, the following standard use-cases are recommended for DUT

behavior modeling:

- `RST_CHECKS_ON = 1`: It is recommended for formal analysis that the reset signal is de-asserted for optimal tool performance. Keeping this recommendation in view, the reset checks have been partitioned using a separate parameter to be verified separately in a use-case. Reset checks perform checking when reset is free to toggle. To enable this use case you must remove pin constraint from the reset signal and leave it free. For all other use-cases, this parameter should be set to 0 and reset signal must be pin constrained (asserted) to avoid false failures.
- `DATA_BEFORE_CONTROL_ON = 1` and `BYTE_STROBE_ON = 0`: This verifies the write channel excluding the checking for WSTRB signal.
The behavior of the master and slave in this use-case is:
 - Write data after write control
 - Write data with write control
 - Write data before write control (DBC).
- `DATA_BEFORE_CONTROL_ON = 1` and `BYTE_STROBE_ON = 1`: This verifies the write channel including the checking for WSTRB signal. In this use-case, the behavior of the master is:
 - Write data after write control
 - Write data with write controlThe behavior of the slave is:
 - Write data after write control
 - Write data with write control
 - Write data before write control (DBC)The checker is capable of monitoring write transactions from the DUT and check for accurate WSTRB values.
- `MAX_PENDING = N+1`: This use-case is to verify FIFO full scenarios in the DUT. Here the value of N is decided by the pipeline depth of the DUT. If there is a FIFO overflow scenario in the DUT, then this use-case will be able to find it out.

To proceed with formal analysis, use prove command.

Using Prove

The assumptions to be added and the assertions to be verified in the ABVIP are decided based on the configuration. Refer to Section, [Module Configuration](#). You do not have to add or delete any assertions or assumption in the ABVIP to run formal verification. However, you can add specific assertions or assumptions, if required for the verification.

The verification settings recommended with Jasper are:

```
set_engine_mode {Ht L B N}

set_prove_per_property_time_limit_factor 0

set_prove_per_property_time_limit 30s

prove -all
```

If there are properties with "undetermined" status, rerun these properties with increased effort as:

```
set_prove_per_property_time_limit 1m

prove -all
```

The engine setting can be customized using `set_engine_mode` command and the prove settings can be customized using `set_prove_*` commands. For more details on these commands, you can refer to *Jasper Command Reference Manual*.

Sanity of Assumptions

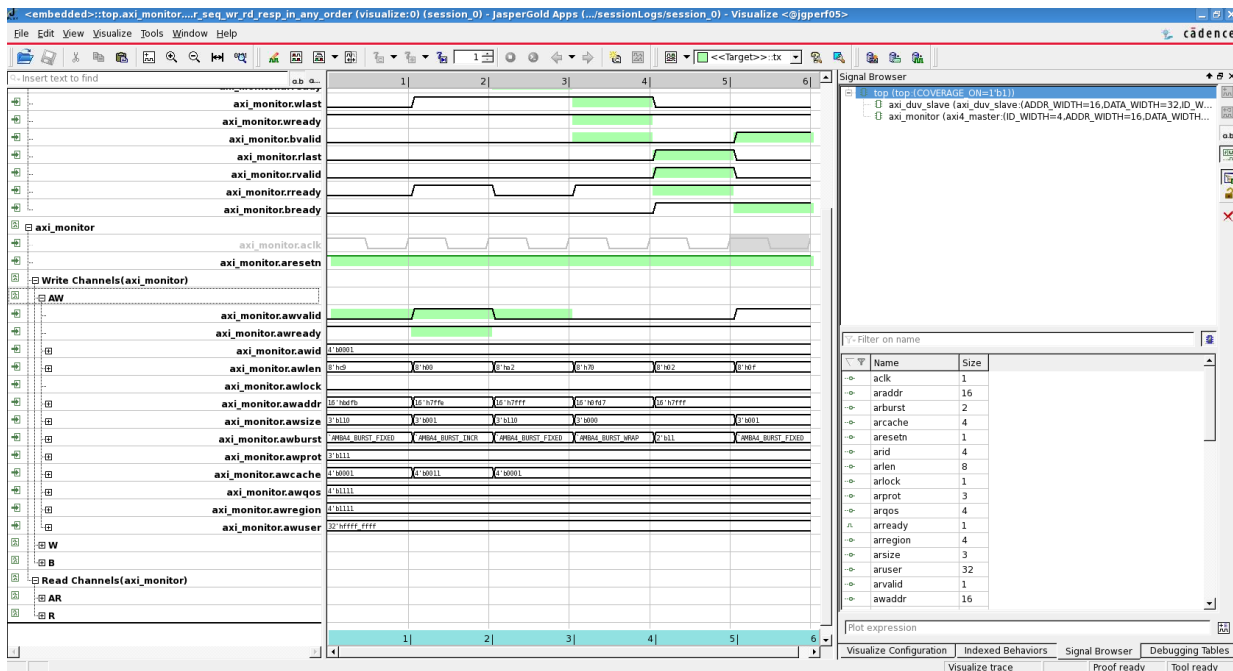
If there are any user defined assumptions in the DUT, a dead end/conflict may occur in the case of over-constrained environment. This over-constrained environment can potentially lead to a bug being missed. Therefore, it is important to sanitize the constraint environment before starting with prove.

The prove command will error out in such cases. However, before running the proof, the conflicts and dead ends can be identified using `check_assumptions` command. For further details on these commands, refer to *Jasper Command Reference Manual*.

To assist in debugging AXI traffic, Jasper and ABVIP will automatically create signal groups to organize relevant bus signals in an easy to understand grouping. These groups do not need to be created manually. In addition, an ABVIP table window is provided to visualize the transaction state across various ABVIP instances. Waveform groups and ABVIP table are explained as follows:

Waveform Groups

In order to efficiently debug AXI bus traffic it is imperative to organize bus signals for easy comprehension of bus traffic. The AXI ABVIP in conjunction with Jasper Visualize provides groups for easier debugging, as shown in the following figure:



Each group AW, W, B, AR, R, and (AC, CD, and CR for ACE) provide detail signal groupings for each of the AXI bus bundles. You can click and view the information associated with each group and mnemonics that help with debugging.

The description of each signal grouping is explained as:

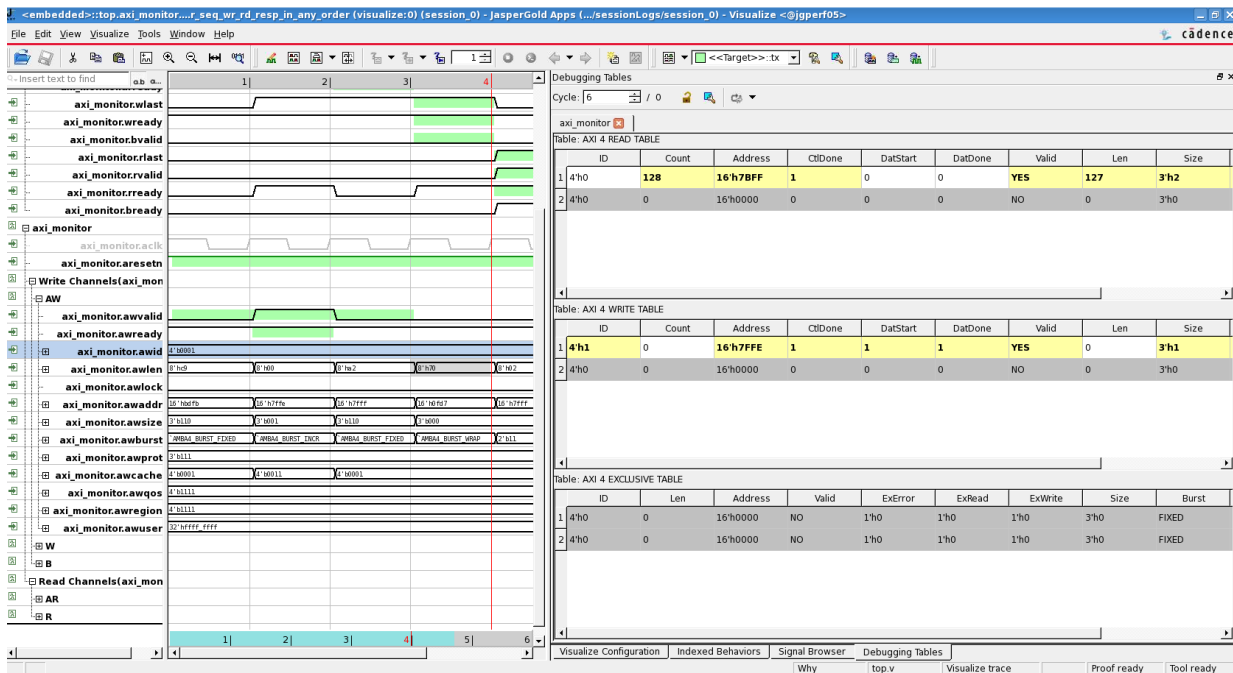
Group	Description
AR	Read Address Channel. All control signals related to the read address channel are visible in this group.
R	Read Data Channel. All signals related to the read data channel are visible in this group.
AW	Write Address Channel. All control signals related to the write address channel are visible in this group.

W	Write Data Channel. All signals related to the write data channel are visible in this group.
B	Write Response Channel. All signals related to the write response channel are visible in this group.
Specific to ACE	
AC	Snoop Address Channel. All control signals related to the snoop address channel are visible in this group.
CD	Snoop Data Channel. All signals related to the snoop data channel are visible in this group.
CR	Snoop Response Channel. All signals related to the snoop response channel are visible in this group.

ABVIP Table Window

The ABVIP table window presents a view of the transactions being processed by the ABVIP. This window is connected to the Visualize waveform window. From the bottom right corner, select "Debugging Tables" Pane to open the ABVIP table.

As you select the cycle number on the "Debugging Table" pane, the ABVIP table window will show the transaction progress being monitored by the table.



The rows in the register window for both WRITE TABLE and READ TABLE are determined by the parameter setting for MAX_PENDING. The rows in the register window for the EXCLUSIVE TABLE are determined by the parameter setting for MAX_PENDING_EXCL. It can be observed from the figure above that the parameters MAX_PENDING and MAX_PENDING_EXCL have been set to 2.

ABVIP Table Columns

The description for each of the columns in the WRITE, READ, and EXCLUSIVE TABLE is as follows:

Columns	Description
WRITE TABLE	
ID	ID of write transaction in progress.
Count	Number of data beats remaining in the write request (decremented with every WVALID & WREADY).
CtrlDone	Control phase of write transaction completed (AWVALID & AWREADY).

DatStart	Data phase of write transaction started (first beat of write data observed, WVALID && WREADY).
DatDone	Data phase of write transaction completed (last beat of write data observed, WVALID && WLAST && WREADY).
Valid	Write transaction is in progress (starting from AWVALID && AWREADY or WVALID && WREADY and completing with BVALID && BREADY).
Lock	Exclusive access control information of write transaction in progress.
Burst	Burst type of write transaction in progress
Size	Size of write transaction in progress
Address	Address of write transaction being monitored
Len	Length of write transaction

READ TABLE

ID	ID of read transaction in progress
Count	Number of data beats remaining in the read request (decremented with every RVALID && RREADY)
Valid	Read transaction is in progress (starting from ARVALID && ARREADY and completing with RVALID && RREADY && RLAST)
Lock	Exclusive access control information of read transaction in progress
CtrlDone	Control phase of write transaction completed (AWVALID && AWREADY).

DatStart	Data phase of write transaction started (first beat of write data observed, WVALID && WREADY).
DatDone	Data phase of write transaction completed (last beat of write data observed, WVALID && WLAST && WREADY).
Burst	Burst type of write transaction in progress
Size	Size of write transaction in progress
Address	Address of write transaction being monitored
Len	Length of write transaction

EXCLUSIVE TABLE

ID	Exclusive access ID in progress
Address	Exclusive address being monitored
Len	Length of exclusive access transaction
Size	Size of exclusive access transaction
ExError	An error in exclusive access after which monitoring of the exclusive address is stopped (The cycle in which this is set, observe, AR, R, and AW channels for exclusive access errors)
Valid	An exclusive access is in progress (starting from ARVALID && ARREADY && ARLOCK == "EXCL" on the read channel and completing with BVALID && BREADY for the correlated transaction on the write channel)
ExRead	Exclusive read request completed as indicated by ARVALID && ARREADY && ARLOCK == "EXCL"

ExWrite	Exclusive write request completed as indicated by AWVALID && AWREADY && AWLOCK == "EXCL"
Burst	Burst type of exclusive access transaction
Prot	Protection type of exclusive access transaction
Cache	Cache type of exclusive access transaction
SNOOP TABLE	
Snoop	Type of snoop transaction
Address	Address of snoop transaction being monitored
CtlDone	Control phase of snoop transaction completed (ACVALID && ACREADY).
RspDone	Response phase of snoop transaction completed (CRVALID && CRREADY)
DatStart	Data phase of snoop transaction started (First beat of snoop data observed, CDVALID && CDREADY)
DatDone	Data phase of snoop transaction completed (Last beat of snoop data observed, CDVALID && CDREADY && CDLAST)
CACHE TABLE	
Address	Address of cache line being monitored
State	State of cache line being monitored
BARRIER TABLE	
ID	ID of the barrier transaction in progress.
Barrier	Type of barrier transaction
Domain	Domain of barrier transaction

Prot	Protection type of barrier transaction
RDStart	Read barrier transaction started (ARVALID && ARREADY && ARBAR[0])
RDDone	Read barrier transaction completed
WrStart	Write barrier transaction started (AWVALID && AWREADY && AWBAR[0])
WrDone	Write barrier transaction completed

ABVIP Table Operations

Each table in the ABVIP table window shows the oldest active transaction at the top of each table. As transactions complete, they are retired and the transaction below will bubble up into the empty row. The oldest transaction always stays at the top of each table.

Liveness Properties in ABVIP


AXI/ACE (except AXI3) protocol family ABVIPs have liveness properties for following scenarios-

1. Each *valid should get *ready eventually -
 - a. This will translate into either fairness constraints (assumptions) or deadlock checks (assertions) based on ABVIP mode used.
 - b. For example - awvalid should get awready eventually is a fairness constraint in slave ABVIP and deadlock check in master/monitor ABVIP
 - c. Similarly, rvalid should get rready eventually is a fairness constraint in master ABVIP and deadlock check in slave/monitor ABVIP
2. Each transaction should eventually complete -
 - a. Scenarios-
 - a. arvalid should get rvalid & rl原因
 - b. Leading wvalid should get awvalid
 - c. Leading awvalid should get wvalid & wlast
 - d. awvalid should get bvalid
 - e. awvalid with atomic operation should get rvalid & rl原因

- f. acvalid should get crvalid
 - g. crvalid with data should get cdvalid
 - b. This will translate into either fairness constraints (assumptions) or deadlock checks (assertions) based on ABVIP mode used.
 - c. For example arvalid should get rvalid & rlack is a fairness constraint in slave ABVIP and deadlock check in master/monitor ABVIP
 - d. For these properties, we use counters to count the required number of responses and the property is of the format-
pending_ctr |-> ##[0:\$] response
 - e. However, the way Jasper treats liveness assumptions and assertions while determining deadlocked loops is not intuitive
 - a. A liveness assertion must hold for all cycles of the loop
 - b. A liveness assumptions must only hold on at least one cycle of the loop
 - f. This is why we have a different property format when it is being used as assumption-
pending_ctr |-> ##[0:\$] (response ##1 pending_ctr == 0)
It means that a slave can fairly respond to all the requests as long as master eventually stops sending the requests (There is a finite end to the number of requests).
- 3. Cover properties to ensure dependencies between channel handshake signals are followed (IDI0022F_b Section A3.3.1)
 - a. Cover properties that confirm that each independent signal can be asserted without waiting for other signals
 - b. For example :

cover_awvalid_after_awready : cover property ((!awvalid && awready) [*1:\$] ##1 (awvalid && awready));

cover_awvalid_before_awready : cover property ((!awvalid && !awready) [*1:\$] ##1 (awvalid && !awready));
- 4. Parameter control to enable Deadlock checking mode in ABVIP
 - a. If CONFIG_LIVENESS is set, both fairness constraints and deadlock checks are enabled in ABVIP.
 - b. But some deadlocks can be caught only when the bus communication is not fair.
For example: transfers on write channel are dependent on transfers on read channel

- c. To catch these deadlocks, we need to switch off fairness constraints in ABVIP.
- d. To enable this mode, we have provided DEADLOCK_CHKS_ON parameter, which when set to 1, only enables deadlock checks (assertions) and not the fairness constraints (assumptions)
- e.  If DEADLOCK_CHKS_ON and CONFIG_LIVENESS are both set to 1, DEADLOCK_CHKS_ON takes precedence, that is, only deadlock checks are enabled.

Using ABVIP with Simulation

During simulation, the ABVIP package can verify that your design is compliant with the AXI protocol. To use ABVIP in simulation, you can do the following:

1. Write a testbench, or use your existing simulation-based environment, to provide the stimuli for the design. The ABVIP is responsible for protocol compliance checking and coverage during simulation; it cannot be used to create a simulation-based environment.
2. Instantiate the ABVIP as a monitor along with the DUV, and connect all corresponding interface signals.
3. All of the AXI master or slave DUV signals must be connected to the corresponding signals in the AXI ABVIP monitor. Tie the signals that do not exist in the master or slave interface to fixed values.
4. Use assertions to verify the functional behavior of DUV.
5. Use covers to measure the quality of stimulus generated.
6. This setup can be used at the system level as well.

The main advantage of using ABVIP with simulation is ease of configuration and debugging.

Configuring ABVIP

The ABVIP in simulation can be configured with relative ease. It is instantiated as a passive checker, that is, as a monitor. Refer to section [Monitor](#) and choose your monitor of choice for AXI3, AXI4, or ACE. Another key component of configuring the ABVIP is parameter settings. The user can quickly choose the ABVIP features the DUT supports and configure the ABVIP. Refer to section [Parameter Configuration](#) for parameter description in detail and configure your ABVIP. Once the ABVIP has been configured and instantiated in the testbench, you are all set to verify your DUT in simulation.

Debugging ABVIP

You can debug an assertion failure in simulation by setting the breakpoints on assertions or probing them to a simulation database. You can display assertions in the browser, waveform, and other simulation debugging windows. For more information, you can refer to user guide, *Assertion Checking in Simulation*.

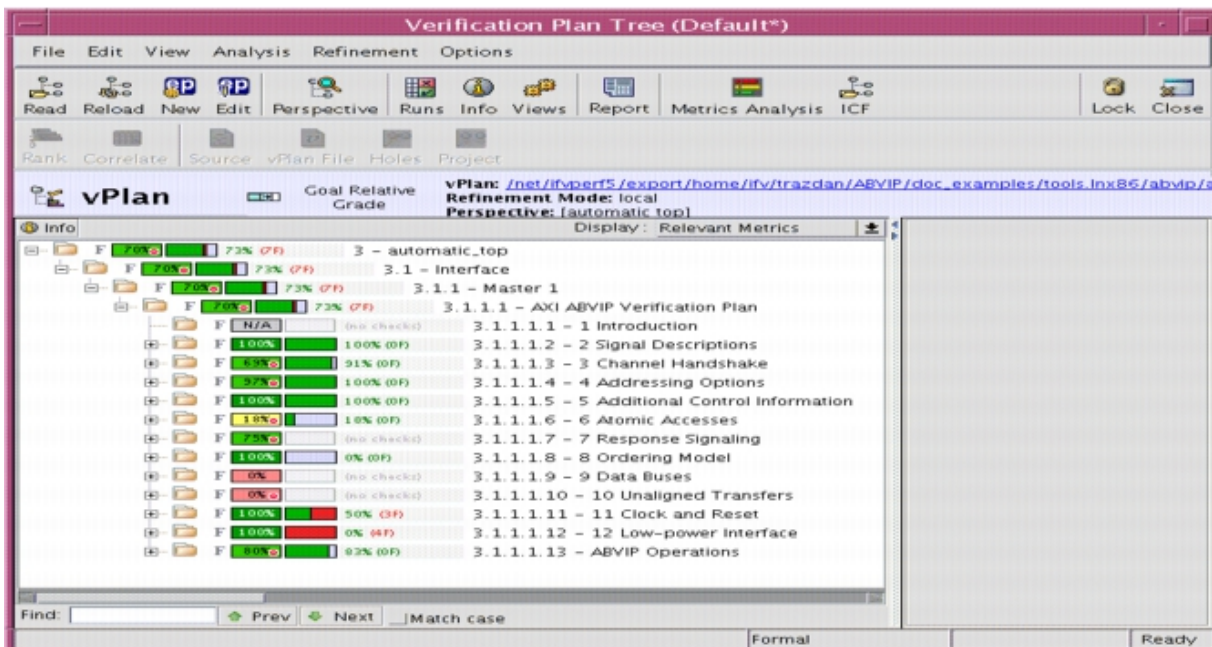
Verification Plan

- AXI3 vPlan
- AXI4 vPlan
- ACE vPlan

A verification plan defines the scope of a verification problem and its solution, that is, the functional specifications for the verification environment. The AXI3, AXI4, and ACE compliance vPlan extracts from the verification plan a hierarchical model of the verification goals in a format that the Enterprise Manager can read and use when displaying coverage results.


AXI3 vPlan

Figure 4-1 illustrates an AXI3 vPlan.



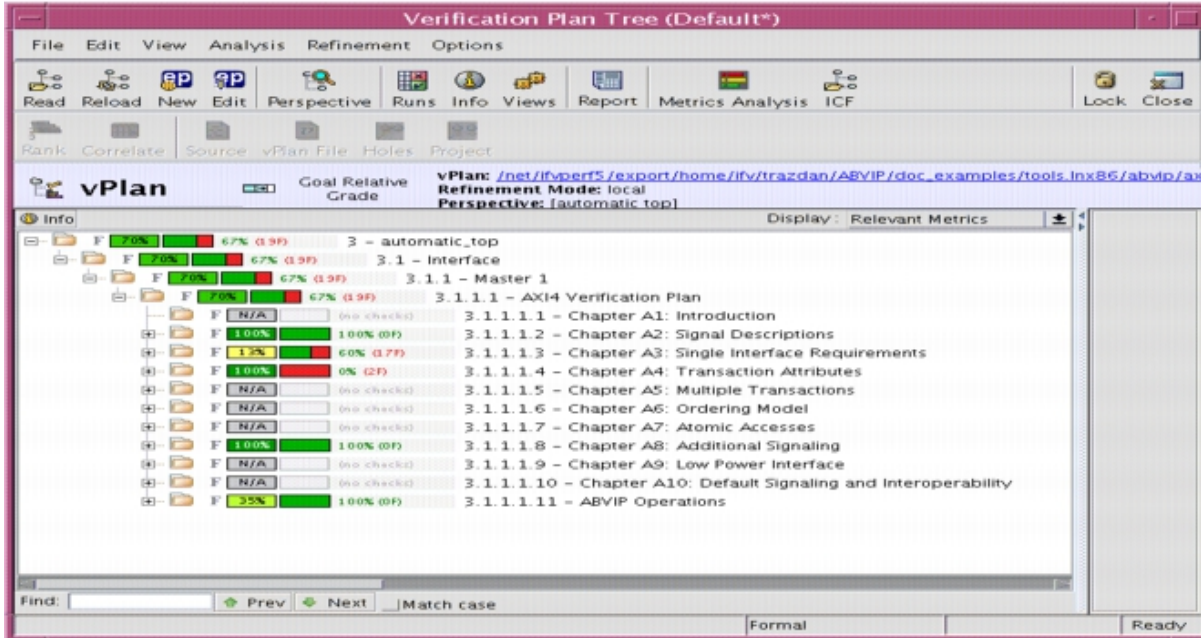
The vPlan structure is same as the ARM AXI specification ver 1.0. The key sections of the AXI3 vPlan include the following:

- Introduction
- Signal descriptions
- Channel Handshake
- Addressing Options
- Additional Control Information
- Atomic Accesses
- Response Signaling
- Ordering Model
- Data Buses
- Unaligned Transfers
- Clock and Reset
- Low-power Interface
- ABVIP Operations

 A 100% score in the AXI3 vPlan indicates that the DUT is compliant with the AXI3 protocol. The AXI3 vplan is compliant to ARM AMBA AXI specification ver1.0 (AMBAaxi.pdf). The AXI3 ABVIP checks and covers are associated with logical instance "AXI3_MON". You need to map ABVIP instances in the design hierarchy to this logical instance.

AXI4 vPlan

Figure 4-2 illustrates an AXI4 vPlan.



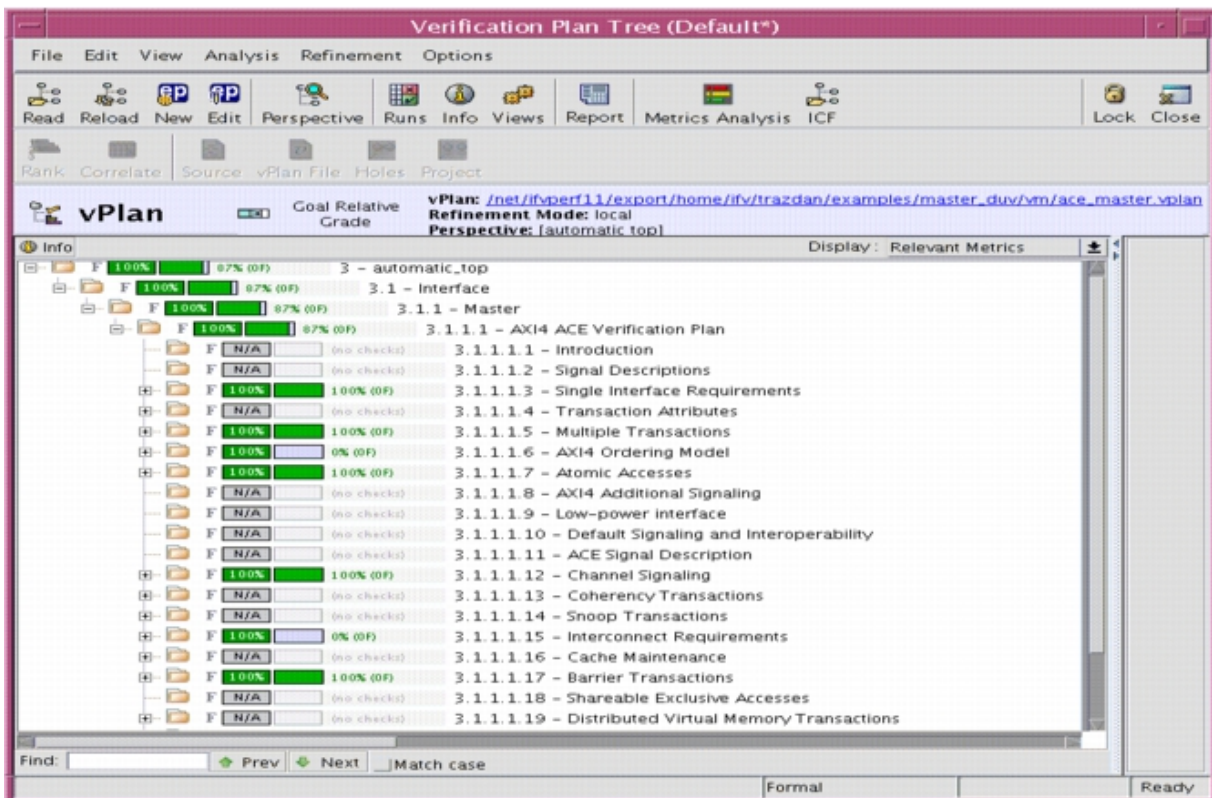
The AXI4 vplan structure is same as the ARM AMBA AXI protocol specifications. Key sections of the AXI4 vPlan include the following:

- Introduction
- Signal Descriptions
- Single Interface Requirements
- Transaction Attributes
- Multiple Transactions
- Ordering Model
- Atomic Accesses
- Additional Signaling
- Low Power Interface
- Default Signaling and Interoperability
- ABVIP Operations

⚠ A 100% score in the AXI4 vPlan indicates that the DUT is compliant with the AXI4 protocol. The AXI4 vplan is compliant to ARM AMBA AXI and ACE protocol specification (IHI0022D_amba_axi_protocol_spec.pdf). The AXI4 ABVIP checks and covers are associated with logical instance "AXI4_MON". You need to map ABVIP instances in the design hierarchy to this logical instance.

ACE vPlan


Figure 4-3 illustrates an ACE vPlan.



The ACE vplan structure is same as the ARM AMBA AXI protocol specifications. Key sections of the ACE vPlan include the following:

- Introduction
- Signal Descriptions
- Single Interface Requirements
- Transaction Attributes
- Multiple Transactions

- AXI4 Ordering Model
- Atomic Accesses
- AXI4 Additional Signaling
- Low-power interface
- Default Signaling and Interoperability
- ACE Signal Description
- Channel Signaling
- Coherency Transactions
- Snoop Transactions
- Interconnect Requirements
- Cache Maintenance
- Barrier Transactions
- Shareable Exclusive Accesses
- Distributed Virtual Memory Transactions

 A 100% score in the ACE vPlan indicates that the DUT is compliant with the ACE protocol. The ACE vplan is compliant with ARM AMBA AXI and ACE protocol specification (IHI0022D_amba_axi_protocol_spec.pdf). The ACE ABVIP checks and covers are associated with logical instance "AXI4_ACE_MON". You need to map ABVIP instances in the design hierarchy to this logical instance.

AXI3 Design Examples

- [Copying the Design Examples](#)
- [Verifying the AXI3 Master using Formal Analysis](#)
 - [Running an Example](#)
 - [Verifying AXI3 Master Design](#)
- [Verifying AXI Design](#)
- [Verifying the AXI3 Slave using Formal Analysis](#)
 - [Running an Example](#)
 - [Verifying AXI3 Slave Design](#)
- [Verifying the AXI3 Bridge using Formal Analysis](#)
 - [Running an Example](#)
 - [Verifying AXI3 Bridge Design](#)
- [Verifying the AXI3 Slave using Simulation](#)
 - [Running an Example](#)
 - [Verifying AXI3 slave Design in Simulation](#)
- [Verifying AXI Using Xcelium](#)

In this chapter, you will verify the AXI3 design using formal verification and simulation.

Copying the Design Examples

To run these examples, and to verify your own designs:

1. Set the environment variable as:
`setenv ABVIP_INST_DIR <VIPCAT_INSTALL_DIR>/tools/abvip`
2. Change to a working directory.

3. Copy the axi3 directory as:

```
cp -r $ABVIP_INST_DIR/axi3/examples .
```

Verifying the AXI3 Master using Formal Analysis

To verify the AXI3 master RTL design using formal verification, the slave behavior must be assumed and the master behavior must be checked. This is achieved by configuring AXI3 ABVIP Monitor as slave. The ABVIP constraints the slave inputs while master behavior is verified through the ABVIP assertions.

The code shown below demonstrates the instantiation of the AXI3 monitor as a slave in the design top.

```
module top();

localparam DATA_WIDTH = 32;

localparam ADDR_WIDTH = 16;


// System Clock and Reset

wire aresetn;

wire aclk;


// AXI specific signals

wire arvalid;

.....

.....


// Instance of the AXI master DUV

axi_duv_master axi_duv_master (

    .aclk_i (aclk ),

    .aresetn_i (aresetn),

    .awid_i (awid ),
```

```
.....  
.....  
        .cactive_i (cactive)  
);  
  
defparam axi_duv_master.ADDR_WIDTH = ADDR_WIDTH;  
defparam axi_duv_master.DATA_WIDTH = DATA_WIDTH;  
.....  
.....  
  
// Instance of the AXI3 Monitor configured as Slave  
axi3_slave axi_monitor (  
    .aclk (aclk ),  
    .aresetn (aresetn),  
    .awid (awid ),  
    .awaddr (awaddr ),  
    .....  
    .....  
    .cactive (cactive)  
);  
  
defparam axi_monitor.ADDR_WIDTH = ADDR_WIDTH;  
defparam axi_monitor.DATA_WIDTH = DATA_WIDTH;  
.....  
.....  
  
endmodule // top
```

The complete code of this example is located at `axi3/examples/master_duv`.

 The sample example in VHDL is present at `axi3/examples/master_duv_vhdl`.

Running an Example

To run an example, follow the steps given below:

1. Copy the example, as shown in [Copying the Design Examples](#).
2. Go to example directory `axi3/examples/master_duv`.
3. To run Jasper, specify the command:

```
jg jg_master.tcl
```
4. To run the example in Enterprise Manager:
 - Go to directory `axi3/examples/master_duv/vm`
 - Launch Enterprise Manager and load vsif file `axi3_master.vsif` present in the directory.
 - After the run completes, read the vmap file `axi3_master.vmap` present in the directory by giving the following command at the Enterprise Manager console.

```
read vmap axi3_master
```
 - Load `axi3_master.vplan` to view the coverage information extracted from the runs.

Verifying AXI3 Master Design


1. To verify your AXI3 master design, refer to [Verifying your AXI Design](#).
2. Run Jasper as explained in the section, [Running an Example](#).

The assertion failures reported are due to protocol violations in the RTL. You can view the counter examples to analyze the assertion failures.

Verifying AXI Design

To verify your AXI master, slave, or bridge design, refer to the following steps:

1. Copy your design to the `examples/rtl` directory.
2. Go to the examples test case directory and edit the `top.v` file.
 - Instantiate your RTL module in `top.v` and connect the ports.
 - Remove the instances of design provided with the example from `top.v`.

 In the example provided, all signals are directly connected between the DUV and the ABVIP instances through connecting nets defined in the design `top`.

3. Edit the `tcl` script to specify any additional Jasper commands that you want to run.
4. Edit the `tg.f` file to remove the reference to design files provided with an example and add the name of your design file.

Verifying the AXI3 Slave using Formal Analysis

In this example, the AXI3 monitor is configured as a master to verify the AXI3 slave RTL using formal verification. The master behavior must be assumed and the slave behavior must be checked during verification. The ABVIP constraints the master input while slave behaviors are verified through the ABVIP assertions.

The code shown below demonstrates the instantiation of the AXI3 Monitor as a master in the design `top`.

```
module top();

localparam DATA_WIDTH = 32;

localparam ADDR_WIDTH = 16;


// System Clock and Reset

wire aresetn;

wire aclk;


// AXI specific signals

wire arvalid;
```

```
.....

.....

// Instance of the AXI slave DUV
axi_duv_slave axi_duv_slave (

    .aclk_i (aclk ),

    .resetn_i (aresetn),

    .awid_i (awid ),

    .....

    .....

    .cactive_i (cactive)
);

defparam axi_duv_slave.ADDR_WIDTH = ADDR_WIDTH;
defparam axi_duv_slave.DATA_WIDTH = DATA_WIDTH;

.....

.....

// Instance of the AXI3 Monitor configured as Master
axi3_master axi_monitor (

    .aclk (aclk ),

    .aresetn (aresetn),

    .awid (awid ),

    .awaddr (awaddr ),

    .....


    .....

    .cactive (cactive)
);
```



```
defparam axi_monitor.ADDR_WIDTH = ADDR_WIDTH;  
  
defparam axi_monitor.DATA_WIDTH = DATA_WIDTH;  
  
.....  
  
.....  
  
endmodule // top
```

The complete code of this example is located at `axi3/examples/slave_duv`.

 The sample example in VHDL is present at `axi3/examples/slave_duv_vhdl`.

Running an Example

To run an example, follow the steps given below:

1. Copy the example, as shown in [Copying the Design Examples](#).
2. Go to example directory `axi3/examples/slave_duv`.
3. To run Jasper, specify the command:
`jg jg_slave.tcl`
4. To run the example in Enterprise Manager:
 - Go to directory `axi3/examples/slave_duv/vm`
 - Launch Enterprise Manager and load vsif file `axi3_slave.vsif` present in the directory.
 - After the run completes, read the vmap file `axi3_slave.vmap` present in the directory by giving the following command at the Enterprise Manager console.
`read vmap axi3_slave`
 - Load `axi3_slave.vplan` to view the coverage information extracted from the runs.

Verifying AXI3 Slave Design

1. To verify your AXI3 slave design, refer to [Verifying your AXI Design](#).
2. Run Jasper as explained in the section, [Running an Example](#).

The reported assertion failures are due to protocol violations in the RTL. You can view the counter-examples to analyze the assertion failures.

Verifying the AXI3 Bridge using Formal Analysis

The example illustrates the configuration of AXI3 ABVIP monitor as master and slave to verify AXI3 bridge RTL using formal verification.

To verify the AXI3 bridge RTL design using formal verification, both master and slave behaviors must be assumed and checked during verification. The ABVIP constraint the master and slave inputs while verifying the functionality of AXI bridge through ABVIP assertions.

The code shown below demonstrates the instantiation of the AXI3 master as master and slave in the design top, top.v file.

```
// Instance of the AXI Slave (connects to the master interface of the bridge)

axi3_slave axi_slave (

    .aclk (aclk_m),

    .aresetn (aresetn_m),

    .awid (awid_m),

    .....

    .....

    .cactive (cactive)

);

defparam axi_slave.ADDR_WIDTH = ADDR_WIDTH;

defparam axi_slave.DATA_WIDTH = DATA_WIDTH;
```

```
.....  
.....  
  
// Instance of the AXI Master (connects to the slave interface of the bridge)  
axi3_master axi_master (  
    .aclk (aclk_s),  
    .aresetn (aresetn_s),  
    .awid (awid_s),  
    .....  
    .....  
    .cactive (cactive)  
);  
  
defparam axi_master.ADDR_WIDTH = ADDR_WIDTH;  
defparam axi_master.DATA_WIDTH = DATA_WIDTH;  
.....  
.....
```

The AXI3 ABVIP slave is instantiated as slave along with master RTL design in the top level module.

The AXI bridge RTL is instantiated in top module. The complete code of this example is located at `axi3/examples/bridge_duv`.

 The sample example in VHDL is present at `axi3/examples/bridge_duv_vhdl`.

Running an Example

To run an example, follow the steps given below:

1. Copy the example, as shown in [Copying the Design Examples](#).

2. Go to example directory `axi3/examples/bridge_duv`.

3. To run Jasper, specify the command:

```
jg jg_brdge.tcl
```

4. To run the example in Enterprise Manager:

- Go to directory `axi3/examples/bridge_duv/vm`
- Launch Enterprise Manager and load vsif file `axi3_bridge.vsif` present in the directory.
- After the run completes, read the vmap file `axi3_bridge.vmap` present in the directory by giving the following command at the Enterprise Manager console.

```
read vmap axi3_bridge
```
- Load `axi3_bridge.vplan` to view the coverage information extracted from the runs.

Verifying AXI3 Bridge Design

1. To verify your AXI3 bridge design, refer to [Verifying your AXI Design](#).

2. Run Jasper as explained in the section, [Running an Example](#).

The reported assertion failures are due to protocol violations in the RTL. You can view counter-examples to analyze the assertion failures.

Verifying the AXI3 Slave using Simulation

During simulation, the ABVIP package can verify the compliance of your design with the AXI3 protocol. The example illustrates the configuration of AXI3 ABVIP as a monitor in the design top to verify the AXI slave RTL design using simulation.

To verify the AXI3 slave in simulation, the slave behavior must be checked during simulation and the simulation based environment is required to provide stimulus to the design.

The ABVIP is responsible for protocol compliance checking and coverage data collection during simulation. It cannot be used to create a simulation-based environment, hence a simulation-based

environment is required to provide stimulus to the design and by configuring AXI3 ABVIP as a monitor, the slave behavior is verified through ABVIP assertions.

The code shown below demonstrates the instantiation of the AXI3 monitor in the design top.

```
module top();

localparam DATA_WIDTH = 32;
localparam ADDR_WIDTH = 32;

// System Clock and Reset

wire aresetn;

wire aclk;

// AXI specific signals

wire arvalid;

.....

.....

// Instance of the AXI slave DUV

axi_duv_slave axi_duv_slave (

.aclk_i (aclk ),

.aresetn_i (aresetn),

.awid_i (awid ),

.....

.....

.cactive_i (cactive)

);

defparam axi_duv_slave.ADDR_WIDTH = ADDR_WIDTH;

defparam axi_duv_slave.DATA_WIDTH = DATA_WIDTH;
```

```
.....  
.....  
  
// Instance of the AXI3 Monitor configured as Checker  
axi3_monitor axi_monitor (  
    .aclk (aclk ),  
    .aresetn (aresetn),  
    .awid (awid ),  
    .awaddr (awaddr ),  
    .....  
    .....  
    .cactive (cactive)  
);  
  
defparam axi_monitor.ADDR_WIDTH = ADDR_WIDTH;  
defparam axi_monitor.DATA_WIDTH = DATA_WIDTH;  
.....  
.....  
  
endmodule // top
```

The complete code of this example is located at `axi3/examples/slave_duv_sim`

Running an Example

1. Copy the example, as shown in [Copying the Design Examples](#).
2. Go to example directory `axi3/examples/slave_duv_sim` and specify the command:
`irun -gui -f irun.f`
3. To run the example with coverage, specify the following command:
`irun -gui -f irun.f -defparam "top.axi_monitor.COVERAGE_ON=1"`

Verifying AXI3 slave Design in Simulation

1. To provide the stimuli for the design, write a test bench, or use your existing simulation-based environment.
2. Copy your AXI3 slave design to `axi3/examples/rtl`
 - Instantiate your slave RTL module in `top.v` and connect the ports.
 - Remove the `axi_duv_slave` DUV instance from `top.v` provided with an example.
3. Edit the `irun.tcl` script to specify any additional nc-sim commands that you want to run.
4. Run `irun` as explained in the section, [Running an Example](#).

Verifying AXI Using Xcelium

To verify, follow the steps given below:

1. Copy the example, as shown in [Copying the Design Examples](#).
2. Go to examples directory and set the environment variable `example_dir` as:
`$ setenv example_dir `pwd``
3. Go to example directory `examples/xcelium`
4. Clean the example - `./CLEAN`
5. To run the example in simulation, specify the following command:
`xrun -gui -f xrun.f`
6. To run the example in formal verification, specify the following command:

```
xrun -jg -gui -f xrun_jg.f
```

AXI4 Design Examples

- [Copying the Design Examples](#)
- [Verifying the AXI4 Master using Formal Analysis](#)
 - [Running an Example](#)
 - [Verifying AXI4 Master Design](#)
- [Verifying AXI Design](#)
- [Verifying the AXI4 Slave using Formal Analysis](#)
 - [Running an Example](#)
 - [Verifying AXI4 Slave Design](#)
- [Verifying the AXI4 Bridge using Formal Analysis](#)
 - [Running an Example](#)
 - [Verifying AXI4 Bridge Design](#)
- [Verifying the AXI4 Slave using Simulation](#)
 - [Running an Example](#)
 - [Verifying AXI4 slave Design in Simulation](#)
- [Verifying AXI4 Using Xcelium](#)

In this chapter, you will verify the AXI4 design using formal verification and simulation.

Copying the Design Examples

To run these examples, and to verify your own designs:

1. Set the environment variable as:
`setenv ABVIP_INST_DIR <VIPCAT_INSTALL_DIR>/tools/abvip`
2. Change to a working directory.

3. Copy the axi4 directory as:

```
cp -r $ABVIP_INST_DIR/axi4/examples .
```

Verifying the AXI4 Master using Formal Analysis

To verify the AXI4 master RTL design using formal verification, the slave behavior must be assumed and the master behavior must be checked. This is achieved by configuring AXI4 ABVIP monitor as slave. The ABVIP constraint the slave inputs while master behavior is verified through the ABVIP assertions.

The code shown below demonstrates the instantiation of the AXI4 monitor as a slave in the design top.

```
module top();

localparam DATA_WIDTH = 32;

localparam ADDR_WIDTH = 16;


// System Clock and Reset

wire aresetn;

wire aclk;


// AXI specific signals

wire arvalid;

.....

.....


// Instance of the AXI master DUV

axi_duv_master axi_duv_master (

.aclk_i (aclk ),

.aresetn_i (aresetn),

.awid_i (awid ),
```

```
.....  
.....  
.cactive_i (cactive)  
);  
  
defparam axi_duv_master.ADDR_WIDTH = ADDR_WIDTH;  
defparam axi_duv_master.DATA_WIDTH = DATA_WIDTH;  
.....  
.....  
  
// Instance of the AXI4 Monitor configured as Slave  
axi4_slave axi_monitor (  
    .aclk (aclk ),  
    .aresetn (aresetn),  
    .awid (awid ),  
    .awaddr (awaddr ),  
    .....  
    .....  
    .cactive (cactive)  
);  
  
defparam axi_monitor.ADDR_WIDTH = ADDR_WIDTH;  
defparam axi_monitor.DATA_WIDTH = DATA_WIDTH;  
.....  
.....  
  
endmodule // top
```

The AXI4 ABVIP monitor is instantiated as slave along with master RTL design in the top level module. The complete code of this example is located at `axi4/examples/master_duv`.

 The sample example in VHDL is present at `axi4/examples/master_duv_vhdl`.

Running an Example

To run an example, follow the steps given below:

1. Copy the example, as shown in [Copying the Design Examples](#).

2. Go to example directory `axi4/examples/master_duv`.

3. To run Jasper, specify the command:

```
jg jg_master.tcl -define coverage 0
```

4. To run the example with coverage, specify either of commands:

```
jg jg_master.tcl -define coverage 1
```

4. To run the example in Enterprise Manager:

- Go to directory `axi4/examples/master_duv/vm`
- Launch Enterprise Manager and load vsif file `axi4_master.vsif` present in the directory.
- After the run completes, read the vmap file `axi4_master.vmap` present in the directory by giving the following command at the Enterprise Manager console.

```
read vmap axi4_master
```
- Load `axi4_master.vplan` to view the coverage information extracted from the runs.


Verifying AXI4 Master Design

1. To verify your AXI4 master design, refer to [Verifying your AXI Design](#).
2. Run Jasper as explained in the section, [Running an Example](#).

Verifying AXI Design

To verify your AXI master, slave, or bridge design, refer to the following steps:

1. Copy your design to the `examples/rtl` directory.
2. Go to the examples test case directory and edit the `top.v` file.
 - Instantiate your RTL module in `top.v` and connect the ports.
 - Remove the instances of design provided with the example from `top.v`.

 In the example provided, all signals are directly connected between the DUV and the ABVIP instances through connecting nets defined in the design `top`.

3. Edit the `tcl` script to specify any additional Jasper commands that you want to run.
4. Edit the `tg.f` file to remove the reference to design files provided with an example and add the name of your design file.

Verifying the AXI4 Slave using Formal Analysis

In this example, the AXI4 monitor is configured as a master to verify the AXI4 slave RTL using formal verification. The master behavior must be assumed and the slave behavior must be checked during verification. The ABVIP constraints the master input while slave behaviors are verified through the ABVIP assertions.

The code shown below demonstrates the instantiation of the AXI4 Monitor as a master in the design `top`.

```
module top();

localparam DATA_WIDTH = 32;

localparam ADDR_WIDTH = 16;


// System Clock and Reset

wire aresetn;

wire aclk;
```

```
// AXI specific signals

wire arvalid;

.....

.....

// Instance of the AXI slave DUV
axi_duv_slave axi_duv_slave (

.aclk_i (aclk ),

.aresetn_i (aresetn),

.awid_i (awid ),

.....

.....

.cactive_i (cactive)

);

defparam axi_duv_slave.ADDR_WIDTH = ADDR_WIDTH;

defparam axi_duv_slave.DATA_WIDTH = DATA_WIDTH;

.....

.....

// Instance of the AXI4 Monitor configured as Master
axi4_master axi_monitor (

.aclk (aclk ),

.aresetn (aresetn),

.awid (awid ),

.awaddr (awaddr ),

.....
```

```
.....  
.cactive (cactive)  
);  
  
defparam axi_monitor.ADDR_WIDTH = ADDR_WIDTH;  
defparam axi_monitor.DATA_WIDTH = DATA_WIDTH;  
  
.....  
.....  
  
endmodule // top
```

The complete code of this example is located at `axi4/examples/slave_duv`.

 The sample example in VHDL is present at `axi4/examples/slave_duv_vhdl`.

Running an Example

To run an example, follow the steps given below:

1. Copy the example, as shown in [Copying the Design Examples](#).
2. Go to example directory `axi4/examples/slave_duv`.
3. To run Jasper, specify the command:

```
jg jg_slave.tcl -define coverage 0
```

4. To run the example with coverage, specify either of commands:

```
jg jg_slave.tcl -define coverage 1
```

5. To run the example in Enterprise Manager:
 - Go to directory `axi4/examples/slave_duv/vm`
 - Launch Enterprise Manager and load vsif file `axi4_slave.vsif` present in the directory.

- After the run completes, read the vmap file `axi4_slave.vmap` present in the directory by giving the following command at the Enterprise Manager console.
read vmap axi4_slave
- Load `axi4_slave.vplan` to view the coverage information extracted from the runs.

Verifying AXI4 Slave Design

1. To verify your AXI4 slave design, refer to [Verifying your AXI Design](#).
2. Run Jasper as explained in the section, [Running an Example](#).

Verifying the AXI4 Bridge using Formal Analysis

The example illustrates the configuration of AXI4 ABVIP monitor as master and slave to verify AXI4 bridge RTL using formal verification.

To verify the AXI4 bridge RTL design using formal verification, both master and slave behaviors must be assumed and checked during verification. The ABVIP constraint the master and slave inputs while verifying the functionality of AXI bridge through ABVIP assertions.

The code shown below demonstrates the instantiation of the AXI4 monitor as master and slave in the design top through `bind, axi_duv_bridge.sv` file.

```
// Instance of the AXI Slave (connects to the master interface of the bridge)

bind top axi4_slave axi_slave (

    .aclk (aclk_m),

    .aresetn (aresetn_m),

    .awid (awid_m),

    .....

    .....

    .cactive (cactive)

);
```



```
defparam axi_slave.ADDR_WIDTH = ADDR_WIDTH;

defparam axi_slave.DATA_WIDTH = DATA_WIDTH;

.....

.....

// Instance of the AXI Master (connects to the slave interface of the bridge)

bind top axi4_master axi_master (

    .aclk (aclk_s),

    .aresetn (aresetn_s),

    .awid (awid_s),

    .....

    .....

    .cactive (cactive)

);

defparam axi_master.ADDR_WIDTH = ADDR_WIDTH;

defparam axi_master.DATA_WIDTH = DATA_WIDTH;

.....

.....

}
```

The AXI4 ABVIP slave is instantiated as slave along with master RTL design in the top level module.

The AXI bridge RTL is instantiated in top module. The complete code of this example is located at `axi4/examples/bridge_duv`.

 The sample example in VHDL is present at `axi4/examples/bridge_duv_vhdl`.

Running an Example

To run an example, follow the steps given below:

1. Copy the example, as shown in [Copying the Design Examples](#).
2. Go to example directory `axi4/examples/bridge_duv`.
3. To run Jasper, specify the command:

```
jg jg_bridge.tcl -define coverage 0
```
4. To run the example with coverage, specify either of commands:

```
jg jg_bridge.tcl -define coverage 1
```
5. To run the example in Enterprise Manager:
 - Go to directory `axi4/examples/bridge_duv/vm`
 - Launch Enterprise Manager and load vsif file `axi4_bridge.vsif` present in the directory.
 - After the run completes, read the vmap file `axi4_bridge.vmap` present in the directory by giving the following command at the Enterprise Manager console.

```
read vmap axi4_bridge
```
 - Load `axi4_bridge.vplan` to view the coverage information extracted from the runs.

Verifying AXI4 Bridge Design

1. To verify your AXI4 bridge design, refer to [Verifying your AXI Design](#).
2. Run Jasper as explained in the section, [Running an Example](#).

Verifying the AXI4 Slave using Simulation

During simulation, the ABVIP package can verify that your design is compliant with the AXI4 protocol. The example illustrates the configuration of AXI4 ABVIP as monitor in design top to verify AXI slave RTL design using simulation.

To verify the AXI4 slave in simulation, the slave behavior must be checked during simulation and the simulation based environment is required to provide stimulus to the design.

The ABVIP is responsible for protocol compliance checking and coverage data collection during simulation. It cannot be used to create a simulation-based environment, hence a simulation based

environment is required to provide stimulus to the design and by configuring AXI4 ABVIP as a monitor, the slave behavior is verified through ABVIP assertions.

The code shown below demonstrates the instantiation of the AXI4 monitor in the design top.

```
module top();

localparam DATA_WIDTH = 32;
localparam ADDR_WIDTH = 32;

// System Clock and Reset

wire aresetn;

wire aclk;

// AXI specific signals

wire arvalid;

.....

.....

// Instance of the AXI slave DUV

axi_duv_slave axi_duv_slave (

.aclk_i (aclk ),

.aresetn_i (aresetn),

.awid_i (awid ),

.....

.....

.cactive_i (cactive)

);

defparam axi_duv_slave.ADDR_WIDTH = ADDR_WIDTH;

defparam axi_duv_slave.DATA_WIDTH = DATA_WIDTH;
```

```
.....  
.....  
  
// Instance of the AXI4 Monitor configured as Checker  
axi4_monitor axi_monitor (  
    .aclk (aclk ),  
    .aresetn (aresetn),  
    .awid (awid ),  
    .awaddr (awaddr ),  
    .....  
    .....  
    .cactive (cactive)  
);  
  
defparam axi_monitor.ADDR_WIDTH = ADDR_WIDTH;  
defparam axi_monitor.DATA_WIDTH = DATA_WIDTH;  
.....  
.....  
  
endmodule // top
```

The complete code of this example is located at `axi4/examples/slave_duv_sim`.

Running an Example

To run an example, follow the steps given below:

1. Copy the example, as shown in [Copying the Design Examples](#).
2. Go to example directory `axi4/examples/slave_duv_sim` and specify the command:
`irun -gui -f irun.f`

3. To run the example with coverage, specify the following command:

```
irun -gui -f irun.f -defparam "top.axi_monitor.COVERAGE_ON=1"
```

Verifying AXI4 slave Design in Simulation

1. To provide the stimuli for the design, write a test bench, or use your existing simulation-based environment.
2. Copy your AXI4 slave design to `axi4/examples/rtl`
 - Instantiate your slave RTL module in `top.v` and connect the ports.
 - Remove the `axi_duv_slave` DUV instance from `top.v` provided with an example.
3. Edit the `irun.tcl` script to specify any additional nc-sim commands that you want to run.
4. Edit the `irun.f` file to remove the reference to `axi_duv_slave_sim.v` file , and add the name of your design file.

Run `irun` as explained in the section, [Running an Example](#).

Verifying AXI4 Using Xcelium

To verify, follow the steps given below:

1. Copy the example, as shown in [Copying the Design Examples](#).
2. Go to examples directory and set the environment variable `example_dir` as:

```
$ setenv example_dir `pwd`
```

3. Go to example directory *examples/xcelium*
4. Clean the example - `./CLEAN`
5. To run the example in simulation, specify the following command:

```
xrun -gui -f xrun.f
```

6. To run the example in formal verification, specify the following command:

```
xrun -jg -gui -f xrun_jg.f
```

ACE Design Examples

- [Copying the Design Examples](#)
- [Verifying the ACE Master using Formal Analysis](#)
 - [Running an Example](#)
 - [Verifying ACE Master Design](#)
- [Verifying ACE Design](#)
- [Verifying the ACE Bridge using Formal Analysis](#)
 - [Running an Example](#)
 - [Verifying ACE Bridge Design](#)
- [Verifying the ACE-LITE Master using Formal Analysis](#)
 - [Running an Example](#)
 - [Verifying ACE Master Design](#)
- [ABVIP_SIM : Running Formal Setup in Simulation](#)
 - [Running using ABVIP_SIM option](#)
- [Verifying AXI4_ACE Using Xcelium](#)

In this chapter, you will verify the ACE design using formal verification.

Copying the Design Examples

To run these examples, and to verify your own designs:

1. Set the environment variable as:
`setenv ABVIP_INST_DIR <VIPCAT_INSTALL_DIR>/tools/abvip`
2. Change to a working directory.
3. Copy the ace directory as:

```
cp -r $ABVIP_INST_DIR/axi4_ace/examples .
```

Verifying the ACE Master using Formal Analysis

To verify the ACE master RTL design using formal verification, the slave behavior must be assumed and the master behavior must be checked. This is achieved by configuring ACE ABVIP monitor as slave. The ABVIP constraint the slave inputs while master behavior is verified through the ABVIP assertions.

The code shown below demonstrates the instantiation of the ACE monitor as a slave in the design top.

```
module top();

localparam ID_WIDTH = 2;
localparam DATA_WIDTH = 64;

input ARESETn;
input ACLK;


input ARVALID;
.....
.....

// Instance of ACE Master DUV
axi4_ace_master
#(
  .DATA_WIDTH(DATA_WIDTH),
  .ID_WIDTH(ID_WIDTH),
  .....
  .....
)
MI (
  .aclk (ACLK),
  .aresetn(ARESETn),
  .arvalid(ARVALID),
  .....
  .....
);

// Instance of ACE monitor configured as slave
axi4_ace_slave
#(
  .DATA_WIDTH(DATA_WIDTH),
  .ID_WIDTH(ID_WIDTH),
  .....
  .....
)
SI (
  .aclk (ACLK),
  .aresetn(ARESETn),
  .arvalid(ARVALID),
  .....
  .....
);

endmodule // top
```

The complete code of this example is located at `axi4_ace/examples/master_duv`.

 The sample example in VHDL is present at `axi4_ace/examples/master_duv_vhdl`.

Running an Example

To run an example, follow the steps given below:

1. Copy the example, as shown in [Copying the Design Examples](#).

2. Go to example directory `axi4_ace/examples/master_duv`.

3. To run Jasper, specify the command:

```
jg jg_master.tcl -define coverage 0
```

4. To run the example with coverage, specify either of the two commands:

```
jg jg_master.tcl -define coverage 1
```

5. To run the example in Enterprise Manager:

- Go to directory `axi4_ace/examples/master_duv/vm`
- Launch Enterprise Manager and load vsif file `master_duv.vsif` present in the directory.
- After the run completes, read the vmap file `master_duv.vmap` present in the directory by giving the following command at the Enterprise Manager console.

```
read vmap master_duv
```
- Load `master_duv.vplan` to view the coverage information extracted from the runs.


Verifying ACE Master Design

1. To verify your ACE master design, refer to [Verifying your ACE Design](#).
2. Run Jasper as explained in the section, [Running an Example](#).

Verifying ACE Design

To verify your ACE master or bridge design, refer to the following steps:

1. Copy your design to the `examples/rtl` directory.
2. Go to the examples test case directory and edit the `top.vfile`.
 - Instantiate your RTL module in `top.v` and connect the ports.
 - Remove the instances of design provided with the example from `top.v`.

 In the example provided, all signals are directly connected between the DUV and the ABVIP instances through connecting nets defined in the design top.

3. Edit the tcl script to specify any additional Jasper commands that you want to run.
4. Edit the jg.f file to remove the reference to design files provided with an example and add the name of your design file.

Verifying the ACE Bridge using Formal Analysis

The example illustrates the configuration of ACE ABVIP monitor as master and slave to verify ACE bridge RTL using formal verification.

To verify the ACE bridge RTL design using formal verification, both master and slave behaviors must be assumed and checked during verification. The ABVIP constraint the master and slave inputs while verifying the functionality of ACE bridge through ABVIP assertions.

The code shown below demonstrates the instantiation of two ACE monitor as master and two slaves in the design.

```
// Instantiation of master 1
axi4_ace_master
#(
  .DATA_WIDTH(DATA_WIDTH),
  .ID_WIDTH(ID_WIDTH),
  .....
)
MI1 (
  .aclk (ACLK),
  .aresetn(ARESETn),
  .arvalid(ARVALID_M1),
  .....
);

// Instantiation of master 2
axi4_ace_master
#(
  .DATA_WIDTH(DATA_WIDTH),
  .ID_WIDTH(ID_WIDTH),
  .....
)
```

```
.....
)
MI2 (
  .aclk (ACLK),
  .aresetn(ARESETn),
  .arvalid(ARVALID_M2),
  .....
  .....
);


// Instantiation of bridge
ace_duv_bridge
#(
  .DATA_WIDTH(DATA_WIDTH),
  .ID_WIDTH(ID_WIDTH),
  .....
  .....
)
MI2 (
  .aclk (ACLK),
  .aresetn(ARESETn),
  .arvalid_s1(ARVALID_M1),
  .....
  .....
  .arvalid_s2(ARVALID_M2),
  .....
  .....
);

// Instantiation of slave 1
axi4_ace_slave
#(
  .DATA_WIDTH(DATA_WIDTH),
  .ID_WIDTH(ID_WIDTH),
  .....
  .....
)
SI1 (
  .aclk (ACLK),
  .aresetn(ARESETn),
  .arvalid(ARVALID_S1),
  .....
  .....
);

// Instantiation of slave 1
axi4_ace_slave
#(
  .DATA_WIDTH(DATA_WIDTH),
  .ID_WIDTH(ID_WIDTH),
  .....
  .....
)
- - -
```

```
SI2 (  
  .aclk (ACLK),  
  .aresetn (ARESETn),  
  .arvalid (ARVALID_S2),  
  .....  
  .....  
);
```

The complete code of this example is located at `axi4_ace/examples/bridge_duv`.

 The sample example in VHDL is present at `axi4_ace/examples/bridge_duv_vhdl`.

Running an Example

To run an example, follow the steps given below:

1. Copy the example, as shown in [Copying the Design Examples](#).
2. Go to example directory `axi4_ace/examples/bridge_duv`.
3. To run Jasper, specify the command:
`jg jg_bridge.tcl -define coverage 0`
4. To run the example with coverage, specify the command:
`jg jg_bridge.tcl -define coverage 1`
5. To run the example in Enterprise Manager:
 - Go to directory `axi4_ace/examples/bridge_duv/vm`
 - Launch Enterprise Manager and load vsif file `bridge_duv.vsif` present in the directory.
 - After the run completes, read the vmap file `bridge_duv.vmap` present in the directory by giving the following command at the Enterprise Manager console.
`read vmap bridge_duv`
 - Load `bridge_duv.vplan` to view the coverage information extracted from the runs.

Verifying ACE Bridge Design

1. To verify your ACE bridge design, refer to [Verifying your ACE Design](#).
2. Run Jasper as explained in the section, [Running an Example](#).

Verifying the ACE-LITE Master using Formal Analysis

To verify the ACE-LITE master RTL design using formal verification, the slave behavior must be assumed and the master behavior must be checked. This is achieved by configuring ACE-LITE ABVIP monitor as slave. The ABVIP constraint the slave inputs while master behavior is verified through the ABVIP assertions.

The code shown below demonstrates the instantiation of the ACE-LITE monitor as a slave in the design top.

```
module top();

localparam ID_WIDTH = 2;
localparam DATA_WIDTH = 64;

input ARESETn;
input ACLK;

input ARVALID;
.....
.....

// Instance of ACE-LITE Master DUV
axi4_ace_lite_master
#(
    .DATA_WIDTH(DATA_WIDTH),
    .ID_WIDTH(ID_WIDTH),
    .....
    .....
)
MI (
    .aclk (ACLK),
    .aresetn(ARESETn),
    .arvalid(ARVALID),
    .....
    .....
);

// Instance of ACE-LITE monitor configured as slave
axi4_ace_lite_slave
#(
    .DATA_WIDTH(DATA_WIDTH),
    .ID_WIDTH(ID_WIDTH),
    .....
    .....
)
SI (
    .aclk (ACLK),
    .aresetn(ARESETn),
    .arvalid(ARVALID),
    .....
    .....
);

endmodule // top
```

The complete code of this example is located at `axi4_ace/examples/ace_lite_master_duv`.

Running an Example

To run an example, follow the steps given below:

1. Copy the example, as shown in [Copying the Design Examples](#).
2. Go to example directory `axi4_ace/examples/ace_lite_master_duv`.
3. To run Jasper, specify the command:

```
jg jg_master.tcl -define coverage 0
```
4. To run the example with coverage, specify the following command:

```
jg jg_master.tcl -define coverage 1
```
5. To run the example in Enterprise Manager:
 - Go to directory `axi4_ace/examples/ace_lite_master_duv/vm`
 - Launch Enterprise Manager and load vsif file `ace_lite_master.vsif` present in the directory.
 - After the run completes, read the vmap file `ace_lite_master.vmap` present in the directory by giving the following command at the Enterprise Manager console.

```
read vmap master_duv
```
 - Load `master_duv.vplan` to view the coverage information extracted from the runs.

Verifying ACE Master Design

1. To verify your ACE master design, refer to [Verifying your ACE Design](#).
2. Run Jasper as explained in the section, [Running an Example](#).

ABVIP_SIM : Running Formal Setup in Simulation

Option `ABVIP_SIM` enables you to use same formal setup in simulation.

For example, consider the following instance defined in the formal analysis:

```
axi4_ace_master
# (
    // AXI 4 USER DEFINED PARAMS
    .ID_WIDTH          (M_ID_WIDTH          ), // Size of ID field
    .ADDR_WIDTH        (ADDR_WIDTH         ), // Size of Address field
    .DATA_WIDTH        (DATA_WIDTH         ), // Size of Data field
    .....
    .....
)

ACE_Master (

    // Global signals
    .aclk      (ACLK),
    .aresetn   (ARESETn),
    .csysreq   (CSYSREQ),
    .csysack   (CSYSACK),
    .cactive   (CACTIVE),

    // Write Address Channel
    .awid      (AWIDS4),
    .awaddr    (AWADDRS4),
    .awlen     (AWLENS4),
    .awsiz     (AWSIZES4),
    .....
    .....
);
```

In formal analysis, the above instantiation works as a ACE master by driving the ACE master interface signals and checking the ACE slave interface signals.

You can use the same instance as monitor in simulation without changing the instantiation. The monitor will check both master and slave interface signals.

Running using ABVIP_SIM option

- Run xrun as explained in section, [Verifying AXI4_ACE Using Xcelium](#):

Verifying AXI4_ACE Using Xcelium

To verify, follow the steps given below:

1. Copy the example, as shown in [Copying the Design Examples](#).
2. Go to examples directory and set the environment variable `example_dir` as:

```
$ setenv example_dir `pwd`
```


3. Go to example directory *examples/xcelium*
4. Clean the example - `./CLEAN`
5. To run the example in simulation, specify the following command:
`xrun -gui -f xrun.f`
6. To run the example in formal verification, specify the following command:
`xrun -jg -gui -f xrun_jg.f`

AXI5 Design Examples

- [Copying the Design Examples](#)
- [Verifying the AXI5 Slave using Formal Verification \(Jasper\)](#)
 - [Running an Example](#)
 - [Verifying AXI5 Master Design](#)
- [Verifying the AXI5-G Slave using Formal Verification \(Jasper\)](#)
 - [Running an Example](#)
 - [Verifying AXI5-G Master Design](#)
- [Verifying AXI5 Using Xcelium](#)

In this chapter, you will verify the AXI5 design using formal verification and simulation.

Copying the Design Examples

To run these examples, and to verify your own designs:

1. Set the environment variable as:
`setenv ABVIP_INST_DIR <VIPCAT_INSTALL_DIR>/tools/abvip`
2. Change to a working directory.
3. Copy the AMBA5 AXI example directory as:
`cp -r $ABVIP_INST_DIR/axi5/examples .`

Verifying the AXI5 Slave using Formal Verification (Jasper)

To verify the AXI5 slave RTL design using formal verification, the master behavior must be assumed and the slave behavior must be checked. The AXI5 ABVIP master module is bound along with slave RTL design module. AXI5 master module will constraint the slave inputs and verify the slave outputs. The example also binds AXI5 ABVIP slave module to the dummy slave RTL to get the behavior of a AXI5 slave. The code of this example is located at `axi5/examples/slave_duv`.

Running an Example

To run an example, follow the steps given below:

1. Copy the example, as shown in [Copying the Design Examples](#).
2. Go to example directory `axi5/examples/slave_duv`.
3. To run Jasper, specify the command:

```
jg jg_slave_duv.tcl
```

Verifying AXI5 Master Design

To verify your AXI5 master design, you should bind AXI5 slave module to master duv. AXI5 slave module constraints the master inputs and verify the master outputs.

Verifying the AXI5-G Slave using Formal Verification (Jasper)

To verify the AXI5-G slave RTL design using formal verification, the master behavior must be assumed and the slave behavior must be checked. The AXI5 ABVIP master module is bound along with slave RTL design module. AXI5 master module will constraint the slave inputs and verify the slave outputs. The example also binds AXI5 ABVIP slave module to the dummy slave RTL to get the behavior of a AXI5 slave. The code of this example is located at `axi5/examples/slave_duv`.

Running an Example

To run an example, follow the steps given below:

1. Copy the example, as shown in [Copying the Design Examples](#).
2. Go to example directory `axi5/examples/slave_duv`.
3. Change the `CDNS_AMBA5_G` parameter value from 0 to 1 in the `amba5_axi_slave_duv.sv` file.
This parameter will enable all the AMBA5-G features.
4. To run Jasper, specify the command:

```
jg jg_slave_duv.tcl
```

Verifying AXI5-G Master Design

To verify your AXI5 master design, you should bind AXI5 slave module to master duv. AXI5 slave module will constraint the master inputs and verify the master outputs.

Verifying AXI5 Using Xcelium

To verify, follow the steps given below:

1. Copy the example, as shown in [Copying the Design Examples](#).
2. Go to examples directory and set the environment variable `example_dir` as:

```
$ setenv example_dir `pwd`
```
3. Go to example directory `examples/xcelium`
4. Clean the example - `./CLEAN`
5. To run the AXI-G version, change the `CDNS_AMBA5_G` parameter value from 0 to 1 in `cdns_abvip_axi5_master.sv` and `cdns_abvip_axi5_slave.sv` file.
6. To run the example in simulation, specify the following command:

```
xrun -gui -f xrun.f
```

7. To run the example in formal verification, specify the following command:

```
xrun -jg -gui -f xrun_jg.f
```

ACE5 Design Examples

- [Copying the Design Examples](#)
- [Verifying the ACE5 Slave using Formal Verification \(Jasper\)](#)
 - [Running an Example](#)
 - [Verifying ACE5 Master Design](#)
- [Verifying the ACE5-G Slave using Formal Verification \(Jasper\)](#)
 - [Running an Example](#)
 - [Verifying ACE5-G Master Design](#)
- [Verifying ACE5 Using Xcelium](#)

In this chapter, you will verify the ACE5 design using formal verification and simulation.

Copying the Design Examples

To run these examples, and to verify your own designs:

1. Set the environment variable as:
`setenv ABVIP_INST_DIR <VIPCAT_INSTALL_DIR>/tools/abvip`
2. Change to a working directory.
3. Copy the ace5 directory as:
`cp -r $ABVIP_INST_DIR/ace5/examples .`

Verifying the ACE5 Slave using Formal Verification (Jasper)

To verify the ACE5 slave RTL design using formal verification, the master behavior must be assumed and the slave behavior must be checked. The ACE5 ABVIP master module is bound along with slave RTL design module. ACE5 master module will constraint the slave inputs and verify the slave outputs. The example also binds ACE5 ABVIP slave module to the dummy slave RTL to get the behavior of a ACE5 slave. The code of this example is located at `ace5/examples/slave_duv`.

Running an Example

To run an example, follow the steps given below:

1. Copy the example, as shown in [Copying the Design Examples](#).
2. Go to example directory `ace5/examples/slave_duv`.
3. To run Jasper, specify the command:

```
jg jg_slave_duv.tcl
```

Verifying ACE5 Master Design

To verify your ACE5 master design, you should bind ACE5 slave module to master duv. ACE5 slave module constraints the master inputs and verify the master outputs.

Verifying the ACE5-G Slave using Formal Verification (Jasper)

To verify the ACE5-G slave RTL design using formal verification, the master behavior must be assumed and the slave behavior must be checked. The ACE5-G ABVIP master module is bound along with slave RTL design module. ACE5-G master module will constraint the slave inputs and verify the slave outputs. The example also binds ACE5-G ABVIP slave module to the dummy slave RTL to get the behavior of a ACE5-G slave. The code of this example is located at `ace5/examples/slave_duv`.

Running an Example

To run an example, follow the steps given below:

1. Copy the example, as shown in [Copying the Design Examples](#).
2. Go to example directory `ace5/examples/slave_duv`.
3. Change the `CDNS_AMBA5_G` parameter value from 0 to 1 in the `amba5_ace_slave_duv.sv` file.
This parameter will enable all the AMBA5-G features
4. To run Jasper, specify the command:

```
jg jg_slave_duv.tcl
```

Verifying ACE5-G Master Design

To verify your ACE5-G master design, you should bind ACE5-G slave module to master duv. ACE5-G slave module will constraint the master inputs and verify the master outputs.

Verifying ACE5 Using Xcelium

To verify, follow the steps given below:

1. Copy the example, as shown in [Copying the Design Examples](#).
2. Go to examples directory and set the environment variable `example_dir` as:

```
$ setenv example_dir `pwd`
```
3. Go to example directory `examples/xcelium`
4. Clean the example - `./CLEAN`
5. To run the AXI-G version , change the `CDNS_AMBA5_G` parameter value from 0 to 1
in `cdns_abvip_ace5_master.sv` and `cdns_abvip_ace5_slave.sv` file
6. To run the example in simulation, specify the following command:

```
xrun -gui -f xrun.f
```

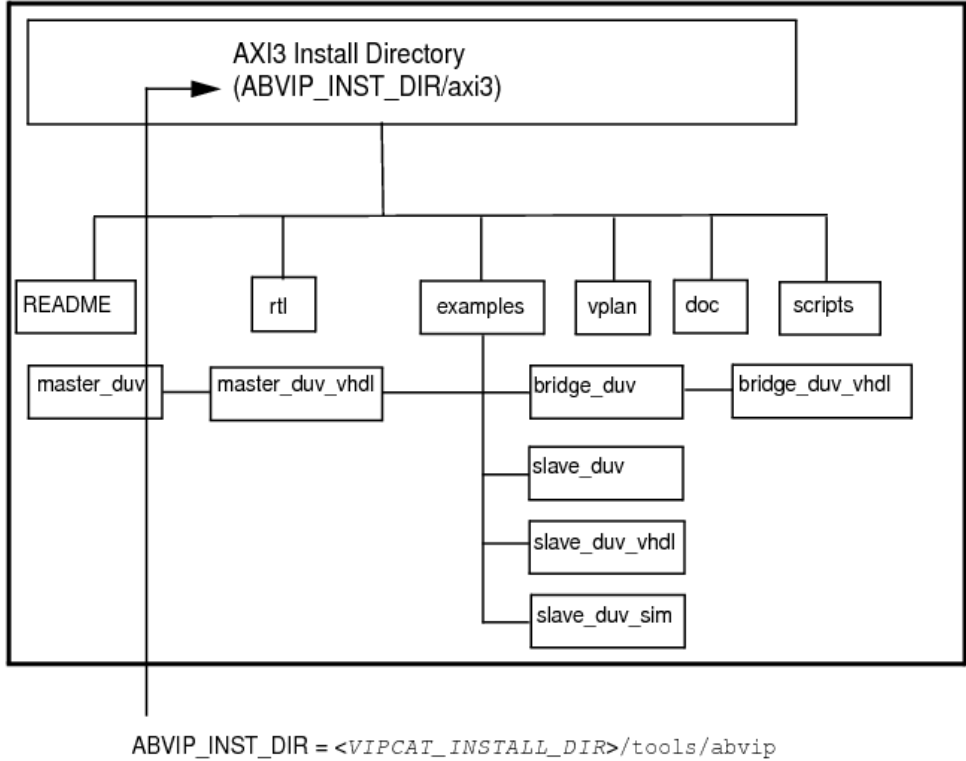
7. To run the example in formal verification, specify the following command:

```
xrun -jg -gui -f xrun_jg.f
```

Appendix A: AXI3 ABVIP Package Content

- [Package Contents](#)
- [AXI3 Monitor Pin-Level Interface](#)
- [AXI3 ABVIP Debug Signals](#)
- [AXI3 Monitor Parameters](#)
- [AXI3 Checks](#)
 - [Compliance Checks](#)
- [Coverage Checks](#)

This appendix describes the content of the AXI3 ABVIP package. [Figure A-1](#) shows the package directory structure. **Figure A-1 Directory Structure of the AXI3 ABVIP Package**



Package Contents

[Table A-1](#) shows the contents of the AXI3 installation directory.

Table A-1 Contents of the AXI3 Installation Directory

README	Mentions the contents of the package.
--------	---------------------------------------

rtl	Contains the code for AXI3 ABVIP.
examples	Contains the examples of AXI3 ABVIP to be used in formal and simulation based verification.
vplan	Contains the AXI1.0 specification based verification plan for AXI3 ABVIP. This can be used to view check status and coverage collected in the Enterprise Manager run.
doc	Contains the AXI ABVIP user guide and the migration guide.
scripts	Contains the scripts to convert log file from the original AXI3 log file to new assertion names and compares results of original and new log files for AXI3 ABVIP.

Table A-2 shows the files stored in the rtl folder.

Table A-2 Contents of the rtl Folder

Version.v	Prints the version number of the AXI3 ABVIP.
cdn_abvip_axi3_monitor.svp	Encrypted SVA based monitor with protocol compliance checks and functional covers.
cdn_abvip_axi3_master.svp	Encrypted master module driving AXI3 master interface signals and checking AXI3 slave interface signals.
cdn_abvip_axi3_slave.svp	Encrypted slave module driving AXI3 slave interface signals and checking AXI3 master interface signals.
cdn_abvip_axi3.sv	Configurable abvip module which instantiates master/slave/monitor based on VERIFY_BLK parameter
axi3_monitor.sv	Configure cdn_abvip_axi3_monitor.svp with an old abvip module name. This file is kept for backward compatibility and will be removed in a future release.
axi3_master.sv	Configure cdn_abvip_axi3_master.svp with an old abvip module name. This file is kept for backward compatibility and will be removed in a future release.
axi3_slave.sv	Configure cdn_abvip_axi3_slave.svp with an old abvip module name. This file is kept for backward compatibility and will be removed in a future release.

axi3_defines.svh	AXI3 definition file
------------------	----------------------

The `example` folder contains examples for running AXI monitor in different configurations. Each subdirectory contains a README file that describes the steps for running the example and the content of that folder.

Table A-3 Contents of the Example Folder in AXI3 Installation Directory

README	Lists the contents of example directory.
CLEAN	Contains a command script that removes all intermediate log files.
demo.sh	Is an executable for running a simple ABVIP.
bridge_duv	Contains Verilog example for verifying AXI bridge using AXI3 ABVIP in formal.
bridge_duv_vhdl	Contains VHDL example for verifying AXI bridge using AXI3 ABVIP in formal.
master_duv	Contains Verilog example for verifying AXI master using AXI3 ABVIP in formal.
master_duv_vhdl	Contains VHDL example for verifying AXI master using AXI3 ABVIP in formal.
slave_duv	Contains Verilog example for verifying AXI slave using AXI3 ABVIP in formal.
slave_duv_vhdl	Contains VHDL example for verifying AXI slave using AXI3 ABVIP in formal.
slave_duv_sim	Contains Verilog example for verifying AXI bridge using AXI3 ABVIP in simulation.
axi3_wrapper	Contains Verilog example for verifying AXI master/slave using AXI3 ABVIP wrapper module.
examples_regression.vsim	Input file to run all examples in Enterprise manager.
rtl	Contains AXI master, slave, and bridge designs in Verilog and VHDL that are used in the examples.
xcelium	Contains Verilog example for verifying AXI3 ABVIP using Xcelium.

AXI3 Monitor Pin-Level Interface

The pin-level interface is a collection of HDL signals contained in an HDL module and connected to the DUV. A description of the pin-level interface signals for the AXI3 monitor is listed in [Table A-4](#).

Table A-4 The AXI3 Monitor Pin Interface

Pin	Source	Description
-----	--------	-------------

acclk	Clock source	System clock
aresetn	Reset controller	System reset--active low
awid[ID_WIDTH-1:0]	Master	Write address ID--AWID
awaddr[ADDR_WIDTH-1:0]	Master	Write address--AWADDR
awlen[LEN_WIDTH-1:0]	Master	Burst length--AWLEN
awsiz[SIZE_WIDTH:0]	Master	Burst size--AWSIZE
awburst[BURST_WIDTH:0]	Master	Burst type--AWBURST
awlock[LOCK_WIDTH:0]	Master	Lock type--AWLOCK
awcache[CACHE_WIDTH:0]	Master	Cache type--AWCACHE
awprot[PROT_WIDTH:0]	Master	Protection type--AWPROT
awuser[AWUSER_WIDTH-1:0]	Master	Write address channel sideband signal
awvalid	Master	Write address valid-AWVALID
awready	Slave	Write address ready-AWREADY
wid[ID_WIDTH-1:0]	Master	Write ID tag--WID
wdata[DATA_WIDTH-1:0]	Master	Write data--WDATA
wstrb[DATA_WIDTH/8-1:0]	Master	Write strobes--WSTRB
wuser[WUSER_WIDTH-1:0]	Master	Write data channel sideband signal
wlast	Master	Write last--WLAST
wvalid	Master	Write valid--WVALID
wready	Slave	Write ready--WREADY
bid[ID_WIDTH-1:0]	Slave	Response ID--BID
bresp[BRESP_WIDTH-1:0]	Slave	Write response--BRESP
buser[BUSER_WIDTH-1:0]	Slave	Write response channel sideband signal
bvalid	Slave	Write response valid--BVALID
bready	Master	Response ready--BREADY
arid[ID_WIDTH-1:0]	Master	Read address ID--ARID
araddr[ADDR_WIDTH-1:0]	Master	Read address--ARADDR
arlen[LEN_WIDTH-1:0]	Master	Burst length--ARLEN

arsize[SIZE_WIDTH-1:0]	Master	Burst size--ARSIZE
arburst[BURST_WIDTH-1:0]	Master	Burst type--ARBURST
arlock[LOCK_WIDTH-1:0]	Master	Lock type--ARLOCK
arcache[CACHE_WIDTH-1:0]	Master	Cache type--ARCACHE
arprot[PROT_WIDTH-1:0]	Master	Protection type--ARPROT
aruser[ARUSER_WIDTH-1:0]	Master	Read address channel sideband signal
arvalid	Master	Read address valid--ARVALID
arready	Slave	Read address ready--ARREADY
rid[ID_WIDTH-1:0]	Slave	Read ID tag--RID
rdata[DATA_WIDTH-1:0]	Slave	Read data--RDATA
rresp[RRESP_WIDTH-1:0]	Slave	Read response--RRESP
ruser[RUSER_WIDTH-1:0]	Slave	Read Data Channel sideband signal
rlast	Slave	Read last--RLAST
rvalid	Slave	Read valid--RVALID
rready	Slave	Read ready--RREADY
csysreq	Clock controller	System low-power request--CSYSREQ
csysack	Peripheral device	Low-power request acknowledgment--CSYSACK
cactive	Peripheral device	Clock active--CACTIVE

AXI3 ABVIP Debug Signals

AXI3 ABVIP supports the following debug or utility signals for transaction tracking. For an ongoing data beat or response, the corresponding control information will be valid for the following signals. In case of data or response before control, these signals are not valid and hold the value 0.

Signal Name	Description
waddr	awaddr for current write data beat. Valid only if control is previously received or ongoing.
wlen	awlen for current write data beat. Valid only if control is previously received or ongoing.
wsiz	awsiz for current write data beat. Valid only if control is previously received or ongoing.

wburst	awburst for current write data beat. Valid only if control is previously received or ongoing.
baddr	awaddr for current write response. Valid only if control is previously received or ongoing.
blen	awlen for current write response. Valid only if control is previously received or ongoing.
bsize	awsize for current write response. Valid only if control is previously received or ongoing.
bburst	awburst for current write response. Valid only if control is previously received or ongoing.
raddr	araddr for current read data beat
rlen	arlen for current read data beat
rsize	arsize for current read data beat
rburst	arburst for current read data beat

AXI3 Monitor Parameters


The AXI3 ABVIP is highly configurable. The monitor uses some parameters (Verilog localparam) for defining burst types, sizes, and access types. These parameters are internal to the AXI monitor and are not supposed to change. However, you can change the parameters listed in [Table A-5](#).

Table A-5 AXI3 Monitor Parameters


Parameter	Description
MAX_WAIT_CYCLES_W_B	The maximum number of cycles for which write response is low after wvalid goes high.
ADDR_WIDTH	Address bus width. Default value is 32.
DATA_WIDTH	Data bus width. Default value is 64.
ID_WIDTH	The number of bits needed to capture the number of IDs supported. Default value is 2.
LEN_WIDTH	Number of bits needed to capture the length of data per request. Default value is 4.
ALL_STROBES_HIGH_ON	Whether all strobes are set to high. The default value is 0, which means this parameter is not set.

ALLOW_SPARSE_STROBE	When this parameter is set, manager ABVIP drives <code>wstrb</code> bits to 0 for one or more valid byte lanes. The default value of this parameter is 1. This parameter is effective only if parameter <code>BYTE_STROBE_ON</code> is set to 1.
BYTE_STROBE_ON	Whether write byte strobe (WSTRB) checking is required. The default value is 0, which means byte strobe checking is not required.
COVERAGE_ON	Whether coverage generation is set. The default value is 0, which is no coverage.
DATA_BEFORE_CONTROL_ON	Whether data before control is supported. The default value is 1, which means data before control is supported.
DATA_ACCEPT_WITH_OR_AFTER_CONTROL	Whether slave can accept data with or after control. The default value is 0, which means data can be accepted before control.
DEADLOCK_CHKS_ON	Liveness properties are now created for the scenarios whose <code>MAX_WAIT_CYCLES_*</code> parameter is 0. Setting this parameter to 1 enables deadlock checking mode in ABVIP, that is, only liveness asserts are enabled and fairness constraints are disabled. The default value is 0. Also, if both <code>CONFIG_LIVENESS</code> and <code>DEADLOCK_CHKS_ON</code> parameters are enabled then <code>DEADLOCK_CHKS_ON</code> takes the priority. It disables all fairness constraints and enables only deadlock checks. Define macro <code>JG_ABVIP_STRONG_SEMANTICS</code> to enable strong semantics for liveness properties when using sv09 and above.
EXCL_ACCESS_ON	Whether exclusive access is supported. The default value is 0, which means no exclusive access.
RECM_CHECKS_EXCL_ON	Enables recommended behavior in spec for Exclusive access. Default value is 1.
LOCKED_ACCESS_ON	Whether locked access is supported. The default value is 0, which means no locked access.
LOCKED_ACCESS_SWP_ONLY	Executes SWP instruction using locked access. This enables a read (locked) followed by write (unlocked) with the same size. Default value is 0.

RECM_CHECKS_LOCK_ON	Enable recommended checks for locked access. Default value is 1.
READ_INTERLEAVE_ON	Whether read data interleaving is supported. The default value is 1, which means read interleave mode is supported.
WRITE_INTERLEAVE_ON	Whether write data interleaving is supported. Default value is 1, which means write interleave mode is supported.
INTL_DEPTH	Number of write data bursts that can interleave. This value takes effect only if <code>WRITE_INTERLEAVE_ON</code> is set. Default value is <code>MAX_PENDING_WR</code> .
READ_RESP_IN_ORDER_ON	Whether read response in order is supported. The default value is 0, which means read response can come in any order.
READ_DATA_FULL_STRB_ON	All byte lanes are active for Read data
READ_DATA_MASK_ON	RDATA X-propagation and stability properties apply to valid byte lanes only
WRITE_RESP_IN_ORDER_ON	Whether write response in order is supported. The default value is 0, which means write response can come in any order.
WRITE_RESP_AFTER_CONTROL_ON	Normally, AXI3 spec allows write response to follow data, irrespective of control. Set this parameter to enable write response to follow control.
RST_CHECKS_ON	Whether reset checks are supported. The default value is 0, which means no reset checks.
X_CHECKS_ON	Support of X-checks. The default value is 0, meaning no X-checks. <div> Use the Jasper elaborate switch “<code>enable_sva_isunknown</code>” for X-Prop analysis when this parameter is enabled.</div>
LOW_POWER_ON	Whether low power interface checking is required. The default value is 0, which means no low power interface checking. If low power mode is not supported, low power interface signals such as <code>CSYSREQ</code> , <code>CACTIVE</code> , and <code>CSYSACK</code> must be tied to 1'b1.

MAX_WAIT_CYCLES_ON	<p>If set to 1, all MAX_WAIT_CYCLES_* parameters can be used. If set to 0, the MAX_WAIT_CYCLES_* parameters are assigned a value of infinity by using liveness properties. This value means, for instance, if an AWVALID goes high, AWREADY must eventually go high.</p> <div>  Define macro JG_ABVIP_STRONG_SEMANTICS to enable strong semantics for liveness properties when using sv09 and above. </div>
MAX_WAIT_CYCLES_AW	The maximum number of cycles for which AWREADY is low after AWVALID goes high.
MAX_WAIT_CYCLES_W	The maximum number of cycles for which WREADY is low after WVALID goes high.
MAX_WAIT_CYCLES_B	The maximum number of cycles for which BREADY is low after BVALID goes high.
MAX_WAIT_CYCLES_AR	The maximum number of cycles for which ARREADY is low after ARVALID goes high.
MAX_WAIT_CYCLES_R	The maximum number of cycles for which RREADY is low after RVALID goes high.
MAX_WAIT_CYCLES_AW_W	The maximum number of cycles for which write signals(WVALID, WREADY) are low after AWVALID goes high
MAX_WAIT_CYCLES_W_AW	The maximum number of cycles for which write address channel signals(AWVALID, AWREADY) are low after WVALID goes high
MAX_WAIT_CYCLES_AW_B	The maximum number of cycles for which write response is low after AWVALID goes high
MAX_WAIT_CYCLES_AR_R	The maximum number of cycles for which read response is low after ARVALID goes high
MAX_PENDING	The maximum number of pending read or write requests for which the response or data are incomplete.
MAX_PENDING_WR	The maximum number of pending write requests for which the response or data are incomplete. The value of this parameter overrides MAX_PENDING.
MAX_PENDING_RD	The maximum number of pending read requests for which the response or data are incomplete. The value of this parameter overrides MAX_PENDING.

RMAXLEN	Maximum Number of beats in a read burst. Default value is 16.
WMAXLEN	Maximum Number of beats in a write burst. Default value is 16.
MAX_PENDING_EXCL	The maximum number of pending exclusive requests for which the response or data are incomplete.
MAX_LOCKED_SEQ	The maximum number of locked transactions in a locked sequence. When <code>RECM_CHECKS_LOCK_ON</code> is set, this parameter is ignored.
CDNS_ABVIP_STOP_OVERFLOW	Enables ABVIP assumptions to prevent table overflow. Default value is 1.
READONLY_INTERFACE	Excludes all properties that are not for read accesses. Default value is 0.
WRITEONLY_INTERFACE	Excludes all properties that are not for write accesses. Default value is 0.
PARAM_CHECKS_ON	Enable properties that check if parameter setting is correct. Default value is 1.
USER_SIGNALS_ON	Enable optional USER sideband signal properties. Default value is 0.
AWUSER_WIDTH	Width of the user AW sideband field. Default value is 32.
WUSER_WIDTH	Width of the user W sideband field. Default value is 32.
BUSER_WIDTH	Width of the user B sideband field. Default value is 32.
ARUSER_WIDTH	Width of the user AR sideband field. Default value is 32.
RUSER_WIDTH	Width of the user R sideband field. Default value is 32.
ERROR_CHECK_ON	Allow for errors: address not aligned, so they can be checked. Default value is 0.
CONFIG_CSR_INTERFACE	This parameter is used to connect ABVIP with Jasper CSR App. The default value of this parameter is 0. You can set the parameter value to 1 to enable ABVIP with CSR checkers.

CONFIG_CSR_FEED_EARLY_W	<p>Tune internal CSR logic behavior: Set to 0 feed write data to CSR checker on write response transaction Set to 1 feeds write data to CSR checker on write data transaction Default value is 0.</p> <div>  Arrays of size MAX_PENDING_WR*WLAXLEN*DATA_WIDTH will be created in ABVIP when this parameter is 0. -disable_auto_bbox or -no_bbox_m or -no_bbox_i or -bbox_a switch needs to be added in elaborate command to avoid black-boxing of these arrays in ABVIP. Elaboration time will increase in this mode. </div>
CONFIG_CSR_FEED_ON_BVALID	<p>Tune internal CSR logic behavior, Set to 0 will feed write data to CSR checker on write response channel <code>bvalid</code> and <code>bready</code> assertion. Set to 1 will feed write data to CSR checker on write response channel <code>bvalid</code> assertion. Default value is 0.</p>
CONFIG_CSR_FEED_ON_RVALID	<p>Tune internal CSR logic behavior, Set to 0 will feed read data to CSR checker on read response channel <code>rvalid</code> and <code>rready</code> assertion. Set to 1 will feed read data to CSR checker on read response channel <code>rvalid</code> assertion. Default value is 0.</p>
CONFIG_CSR_DISCARD_ON_ERROR	<p>Tune internal CSR logic behavior to handle the error response (SLVERR/DECERR) scenarios. Set to 0 sends the erroneous read/write data to the CSR checker. Set to 1 tune Read/Write behavior according to the following description:</p> <ul style="list-style-type: none"> • For Read: Discard the read data beat with an error response to the CSR checker • For Write: Discard the all write data beat of the burst with an error response to the CSR checker

EXCL_ACCESS_WRITE_MATCH_EX_READ	Set to 0 to enable the master to generate the illegal transactions in the exclusive access. By default, it will generate the exclusive write transaction matching with exclusive read
EXCL_ACCESS_SLAVE_ON	By default it is 1. Set to 0 to disable exclusive access of slave
EXCL_ACCESS_WR_RESP_OKAY_ALLOWED	Set to 1 to allow that slave response can send EXOKAY or OKAY response in case of valid exclusive access. By default, It is set to 0 means slave can send only exokay response

AXI3 Checks

Compliance Checks

Table A-6 describes the SVA compliance checks for the master and slave in the AXI monitor. The properties have been grouped into sub-categories based on different aspects of the AXI protocol. The property naming conventions are also shown in the table. You can view the complete hierarchical property name in Property Table pane of Jasper GUI

Table A-6 AXI Monitor Compliance Checks

Property Name	Description	AXI Spec
1.1 Master Stable		
master_ar_arvalid_stable	The ARVALID signal must remain stable until ARREADY goes high.	AXI Spec v1.0, Section 3.1, pg 3-2 to pg 3-4
master_ar_arvalid_throttle	The component driving <code>arready</code> (SLAVE) should NEVER issue <code>arready</code> when <code>tbl</code> is <code>rd_tbl_full</code> .	None
master_ar_arid_stable	The ARID signal must remain stable when ARVALID is high until ARREADY goes high.	AXI Spec v1.0, Section 3.1, pg 3-2 to pg 3-4

master_ar_araddr_stable	The ARADDR signal must remain stable when ARVALID is high until ARREADY goes high.	AXI Spec v1.0, Section 3.1, pg 3-2 to pg 3-4
master_ar_arlen_stable	The ARLEN signal must remain stable when ARVALID is high until ARREADY goes high.	AXI Spec v1.0, Section 3.1, pg 3-2 to pg 3-4
master_ar_arsize_stable	The ARSIZE signal must remain stable when ARVALID is high until ARREADY goes high.	AXI Spec v1.0, Section 3.1, pg 3-2 to pg 3-4
master_ar_arburst_stable	The ARBURST signal must remain stable when ARVALID is high until ARREADY goes high.	AXI Spec v1.0, Section 3.1, pg 3-2 to pg 3-4
master_ar_arlock_stable	The ARLOCK signal must remain stable when ARVALID is high until ARREADY goes high.	AXI Spec v1.0, Section 3.1, pg 3-2 to pg 3-4
master_ar_arprot_stable	The ARPROT signal must remain stable when ARVALID is high until ARREADY goes high.	AXI Spec v1.0, Section 3.1, pg 3-2 to pg 3-4

master_ar_arcache_stable	The ARCACHE signal must remain stable when ARVALID is high until ARREADY goes high.	AXI Spec v1.0, Section 3.1, pg 3-2 to pg 3-4
master_ar_aruser_stable	The ARUSER should remain stable if ARVALID is high until and ARREADY goes high	AXI Spec v1.0, Section 3.1, pg 3-2 to pg 3-4
master_aw_awvalid_stable	The AWVALID signal must remain stable until AWREADY goes high.	AXI Spec v1.0, Section 3.1, pg 3-2 to pg 3-4
master_aw_awvalid_throttle	The component driving awready (MASTER) should NEVER issue awready when there is a collision in case of DBC (different ID) on the same clock cycle AND there is no vacant slot to accommodate the NEW CONTROL transaction.	None
master_aw_awid_stable	The AWID signal must remain stable when AWVALID is high until AWREADY goes high.	AXI Spec v1.0, Section 3.1, pg 3-2 to pg 3-4
master_aw_awaddr_stable	The AWADDR signal must remain stable when AWVALID is high until AWREADY goes high.	AXI Spec v1.0, Section 3.1, pg 3-2 to pg 3-4

master_aw_awlen_stable	The AWLEN signal must remain stable when AWVALID is high until AWREADY goes high.	AXI Spec v1.0, Section 3.1, pg 3-2 to pg 3-4
master_aw_awsizestable	The AWSIZE signal must remain stable when AWVALID is high until AWREADY goes high.	AXI Spec v1.0, Section 3.1, pg 3-2 to pg 3-4
master_aw_awburst_stable	The AWBURST signal must remain stable when AWVALID is high until AWREADY goes high.	AXI Spec v1.0, Section 3.1, pg 3-2 to pg 3-4
master_aw_awlock_stable	The AWLOCK signal must remain stable when AWVALID is high until AWREADY goes high.	AXI Spec v1.0, Section 3.1, pg 3-2 to pg 3-4
master_aw_awprot_stable	The AWPROT signal must remain stable when AWVALID is high until AWREADY goes high.	AXI Spec v1.0, Section 3.1, pg 3-2 to pg 3-4
master_aw_awcache_stable	The AWCACHE signal must remain stable when AWVALID is high until AWREADY goes high.	AXI Spec v1.0, Section 3.1, pg 3-2 to pg 3-4

master_aw_awuser_stable	The AWUSER should remain stable if AWVALID is high until and AWREADY goes high	AXI Spec v1.0, Section 3.1, pg 3-2 to pg 3-4
master_w_wvalid_stable	The WVALID signal must remain stable until WREADY goes high.	AXI Spec v1.0, Section 3.1, pg 3-2 to pg 3-4
master_w_wdata_stable	The WDATA signal must remain stable when WVALID is high WREADY goes high.	AXI Spec v1.0, Section 3.1, pg 3-2 to pg 3-4
master_w_wstrb_stable	The WSTRB signal must remain stable when WVALID is high WREADY goes high.	AXI Spec v1.0, Section 3.1, pg 3-2 to pg 3-4
master_w_wid_stable	The WID signal must remain stable when WVALID is high WREADY goes high.	AXI Spec v1.0, Section 3.1, pg 3-2 to pg 3-4
master_w_wuser_stable	The WUSER should remain stable if WVALID is high until and WREADY goes high	AXI Spec v1.0, Section 3.1, pg 3-2 to pg 3-4

master_w_wlast_stable	The WLAST signal must remain stable when WVALID is high WREADY goes high.	AXI Spec v1.0, Section 3.1, pg 3-2 to pg 3-4
master_w_wvalid_throttle	Write data overflow, which means the number of outstanding write data transfers is more than MAX_PENDING_WR.	None
1.2 Master Read		
master_ar_arlock_excl_throttle	This assertion checks that the exclusive table never overflows. This ensures protocol verification results are valid to be used as an assertion only	None
master_arvalid_low_when_wr_only_inf	arvalid should be low when write only interface is enabled (WRITEONLY_INTERFACE = 1)	
slave_rvalid_low_when_wr_only_inf	rvalid should be low when write only interface is enabled (WRITEONLY_INTERFACE = 1)	
master_ar_arlen_less_than_RMAXLEN	arlen should be less than RMAXLEN parameter value	
master_ar_araddr_wrap_aligned	WRAP bursts should be aligned to the transfer size.	AXI Spec v1.0, Section 4.4.3, pg 4-6
master_ar_araddr_wrap_arlen	WRAP bursts should be of length 2,4,8 or 16.	AXI Spec v1.0, Section 4.4.3, pg 4-6
master_ar_araddr_never_cross_4K_boundary	Read burst must not cross 4K boundary.	AXI Spec v1.0, Section 4.1, pg 4-2
master_ar_arburst_no_reserved	Read burst cannot take reserved value 2'b11.	AXI Spec v1.0, Section 4.4, pg 4-5

master_ar_arsize_lte_datawidth	Size of any transfer must not exceed the data bus width of the components in the transaction.	AXI Spec v1.0, Section 4.3, pg 4-4
master_ar_arcache_no_ra_wa_for_uncacheable	Read Allocate(RA) bit and Write Allocate(WA) bit of arcache, arcache[2] and arcache[3] must not be HIGH if the modifiable bit arcache[1] is low.	AXI Spec v1.0, Section 5.1, pg 5.2
master_ar_arlock_no_reserved	The arlock should not take the reserved value.	AXI Spec v1.0, Section 6.1, pg 6-2
master_r_ready_wait_cycles	The maximum number of cycles for which RREADY remains low after RVALID going high must not exceed the MAX_WAIT_CYCLES_R cycles specified.	None
master_r_ready_eventually	All read data must be eventually accepted.	None
master_ar_rready_wait_cycles	All read data must be accepted within MAX_WAIT_CYCLES_AR_R cycles.	None
1.3 Master Write		
master_awvalid_low_when_rd_only_inf	awvalid should be low when read only interface is enabled (READONLY_INTERFACE = 1)	
master_wvalid_low_when_rd_only_inf	wvalid should be low when read only interface is enabled (READONLY_INTERFACE = 1)	
slave_bvalid_low_when_rd_only_inf	bvalid should be low when read only interface is enabled (READONLY_INTERFACE = 1)	
master_aw_awlen_less_than_maxlen	awlen should be less than WMAXLEN parameter value	
master_aw_awaddr_wrap_aligned	The start address must be aligned to the size of the transfer.	AXI Spec v1.0, Section 4.4.3, pg 4-6

master_aw_awaddr_wrap_awlen	The length of the burst must be 2,4,8 or 16.	AXI Spec v1.0, Section 4.4.3, pg 4-6
master_aw_awaddr_never_cross_4K_boundary	Bursts must not cross 4KB boundaries to prevent them from crossing boundaries between slaves.	AXI Spec v1.0, Section 4.1, pg 4-2
master_aw_awburst_no_reserved	Burst signal cannot take the reserved value 2'b11.	AXI Spec v1.0, Section 4.4, Table 4-3, pg 4-5
master_aw_awsz_lte_datawidth	The size of any transfer must not exceed the data bus width of the components in the transaction.	AXI Spec v1.0, Section 4.3, pg 4-4
master_aw_awcache_no_ra_wa_for_uncacheable	Write Cache RA and WA bits must de-asserted if modifiable bit is not set.	AXI Spec v1.0, Section 5.1, pg 5.2
master_aw_awlock_no_reserved	The awlock should not take the reserved value (2'b11).	AXI Spec v1.0, Section 6.1, pg 6-2
master_aw_w_vvalid_no_dbc	If data before control is not allowed, wvalid has to follow awvalid and awready.	AXI Spec v1.0, Section 3.1.2, pg 3-3

master_w_order_wid_first_beat	The order in which the slave receives the first data item of each transaction must be the same as the order in which it receives the address for the transaction.	AXI Spec v1.0, Section 8.5, pg 8-6
master_w_order_wid_no_il	If a slave does not support write data interleaving, the master must issue the data of write transactions in the same order in which it issues the transaction address.	AXI Spec v1.0, Section 8.4, pg 8-5
master_w_interleave_depth	Number of outstanding write data transfers should be less than or equal to INTL_DEPTH parameter	None
master_aw_w_wlast_exact_len	wlast should be asserted in the last beat of a write transaction.	AXI Spec v1.0, Section 3.1.2 - pg 3-4
master_w_aw_wstrb_valid_dbc	In case where data beats are received before control, check the strobes of all the previously received beats when control is received.	ARM IHI 0022D, Section A3.4.1 on pg A3.46-A3.47
master_w_aw_wstrb_valid_curr_dbc	In case where data beats are received before control, check the strobes of the beat received in the same cycle as control.	ARM IHI 0022D, Section A3.4.1 on pg A3.46-A3.47
master_w_aw_wstrb_valid_curr	check strobe received in current cycle whose control is already received	ARM IHI 0022D, Section A3.4.1 on pg A3.46-A3.47

master_w_aw_wstrb_valid_collision	check strobe for the first beat of a transfer which comes in the same cycle as control for the same transfer	ARM IHI 0022D, Section A3.4.1 on pg A3.46-A3.47
master_w_wstrb_all_active	Checks if the master is using full length data transfer.	None
master_aw_awsizelanes_active	When a master performs full data bus width write transactions, AWSIZE must be equal to WRITE_TRSIZE	None
master_b_bready_wait_cycles	The maximum number of cycles for which bready remains low after bvalid going high must not exceed the MAX_WAIT_CYCLES_B specified.	None
master_w_wvalid_eventually	All pending write data requests must start eventually.	None
master_w_wvalid_wait_cycles	All pending write data requests must start within MAX_WAIT_CYCLES_AW_W cycles.	None
master_b_bready_eventually	All write responses must be eventually accepted.	None
master_b_bready_wait_cycles	Master must assert bready within MAX_WAIT_CYCLES_AW_B cycles if there are pending write response	None

1.4 Master Atomic Access

master_ar_excl_araddr_aligned	The address of exclusive access must be aligned to the total number of bytes in the transaction.	AXI Spec v1.0, Section 6.2.4, pg 6-4
master_ar_excl_arlen_arsize_correct_bytes	The number of bytes in an exclusive burst should be 1,2,4,8,16,32,64 or 128.	AXI Spec v1.0, Section 6.2.4, pg 6-4
master_access_pwdata_xcheck	pwdata remains valid until the transfer completes at the end of the access phase.	v1.0, Section 2.1.1 -2

master_ar_excl_arcache_low	Exclusive access burst cannot be cacheable and bufferable.	AXI Spec v1.0, Section 6.2.4, pg 6-5
master_aw_excl_awcache_low	Exclusive access burst cannot be cacheable and bufferable.	ARM IHI 0022F: Section A7.2.4 on pg A7-99
master_aw_excl_control_signals_match	The control signals for the read and write portions of the exclusive access must be identical.	AXI Spec v1.0, Section 6.2.4, pg 6-4
master_ar_aw_excl_exwrite_after_exread	The master must not commence the write portion of exclusive access until the read portion is complete.	AXI Spec v1.0, Section 6.2.2, pg 6-4
master_ar_aw_no_exwrite_while_exread	The master must not commence the write portion of exclusive access until the read portion is complete.	AXI Spec v1.0, Section 6.2.2, pg 6-4
master_aw_excl_awlen_awsizesize_correct_bytes	The number of bytes to be transferred in an exclusive access burst must be a power of 2, that is 1,2,4,8,16,32,64 or 128 bytes.	AXI Spec v1.0, Section 6.2.4, pg 6-4
master_ar_aw_locked_no_pend_trn_at_start	If master starts a locked sequence of either read or write transactions, then it must ensure that no outstanding transactions are waiting to complete.	AXI Spec v1.0, Section 6.3, pg 6-7

master_ar_aw_locked_no_pend_trn_before_unlock	When completing a locked sequence a master must ensure that all previous locked transactions are complete before issuing the final unlocking transaction.	AXI Spec v1.0, Section 6.3, pg 6-7
master_ar_aw_locked_unlock_complete	It must then ensure that the final unlocking transaction has fully completed before any further transactions are commenced.	AXI Spec v1.0, Section 6.3, pg 6-7
master_ar_aw_locked_same_arid_no_dbc	The master must ensure that all transactions within a locked sequence have the same arid value.	AXI Spec v1.0, Section 6.3, pg 6-7
master_ar_aw_locked_same_awid_no_dbc	The master must ensure that all transactions within a locked sequence have the same awid value.	AXI Spec v1.0, Section 6.3, pg 6-7
master_ar_locked_awaddr_never_cross_4K_boundary	Locked sequence should not cross 4K address boundary	AXI Spec v1.0, Section 6.3, pg 6-7
master_ar_locked_araddr_never_cross_4K_boundary	Locked sequence should not cross 4K address boundary	AXI Spec v1.0, Section 6.3, pg 6-7
master_ar_aw_locked_araddr_never_cross_4K_boundary	Locked sequence should not cross 4K address boundary	AXI Spec v1.0, Section 6.3, pg 6-7
master_ar_locked_first_swp_only	During locked access use for SWP only, locked sequence should begin with read only	

master_ar_locked_no_read_swp_in_progress	Once lock is set during SWP instruction, a master must ensure that there is no read transaction in between	
master_aw_locked_write_unlock	For SWP instruction, the lock sequence ends with an unlocking write with the same size as locking read	
master_ar_excl_req_with_wr_ctl_done	The master should not send an exclusive read req when there is already excl write req occurred with same ID	ARM IHI 0022F: Section A7.2.4 on pg A7.99

1.5 Master Write Data Before Control (DBC)

master_aw_w_awlen_dbc_exact_len	If data phase finishes before control, the subsequent control phase must match that data transfer length.	None
master_aw_w_wlast_dbc_exact_len	wlast should be asserted in the last beat of a write transaction.	None
master_aw_awvalid_eventually	All pending write control requests must start eventually.	None
master_aw_awvalid_wait_cycles	All pending write control requests must start within MAX_WAIT_CYCLES_W_AW cycles.	None
master_ar_aw_locked_same_arid_dbc	The master must ensure that all transactions within a locked sequence have the same arid value.	None
master_ar_aw_locked_same_awid_dbc	The master must ensure that all transactions within a locked sequence have the same awid value.	None

1.6 Master Reset

master_rst_arvalid_low_reset	Master should drive arvalid low during reset.	None
master_rst_arvalid_low_reset_released	Master should not drive arvalid at a posedge of aclk after aresetn is high.	None
master_rst_awvalid_low_reset	Master should drive awvalid low during reset.	None
master_rst_awvalid_low_reset_released	Master should not drive awvalid at a posedge of aclk after aresetn is high.	None
master_rst_wvalid_low_reset	Master should drive wvalid low during reset.	None
master_rst_wvalid_low_reset_released	Master should not drive wvalid at a posedge of aclk after aresetn is high.	None

master_rst_csysreq_high_reset_released	Master should drive <code>csysreq</code> high after <code>aresetn</code> is high	None
1.7 Master X-Checks		
master_xcheck_arvalid	When reset is high, <code>arvalid</code> should not have X or Z value.	None
master_xcheck_arid	When <code>arvalid</code> is asserted, <code>arid</code> should not have X or Z value on any bit.	None
master_xcheck_araddr	When <code>arvalid</code> is asserted, <code>arid</code> should not have X or Z value on any bit.	None
master_xcheck_arlen	When <code>arvalid</code> is asserted, <code>arlen</code> should not have X or Z value on any bit.	None
master_xcheck_arsize	When <code>arvalid</code> is asserted, <code>arsize</code> should not have X or Z value on any bit.	None
master_xcheck_arburst	When <code>arvalid</code> is asserted, <code>arburst</code> should not have X or Z value on any bit.	None
master_xcheck_arlock	When <code>arvalid</code> is asserted, <code>arlock</code> should not have X or Z value on any bit.	None
master_xcheck_arprot	When <code>arvalid</code> is asserted, <code>arprot</code> should not have X or Z value on any bit.	None
master_xcheck_aruser	When <code>arvalid</code> is asserted, <code>aruser</code> should not have X or Z value on any bit.	None
master_xcheck_arcache	When <code>arvalid</code> is asserted, <code>arcache</code> should not have X or Z value on any bit.	None
master_xcheck_rready	When <code>rvalid</code> is asserted, <code>rready</code> should not have X or Z value.	None
master_xcheck_awvalid	When reset is asserted, <code>awvalid</code> should not have X or Z value.	None
master_xcheck_awid	When <code>awvalid</code> is asserted, <code>awid</code> should not have X or Z value on any bit.	None
master_xcheck_awaddr	When <code>awvalid</code> is asserted, <code>awaddr</code> should not have X or Z value on any bit.	None
master_xcheck_awlen	When <code>awvalid</code> is asserted, <code>awlen</code> should not have X or Z value on any bit.	None
master_xcheck_awsiz	When <code>awvalid</code> is asserted, <code>awsiz</code> should not have X or Z value on any bit.	None
master_xcheck_awburst	When <code>awvalid</code> is asserted, <code>awburst</code> should not have X or Z value on any bit.	None

master_xcheck_awlock	When awvalid is asserted, awlock should not have X or Z value on any bit.	None
master_xcheck_awprot	When awvalid is asserted, awprot should not have X or Z value on any bit.	None
master_xcheck_awuser	When awvalid is asserted, awuser should not have X or Z value on any bit.	None
master_xcheck_awcache	When awvalid is asserted, awcache should not have X or Z value on any bit.	None
master_xcheck_wvalid	When reset is asserted, wvalid should not have X or Z value.	None
master_xcheck_wdata	When wvalid is asserted, wdata should not have X or Z value on any bit.	None
master_xcheck_wstrb	When wvalid is asserted, wstrb should not have X or Z value on any bit.	None
master_xcheck_wuser	When wvalid is asserted, wuser should not have X or Z value on any bit.	None
master_xcheck_wid	When wvalid is asserted, wid should not have X or Z value on any bit.	None
master_xcheck_wlast	When wvalid is asserted, wlast should not have X or Z value.	None
master_xcheck_bready	When bvalid is asserted, bready should not have X or Z value.	None

1.8 Master Internal Table Checks

master_ar_tbl_no_overflow	This assertion checks that the number of outstanding read requests should not be greater than MAX_PENDING_RD	None
master_aw_tbl_no_overflow	This assertion checks that the number of write requests should not be greater than MAX_PENDING_WR	None
master_w_tbl_no_overflow	This assertion checks that number of outstanding write data transfers should not be greater than MAX_PENDING_WR	None

1.9 Slave Stable

slave_r_rvalid_stable	The RVALID signal must remain stable until RREADY goes high.	AXI Spec v1.0, Section 3.1, pg 3-2 to pg 3-4
slave_r_rlast_stable	The RLAST signal must remain stable when RVALID is high until RREADY goes high.	AXI Spec v1.0, Section 3.1, pg 3-2 to pg 3-4
slave_r_rdata_stable	The RDATA signal must remain stable when RVALID is high until RREADY goes high.	AXI Spec v1.0, Section 3.1, pg 3-2 to pg 3-4
slave_r_rid_stable	The RID signal must remain stable when RVALID is high until RREADY goes high.	AXI Spec v1.0, Section 3.1, pg 3-2 to pg 3-4
slave_r_rresp_stable	The RRESP signal must remain stable when RVALID is high until RREADY goes high.	AXI Spec v1.0, Section 3.1, pg 3-2 to pg 3-4
slave_r_ruser_stable	The <code>RUSER</code> signal must remain stable when <code>RVALID</code> is high until <code>RREADY</code> goes high.	AXI Spec v1.0, Section 3.1, pg 3-2 to pg 3-4

slave_b_bvalid_stable	The BVALID signal must remain stable until BREADY goes high.	AXI Spec v1.0, Section 3.1, pg 3-2 to pg 3-4
slave_b_bid_stable	The BID signal must remain stable when BVALID is high until BREADY goes high.	AXI Spec v1.0, Section 3.1, pg 3-2 to pg 3-4
slave_b_buser_stable	The BUSER signal must remain stable when BVALID is high until BREADY goes high.	AXI Spec v1.0, Section 3.1, pg 3-2 to pg 3-4
slave_b_bresp_stable	The BRESP signal must remain stable when BVALID is high until BREADY goes high.	AXI Spec v1.0, Section 3.1, pg 3-2 to pg 3-4
1.10 Slave Read		
slave_r_rid_match	The RID value returned from the slave with read data, must match the ARID value provided with a currently outstanding read request.	AXI Spec v1.0, Section 3.1.5, pg 3-4
slave_r_rid_ordered	In a sequence of read transactions with different ARID values, the slave can return read data that is out-of-order with respect to the order in which read requests are received. In case read data interleaving is not supported, RID should not change before end of read data.	AXI Spec v1.0, Section 8.3, pg 8-4
slave_r_rvalid_order	The slave must return RRESP in the same order that the read request has been issued.	None

slave_r_rlast_exact_len	The slave must assert the rlast signal when it drives the final read transfer in the burst.	AXI Spec v1.0, Section 3.1.5, pg 3-4
slave_r_rdata_full_strb	When READ_DATA_FULL_STRB_ON is set, a slave must ensure that read strobe bits are all 1s	
slave_ar_arready_wait_cycles	The maximum number of cycles for which ARREADY remains low after ARVALID goes high must not exceed the MAX_WAIT_CYCLES_AR cycles specified.	None
slave_r_rvalid_wait_cycles	All pending read requests must start within MAX_WAIT_CYCLES_AR_R cycles.	None
slave_ar_arready_eventually	All read requests must be eventually accepted.	None
slave_r_rvalid_eventually	All pending read requests must start eventually.	None
slave_ar_arready_throttle	This assertion checks that the table never overflows. This ensures protocol verification results are valid. To be used as an assertion only.	None
slave_aw_awready_throttle_cnstr	Checker should never overflow otherwise the results become invalid. To be used as an assertion only.	None
slave_w_wready_throttle	Write data overflows, which means the number of outstanding write data transfers is more than MAX_PENDING_WR.	None

2.0 Slave Write

slave_b_bid_match	The slave can assert the bvalid signal only when it drives a valid write response(both address and data have been accepted).	AXI Spec v1.0, Section 3.1.3 - pg 3-4
slave_b_bvalid_order	The slave must return BRESP in the same order that the write request has been issued.	None
slave_aw_awready_wait_cycles	The maximum number of cycles for which awready remains low after awvalid goes high must not exceed the MAX_WAIT_CYCLES_AW specified.	None
slave_w_wready_wait_cycles	The maximum number of cycles for which wready remains low after wvalid goes high must not exceed the MAX_WAIT_CYCLES_W specified.	None

slave_aw_awready_eventually	All write requests must be eventually accepted.	None
slave_w_wready_eventually	All write data must be eventually accepted.	None
slave_aw_wready_wait_cycles	All write data must be accepted within MAX_WAIT_CYCLES_AW_W cycles.	None
slave_b_bvalid_eventually	All pending responses must be given by slave eventually.	None
slave_aw_bvalid_wait_cycles	All pending responses must be given by slave within MAX_WAIT_CYCLES_AW_B cycles.	None
slave_w_aw_wready_dbc	If data before control is allowed for slave, it should issue <code>wready</code> with or after <code>awready</code> when <code>DATA_ACCEPT_WITH_OR_AFTER_CONTROL</code> is high	None

2.1 Slave Atomic Access

slave_b_excl_bresp_exokay	The slave must not send an EXOKAY response till exclusive write is successful.	AXI Spec v1.0, Section 6.2.2, pg 6-3; Section 7.1, pg 7-2
slave_r_excl_rresp_no_exokay	The slave must not send an EXOKAY response for a normal read transaction.	AXI Spec v1.0, Section 6.2.2, pg 6-3
slave_b_excl_bresp_no_exokay	The slave must not send an <code>EXOKAY</code> response for a normal write transaction.	AXI Spec v1.0, Section 6.2.2, pg 6-3
slave_b_aw_excl_resp_overlapping_addr	Exclusive Write fails when same location is updated since the exclusive read has happened	ARM IHI 0022F: Section A7.2.4 on pg A7.99

slave_r_ar_excl_resp_exokay	The slave should not send an OKAY and EXOKAY exclusive read resp in different beat of same transaction	ARM IHI 0022F: Section A7.2.4 on pg A7.99
slave_r_ar_excl_resp_okay	The slave should not send an OKAY and EXOKAY exclusive read resp in different beat of same transaction	ARM IHI 0022F: Section A7.2.4 on pg A7.99
slave_b_excl_bresp_no_exokay	The slave must not send an EXOKAY response for a normal write transaction	ARM IHI 0022F: Section A7.2.4 on pg A7.99
slave_r_excl_rresp_okay	The slave must not send an EXOKAY response when slave does not support exclusive access	ARM IHI 0022F: Section A7.2.4 on pg A7.99

2.2 Slave Reset

slave_rst_rvalid_low_reset	Slave should drive rvalid low during reset.	None
slave_rst_bvalid_low_reset	Slave should drive bvalid low during reset.	None

2.3 Slave X-Checks

slave_xcheck_arready	When arvalid is asserted, arready should not have X or Z value.	None
slave_xcheck_rvalid	When reset is high, rvalid should not have X or Z value.	None
slave_xcheck_rdata	When rvalid is asserted, rdata should not have X or Z value on any bit.	None
slave_xcheck_rid	When rvalid is asserted, rid should not have X or Z value on any bit.	None
slave_xcheck_rresp	When rvalid is asserted, rresp should not have X or Z value on any bit.	None

slave_xcheck_ruser	When <code>rvalid</code> is asserted, <code>ruser</code> should not have X or Z value on any bit.	None
slave_xcheck_awready	When <code>awvalid</code> is asserted, <code>awready</code> should not have X or Z value.	None
slave_xcheck_wready	When <code>awvalid</code> is asserted, <code>awready</code> should not have X or Z value.	None
slave_xcheck_bvalid	When <code>reset</code> is high, <code>bvalid</code> should not have X or Z value.	None
slave_xcheck_bid	When <code>bvalid</code> is asserted, <code>bid</code> should not have X or Z value on any bit.	None
slave_xcheck_buser	When <code>bvalid</code> is asserted, <code>buser</code> should not have X or Z value on any bit.	None
slave_xcheck_bresp	When <code>bvalid</code> is asserted, <code>bresp</code> should not have X or Z value on any bit.	None
2.4 Low Power Checks		
slave_lpi_entry_cactive_after_csysreq	<code>cactive</code> can be low only in low power state.	ARM IHI 0022E: Section A9.2.3 on pg A9-108
slave_lpi_csysack_high_when_cactive_low	<code>csysack</code> must not be low until atleast one cycle after <code>cactive</code> is low.	ARM IHI 0022E: Section A9.2.3 on pg A9-108
slave_lpi_csysack_low_when_cactive_high	<code>csysack</code> can be low for as many cycles as are required to complete exit sequence.	ARM IHI 0022E: Section A9.2.5 on pg A9-109
slave_lpi_csysack_low_after_csysreq_low	<code>csysack</code> is low when a request to enter low power state is recognized.	ARM IHI 0022E: Section A9.2.2 on pg A9-107

slave_lpi_csysack_high_after_csysreq_high	csysack is high when a request to exit low power state is recognized.	ARM IHI 0022E: Section A9.2.2 on pg A9-107
master_lpi_csysreq_low	csysreq remains low until handshake with csysack completes.	ARM IHI 0022E: Section A9.2.2 on pg A9-107
master_lpi_csysreq_high	csysreq remains high until handshake with csysack completes.	ARM IHI 0022E: Section A9.2.2 on pg A9-107
slave_lpi_csysack_follows_csysreq_low	csysreq must be acknowledged by csysack.	AXI Spec v1.0, Section 12.2.1, pg 12-4
slave_lpi_csysack_follows_csysreq_high	csysreq must be acknowledged by csysack.	AXI Spec v1.0, Section 12.2.1, pg 12-4
slave_lpi_cactive_high_exit	If csysreq is high then cactive must be high to indicate clk signal is required.	ARM IHI 0022E: Section A9.2.5 on pg A9-109
assert_xcheck_csysreq	When reset is high, csysreq should not have X or Z value.	None
assert_xcheck_csysack	When reset is high, csysack should not have X or Z value.	None

assert_xcheck_cactive	When reset is high, cactive should not have X or Z value.	None
2.5 Slave Internal Table Checks		
slave_ar_tbl_no_overflow	This assertion checks that the number of outstanding read requests should not be greater than <code>MAX_PENDING_RD</code> .	None
slave_aw_tbl_no_overflow	This assertion checks that the number of outstanding write requests should not be greater than <code>MAX_PENDING_WR</code> .	None
slave_w_tbl_no_overflow	This assertion checks that the number of outstanding write data transfers should not be greater than <code>MAX_PENDING_WR</code> .	None
2.6 Other Checks		
assert_ar_excl_tbl_no_overflow	This assertion checks that the exclusive table never overflows.	None

Coverage Checks

The monitor provides coverage checks to collect coverage information for possible scenarios in an AXI-based design. The coverage information is required for complete verification of the DUV.

[Table A-7](#) describes the SVA coverage checks for the AXI monitor.

Table A-7 AXI Monitor Coverage Points

Coverage Assertions	Description
1.1 TRANSITION ID COVERS	
cover_arid_0	arid should take id 0
cover_arid_1	arid should take id 1
cover_arid_other	arid should take all other possible values
cover_awid_0	awid should take id 0
cover_awid_1	awid should take id 1
cover_awid_other	awid should take all other possible values
1.2 TRANSACTION LENGTH COVERS	
cover_arlen_len1	arlen, awlen can be upto 16
cover_arlen_len2	arlen, awlen can be upto 16

cover_arlen_len3	arlen, awlen can be upto 16
cover_arlen_len4	arlen, awlen can be upto 16
cover_arlen_short	arlen, awlen can be upto 16
cover_awlen_len1	arlen, awlen can be upto 16
cover_awlen_len2	arlen, awlen can be upto 16
cover_awlen_len3	arlen, awlen can be upto 16
cover_awlen_len4	arlen, awlen can be upto 16
cover_awlen_short	arlen, awlen can be upto 16
1.3 TRANSACTION BURST COVERS	
cover_arburst_fixed	arburst can be of fixed type
cover_arburst_incr	arburst can be of increment type
cover_arburst_wrap	arburst can be of wrap type
cover_awburst_fixed	awburst can be of fixed type
cover_awburst_incr	awburst can be of increment type
cover_awburst_wrap	awburst can be of wrap type
cover_arburst_wrap_len_2	If read is of wrap type, the arlen can be 1
cover_arburst_wrap_len_4	If read is of wrap type, the arlen can be 3
cover_arburst_wrap_len_8	If read is of wrap type, the arlen can be 7
cover_arburst_wrap_len_16	If read is of wrap type, the arlen can be 15
cover_awburst_wrap_len_2	If write is of wrap type, the awlen can be 1
cover_awburst_wrap_len_4	If write is of wrap type, the awlen can be 3
cover_awburst_wrap_len_8	If write is of wrap type, the awlen can be 7
cover_awburst_wrap_len_16	If write is of wrap type, the awlen can be 15

cover_arburst_fixed_araddr_unaligned	Unaligned read address during fixed burst
cover_arburst_incr_araddr_unaligned	Unaligned read address during increment burst
cover_awburst_fixed_awaddr_unaligned	Unaligned write address during fixed burst
cover_awburst_incr_awaddr_unaligned	Unaligned write address during increment burst
1.4 SLAVE RESPONSE SIGNALLING COVERS	
cover_rresp_okay	rresp can have okay response
cover_rresp_slverr	rresp can have slave error response
cover_rresp_decerr	rresp can have decode error response
cover_bresp_okay	bresp can have okay response
cover_bresp_slverr	bresp can have slave error response
cover_bresp_decerr	bresp can have decode error response
1.5 SEARCHPOINT COVERS	
cover_searchpoint_rd_control	Covers for read control
cover_searchpoint_rd_data	Covers for read data
cover_searchpoint_rd_data_last	Covers for read last data
cover_searchpoint_wr_control	Covers for write control
cover_searchpoint_wr_data	Covers for write data
cover_searchpoint_wr_data_last	Covers for write last data
cover_searchpoint_wr_response	Covers for write response
1.6 READ/WRITE ORDER COVER	
cover_order_control_before_data	Control can come before data
cover_order_control_with_data	Control can come with data
cover_order_control_during_data	Control can come during data
cover_order_control_after_data	Control can come after data

cover_order_dbc_normal_collision1	Data before control and normal write request collision possible
cover_order_rready_before_rvalid	rready can come before rvalid
cover_order_rready_after_rvalid	rready can come after rvalid
cover_order_rready_with_rvalid	rready can come with rvalid
cover_order_wready_before_wvalid	wready can come before wvalid
cover_order_wready_after_wvalid	wready can come after wvalid
cover_order_wready_with_wvalid	wready can come with wvalid
cover_order_bready_before_bvalid	bready can come before bvalid
cover_order_bready_after_bvalid	bready can come after bvalid
cover_order_bready_with_bvalid	bready can come with bvalid
cover_order_wlast_before_wready	wlast can come before wready
cover_order_wlast_after_wready	wlast can come after wready
cover_order_wlast_with_wready	wlast can come with wready
cover_order_awvalid_arvalid_together	awvalid can come with arvalid
1.7 Max Wait Cycles Cover	
cover_max_wait_for_wr_request	Maximum number of cycles for which awready is low and awvalid is high
cover_max_wait_for_wr_data	Maximum number of cycles for which wready is low and wvalid is high
cover_max_wait_for_wr_response	Maximum number of cycles for which bready is low and bvalid is high
cover_max_wait_for_rd_request	Maximum number of cycles for which arready is low and arvalid is high
cover_max_wait_for_rd_response	Maximum number of cycles for which rready is low and rvalid is high
1.8 Read/Write Sequence Covers	
cover_seq_rd_len_1	Read sequence with arlen 0
cover_seq_rd_len_2	Read sequence with arlen 1

cover_seq_rd_len_3	Read sequence with arlen 2
cover_seq_rd_len_4	Read sequence with arlen 3
cover_seq_wr_len_1	Write sequence with awlen 0
cover_seq_wr_len_2	Write sequence with awlen 1
cover_seq_wr_len_3	Write sequence with awlen 2
cover_seq_wr_len_4	Write sequence with awlen 3
cover_seq_rd_rresp_out_of_order	Read response out of order possible
cover_seq_wr_bresp_out_of_order	Write response out of order possible
1.9 Write Interleave Covers	
cover_write_data_interleaving_two_ids_1	Write interleaving with two different IDs
cover_write_data_interleaving_two_ids_2	Write interleaving with two different IDs
2.0 Exclusive Access Covers	
cover_rresp_exokay	Exokay response for rresp possible
cover_bresp_exokay	Exokay response for bresp possible
cover_ex_arsize_0_bytecount_1	During exclusive read, byte count is 1
cover_ex_arsize_0_bytecount_2	During exclusive read, byte count is 2
cover_ex_arsize_0_bytecount_4	During exclusive read, byte count is 4
cover_ex_arsize_0_bytecount_8	During exclusive read, byte count is 8
cover_ex_arsize_0_bytecount_16	During exclusive read, byte count is 16
cover_ex_arsize_1_bytecount_2	During exclusive read, byte count is 2
cover_ex_arsize_1_bytecount_4	During exclusive read, byte count is 4

cover_ex_arsize_1_bytecount_8	During exclusive read, byte count is 8
cover_ex_arsize_1_bytecount_16	During exclusive read, byte count is 16
cover_ex_arsize_2_bytecount_4	During exclusive read, byte count is 4
cover_ex_arsize_2_bytecount_8	During exclusive read, byte count is 8
cover_ex_arsize_2_bytecount_16	During exclusive read, byte count is 16
cover_ex_arsize_2_bytecount_32	During exclusive read, byte count is 32
cover_ex_arsize_2_bytecount_64	During exclusive read, byte count is 64
cover_ex_arsize_3_bytecount_8	During exclusive read, byte count is 8
cover_ex_arsize_3_bytecount_16	During exclusive read, byte count is 16
cover_ex_arsize_3_bytecount_32	During exclusive read, byte count is 32
cover_ex_arsize_3_bytecount_64	During exclusive read, byte count is 64
cover_ex_arsize_3_bytecount_128	During exclusive read, byte count is 128
cover_ex_awsiz_0_bytecount_1	During exclusive write, byte count is 1
cover_ex_awsiz_0_bytecount_2	During exclusive write, byte count is 2
cover_ex_awsiz_0_bytecount_4	During exclusive write, byte count is 4
cover_ex_awsiz_0_bytecount_8	During exclusive write, byte count is 8
cover_ex_awsiz_0_bytecount_16	During exclusive write, byte count is 16
cover_ex_awsiz_1_bytecount_2	During exclusive write, byte count is 2
cover_ex_awsiz_1_bytecount_4	During exclusive write, byte count is 4

cover_ex_awsizel_bytcount_8	During exclusive write, byte count is 8
cover_ex_awsizel_bytcount_16	During exclusive write, byte count is 16
cover_ex_awsizel_bytcount_4	During exclusive write, byte count is 4
cover_ex_awsizel_bytcount_8	During exclusive write, byte count is 8
cover_ex_awsizel_bytcount_16	During exclusive write, byte count is 16
cover_ex_awsizel_bytcount_32	During exclusive write, byte count is 32
cover_ex_awsizel_bytcount_64	During exclusive write, byte count is 64
cover_ex_awsizel_bytcount_8	During exclusive write, byte count is 8
cover_ex_awsizel_bytcount_16	During exclusive write, byte count is 16
cover_ex_awsizel_bytcount_32	During exclusive write, byte count is 32
cover_ex_awsizel_bytcount_64	During exclusive write, byte count is 64
cover_ex_awsizel_bytcount_128	During exclusive write, byte count is 128
cover_ex_max_number_of_bytes_read	The maximum number of bytes that can be transferred in a read exclusive burst is 128.
cover_ex_max_number_of_bytes_write	The maximum number of bytes that can be transferred in a write exclusive burst is 128.
cover_ex_exread_start	Exclusive read possible
cover_ex_exwrite_start	Exclusive write possible
cover_ex_exread_exwrite_same_cycle	Exclusive read and write at the same cycle
cover_ex_exread_followed_by_exread	Exclusive read followed by another exclusive read
cover_ex_exread_not_exread_exread	Exclusive read sequence

cover_ex_exwrite_followed_by_exwrite	exwrite followed by another exclusive write
cover_ex_wr_is_noncacheable	Exclusive write burst is noncacheable
cover_ex_rd_is_noncacheable	Exclusive read burst is noncacheable
cover_ex_exwrite_after_exread	Exclusive write after exclusive read
cover_ex_arlock_normal	Normal access possible during exclusive read
cover_ex_awlock_normal	Normal access possible during exclusive write
cover_ex_tbl_full	Exclusive table full
cover_ex_tbl_full_to_empty	Exclusive table full to empty
2.1 Locked Access Covers	
cover_lock_arlock_valid	Read locked access possible
cover_lock_awlock_valid	Write locked access possible
cover_lock_awvalid_arvalid_together	Read and write locked access at the same time possible
cover_lock_seq1	Locked access sequence 1
cover_lock_seq2	Locked access sequence 2
cover_lock_seq3	Locked access sequence 3
cover_lock_seq4	Locked access sequence 4
cover_lock_seq5	Locked access sequence 5
cover_lock_seq6	Locked access sequence 6
cover_lock_seq7	Locked access sequence 7
cover_lock_seq8	Locked access sequence 8
cover_lock_read_lock_after_read_lock	Read lock after read lock
cover_lock_write_lock_after_write_lock	Write lock after write lock
cover_lock_write_lock_after_read_lock	Write lock after read lock
cover_lock_read_lock_after_write_lock	Read lock after write lock
cover_lock_write_dbc_write_normal	Data before control and normal write control

2.2 Byte Strobe Covers

cover_wstrb_constant_for_fixed_burst	Strobe remains constant during fixed burst
cover_wstrb_changes_for_incr_burst	Strobe changes during increment burst
cover_wstrb_changes_for_wrap_burst	Strobe changes during wrap burst

2.3 Internal Table Covers

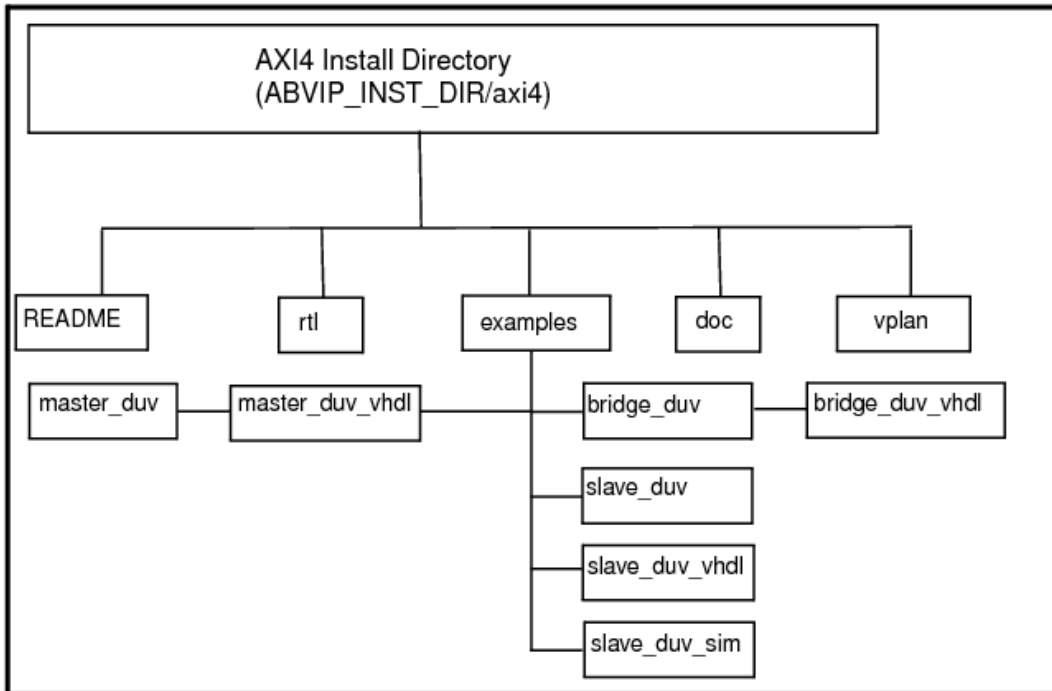
cover_rd_tbl_full	Read table full. This means that <code>MAX_PENDING_RD</code> number of read bursts are outstanding.
cover_rd_tbl_full_2_empty	Read table full to empty cover
cover_wr_tbl_full	Write table full cover. This means that <code>MAX_PENDING_WR</code> number of write bursts are outstanding (AW done and W not done or W done and AW not done or AW and W done and B not done)
cover_wr_tbl_full_2_empty	Write table full to empty cover
cover_wr_tbl_full_control	Write table control full. This means that <code>MAX_PENDING_WR</code> number of write bursts are outstanding with AW for all bursts done but W or B not done.
cover_wr_tbl_full_data	Write table data full. This means that <code>MAX_PENDING_WR</code> number of write bursts are outstanding with W for all bursts done but AW or B not done.
cover_rd_tbl_cnt	Internal read table count cover. For a read burst with <code>ARLEN=4'hf</code> and <code>ARID=1</code> , all the <code>RDATA</code> beats occur in 16 contiguous cycles.
cover_wr_tbl_cnt	Internal write table count cover. For a write burst with <code>AWLEN=4'hf</code> and <code>AWID=1</code> , all the <code>WDATA</code> beats occur in 16 contiguous cycles.

Appendix B: AXI4 ABVIP Package Contents

- [Package Contents](#)
- [AXI4 Monitor Pin-Level Interface](#)
- [AXI4 ABVIP Debug Signals](#)
- [AXI4 Monitor Parameters](#)
- [AXI4 Checks](#)
 - [Compliance Checks](#)
- [Coverage Checks](#)

This appendix describes the contents of the AXI4 ABVIP package. [Figure B-1](#) shows the package directory structure.

Figure B-1 Directory Structure of the AXI4 ABVIP Package



Package Contents

[Table B-1](#) shows the contents of the AXI4 installation directory.

Table B-1 Contents of the AXI4 installation directory

README	Mentions the contents of the package.
rtl	Contains the code for AXI4 ABVIP.

examples	Contains the examples of AXI4 ABVIP to be used in formal and simulation based verification.
doc	Contains the AXI ABVIP user guide and the migration guide.
vplan	Contains the ARM IHI 0022D specification based verification plan for AXI4 ABVIP. This can be used to view check status and coverage collected in the Enterprise Manager run.

Table B-2 shows the files stored in the rtl folder.

Table B-2 Contents of the rtl Folder

Version.v	Prints the version number of the AXI4 ABVIP.
cdn_abvip_axi4_monitor.svp	Encrypted SVA based monitor with protocol compliance checks and functional covers.
cdn_abvip_axi4_master.svp	Encrypted AXI4 master driving AXI4 master interface signals and checking AXI4 slave interface signals.
cdn_abvip_axi4_slave.svp	Encrypted AXI4 slave driving AXI4 slave interface signals and checking AXI4 master interface signals.
cdn_abvip_axi4.sv	Configurable abvip module which instantiates master/slave/monitor based on VERIFY_BLK parameter
axi4_monitor.sv	Configure cdn_abvip_axi4_monitor.svp with old abvip module name. This file is kept for backward compatibility and will be removed in a future release.
axi4_master.sv	Configure cdn_abvip_axi4_master.svp with old abvip module name. This file is kept for backward compatibility and will be removed in a future release.
axi4_slave.sv	Configure cdn_abvip_axi4_slave.svp with old abvip module name. This file is kept for backward compatibility and will be removed in a future release.
axi4_defines.svh	AXI4 definition file

<code>jasper_ipk_axi4_migration_wrapper.sv</code>	IPK to ABVIP migration wrapper file which consists of AXI4 IPK parameters and signal port mapping with corresponding ABVIP to ease the IPK migration.
---	---

The `example` folder contains examples for running AXI monitor in different configurations. Each subdirectory contains a README file that describes the steps for running the example and the content of that folder.

Table B-3 Contents of the Example Folder in AXI4 installation directory

README	Lists the contents of example directory.
CLEAN	Contains a command script that removes all intermediate log files.
demo.sh	Is an executable for running a simple ABVIP.
bridge_duv	Contains Verilog example for verifying AXI bridge using AXI4 ABVIP in formal.
bridge_duv_vhdl	Contains VHDL example for verifying AXI bridge using AXI4 ABVIP in formal.
master_duv	Contains Verilog example for verifying AXI master using AXI4 ABVIP in formal.
master_duv_vhdl	Contains VHDL example for verifying AXI master using AXI4 ABVIP in formal.
slave_duv	Contains Verilog example for verifying AXI slave using AXI4 ABVIP in formal.
slave_duv_vhdl	Contains VHDL example for verifying AXI slave using AXI4 ABVIP in formal.
slave_duv_sim	Contains Verilog example for verifying AXI bridge using AXI4 ABVIP in simulation.
axi4_wrapper	Contains Verilog example for verifying AXI master/slave using AXI4 ABVIP wrapper module.
examples_regression.vsimf	Input file to run all examples in Enterprise Manager.
rtl	Contains AXI master, slave, and bridge designs in Verilog and VHDL that are used in the examples.
xcelium	Contains Verilog example for verifying AXI4 ABVIP using Xcelium.

AXI4 Monitor Pin-Level Interface

The pin-level interface is a collection of HDL signals contained in an HDL module and connected to the DUV. A description of the pin-level interface signals for the AXI4 monitor is listed in [Table B-4](#).

Table B-4 The AXI4 Monitor Pin Interface

Pin	Source	Description
aclk	Clock source	System clock
aresetn	Reset controller	System reset--active low
awid[ID_WIDTH-1:0]	Master	Write address ID--AWID
awaddr[ADDR_WIDTH-1:0]	Master	Write address--AWADDR
awlen[LEN_WIDTH-1:0]	Master	Burst length--AWLEN
awsize[SIZE_WIDTH-1:0]	Master	Burst size--AWSIZE
awburst[BURST_WIDTH-1:0]	Master	Burst type--AWBURST
awlock	Master	Lock type--AWLOCK
awcache[CACHE_WIDTH-1:0]	Master	Cache type--AWCACHE
awprot[PROT_WIDTH-1:0]	Master	Protection type--AWPROT
awqos[QOS_WIDTH-1:0]	Master	Quality of service-AWQOS
awregion[REGION_WIDTH-1:0]	Master	Address region-AWREGION
awuser[AWUSER_WIDTH-1:0]	Master	User signal--AWUSER
awvalid	Master	Write address valid--AWVALID
awready	Slave	Write address ready--AWREADY
wdata[DATA_WIDTH-1:0]	Master	Write data--WDATA
wstrb[DATA_WIDTH/8-1:0]	Master	Write strobes--WSTRB
wuser[WUSER_WIDTH-1:0]	Master	User signal- wuser
wlast	Master	Write last--WLAST
wvalid	Master	Write valid--WVALID
wready	Slave	Write ready--WREADY

bid[ID_WIDTH-1:0]	Slave	Response ID--BID
bresp[BRESP_WIDTH-1:0]	Slave	Write response--BRESP
buser[BUSER_WIDTH-1:0]	Slave	User signal--buser
bvalid	Slave	Write response valid--BVALID
bready	Master	Response ready--BREADY
arid[ID_WIDTH-1:0]	Master	Read address ID--ARID
araddr[ADDR_WIDTH-1:0]	Master	Read address--ARADDR
arlen[LEN_WIDTH-1:0]	Master	Burst length--ARLEN
arsize[SIZE_WIDTH-1:0]	Master	Burst size--ARSIZE
arburst[BURST_WIDTH-1:0]	Master	Burst type--ARBURST
arlock	Master	Lock type--ARLOCK
arcache[CACHE_WIDTH-1:0]	Master	Cache type--ARCACHE
arprot[PROT_WIDTH-1:0]	Master	Protection type--ARPROT
arqos[QOS_WIDTH-1:0]	Master	Quality of service-ARQOS
aruser[ARUSER_WIDTH-1:0]	Master	User signal -ARUSER
arregion[REGION_WIDTH-1:0]	Master	Address region-ARREGION
arvalid	Master	Read address valid--ARVALID
arready	Slave	Read address ready-ARREADY
rid[ID_WIDTH-1:0]	Slave	Read ID tag--RID
rdata[DATA_WIDTH-1:0]	Slave	Read data--RDATA
rresp[RRESP_WIDTH-1:0]	Slave	Read response--RRESP
rlast	Slave	Read last--RLAST
ruser[USER_WIDTH-1:0]	Slave	User signal-RUSER
rvalid	Slave	Read valid--RVALID
rready	Slave	Read ready--RREADY

csysreq	Clock controller	System low-power request--CSYSREQ
csysack	Peripheral device	Low-power request acknowledgment--CSYSACK
cactive	Peripheral device	Clock active--CACTIVE

AXI4 ABVIP Debug Signals

AXI4 ABVIP supports the following debug or utility signals for transaction tracking. For an ongoing data beat or response, the corresponding control information will be valid for the following signals. In case of data or response before control, these signals are not valid and hold the value 0.


Signal Name	Description
waddr	awaddr for current write data beat. Valid only if control is previously received or ongoing.
wlen	awlen for current write data beat. Valid only if control is previously received or ongoing.
wsiz	awsiz for current write data beat. Valid only if control is previously received or ongoing.
wburst	awburst for current write data beat. Valid only if control is previously received or ongoing.
baddr	awaddr for current write response
blen	awlen for current write response
bsiz	awsiz for current write response
bburst	awburst for current write response
raddr	araddr for current read data beat
rlen	arlen for current read data beat
rsiz	arsiz for current read data beat
rburst	arburst for current read data beat



AXI4 Monitor Parameters


The AXI4 ABVIP is highly configurable. The monitor uses some parameters (Verilog `localparam`) for defining burst types, sizes, and access types. These parameters are internal to the AXI monitor and are not supposed to change. However, you can change the parameters listed in [Table B-5](#).

Table B-5 AXI4 Monitor Parameters


Parameter	Description
ADDR_WIDTH	Address bus width. Default value is 32.
DATA_WIDTH	Data bus width. Default value is 64.
ID_WIDTH	The number of bits needed to capture the number of IDs supported. Default value is 2.
LEN_WIDTH	The number of bits needed to capture the length of data per request. Default value is 4.
AXI4_LITE	If set to 1, the AXI4-Lite mode functionality is enabled. The default value is 0, which means AXI4.
ALL_STROBES_HIGH_ON	Whether all strobes are set to high. The default value is 0, which means this parameter is not set.
ALLOW_SPARSE_STROBE	When this parameter is set, master ABVIP can drive <code>wstrb</code> bits to 0 for one or more valid byte lanes. The default value of this parameter is 1. This parameter is effective only if parameter <code>BYTE_STROBE_ON</code> is set to 1.
BYTE_STROBE_ON	Whether write byte strobe (WSTRB) checking is required. The default value is 0, which means byte strobe checking is not required.
COVERAGE_ON	Whether coverage generation is set. The default value is 0, which is no coverage.

ERROR_CHECK_ON	<p>Set to 1 disable the ABVIP's max address boundary properties. The address will be allowed to cross the max boundary range.</p> <p>The default value is 0.</p>
CDNS_EXHAUSTIVE_COVER_ON	Enable exhaustive coverage on AXI4 channels
DATA_BEFORE_CONTROL_ON	<p>Whether data before control is supported.</p> <p>The default value is 1, which means data before control is supported.</p>
DATA_ACCEPT_WITH_OR_AFTER_CONTROL	<p>Whether slave can accept data with or after control.</p> <p>The default value is 0, which means data can be accepted before control.</p>
EXCL_ACCESS_ON	<p>Whether exclusive access is supported.</p> <p>The default value is 0, which means no exclusive access.</p>
READ_INTERLEAVE_ON	<p>Whether read data interleaving is supported.</p> <p>The default value is 1, which means read interleave mode is supported.</p>
READ_RESP_IN_ORDER_ON	<p>Whether read response in order is supported.</p> <p>The default value is 0, which means read response can come in any order.</p>
WRITE_RESP_IN_ORDER_ON	<p>Whether write response in order is supported.</p> <p>The default value is 0, which means write response can come in any order.</p>
RST_CHECKS_ON	<p>Whether reset checks are supported.</p> <p>The default value is 0, which means no reset checks.</p>
XCHECKS_ON	<p>Support of X-checks. The default value is 0, meaning no X-checks.</p> <div>  Use the Jasper elaborate switch “<code>enable_sva_isunknown</code>” for X-Prop analysis when this parameter is enabled. </div>
CONFIG_PARAM_CHECKS	<p>Set to 1 enables the ABVIP parameter checks.</p> <p>Default value is 1.</p>

CONFIG_LIVENESS	<p>Liveness properties will be created for the scenarios whose MAX_WAIT_CYCLES_* parameter is 0. Setting this parameter to 0 disables all liveness properties.</p> <p>Default value is 1.</p> <p>When enabled, all liveness assumptions and assertions are enabled.</p> <p>Also, if CONFIG_LIVENESS and DEADLOCK_CHKS_ON, both parameters are enabled then DEADLOCK_CHKS_ON will take priority. It will disable all liveness properties and will enable only deadlock checks.</p> <div data-bbox="625 688 1334 831">  Define macro JG_ABVIP_STRONG_SEMANTICS to enable strong semantics for liveness properties when using sv09 and above. </div>
DEADLOCK_CHKS_ON	<p>Liveness properties will be created for the scenarios whose MAX_WAIT_CYCLES_* parameter is 0. Setting this parameter to 1 enables all deadlock checks.</p> <p>Default value is 0.</p> <p>When enabled, then only liveness asserts are enabled and constraints are disabled.</p> <p>Also, if CONFIG_LIVENESS and DEADLOCK_CHKS_ON, both parameters are enabled then DEADLOCK_CHKS_ON will take priority. It will disable all liveness properties and will enable only deadlock checks.</p> <div data-bbox="625 1419 1334 1562">  Define macro JG_ABVIP_STRONG_SEMANTICS to enable strong semantics for deadlock checks when using sv09 and above. </div>
CONFIG_STABLE_CHECKS	<p>Set to 1 enables the ABVIP stable properties checks for signals.</p> <p>Default value is 1.</p>

CONFIG_RDATA_MASKED	<p>Set to 0 enables the X-propagation and stability properties applied to full Read data.</p> <p>Set to 1 enables the X-propagation and stability properties applied to valid byte lane of Read data.</p> <p>Default value is 1.</p>
CONFIG_RD_FULL_STRB	<p>Set to 0 enables full read strobe not required for the read transaction.</p> <p>Set to 1 enables full read strobe required for the read transaction.</p> <p>Default value is 0.</p>
CONFIG_CSR_INTERFACE	<p>This parameter is used to connect ABVIP with Jasper CSR App. The default value of this parameter is 0. You can set the parameter value to 1 to enable ABVIP with CSR checkers.</p>
CONFIG_CSR_FEED_EARLY_W	<p>Tune internal CSR logic behavior:</p> <p>Set to 0 feed write data to CSR checker on write response transaction</p> <p>Set to 1 feed write data to CSR checker on write data transaction</p> <p>Default value is 0.</p> <div>  Arrays of size $\text{MAX_PENDING_WR} \times \text{WLAXLEN} \times \text{DATA_WIDTH}$ will be created in ABVIP when this parameter is 0. -disable_auto_bbox or -no_bbox_m or -no_bbox_i or -bbox_a switch needs to be added in elaborate command to avoid black-boxing of these arrays in ABVIP. Elaboration time will increase in this mode. </div>

CONFIG_CSR_FEED_ON_BVALID	<p>Tune internal CSR logic behavior when <code>CONFIG_CSR_FEED_EARLY_W = 0</code>, Set to 0 feed write data to the CSR checker on write response channel "bvalid & bready" assertion. Set to 1 feed write data to the CSR checker on write response channel "bvalid" assertion. Default value is 0.</p>
CONFIG_CSR_FEED_ON_RVALID	<p>Tune internal CSR logic behavior, Set to 0 will feed read data to CSR checker on read response channel <code>rvalid</code> and <code>rready</code> assertion. Set to 1 will feed read data to CSR checker on read response channel <code>rvalid</code> assertion. Default value is 0.</p>
CONFIG_CSR_DISCARD_ON_ERROR	<p>Tune internal CSR logic behavior to handle the error response (SLVERR/DECERR) scenarios. Set to 0 sends the erroneous read/write data to CSR checker. Set to 1 tune Read/Write behavior according to the following description:</p> <ul style="list-style-type: none"> • For Read: Discard the read data beat with an error response to CSR checker • For Write: Discard the all write data beat of the burst with an error response to the CSR checker when <code>CONFIG_CSR_FEED_EARLY_W = 0</code>
LOW_POWER_ON	<p>Whether low power interface checking is required. The default value is 0, which means no low power interface checking. If low power mode is not supported, low power interface signals such as CSYSREQ, CACTIVE, and CSYSACK must be tied to 1'b1.</p>

MAX_WAIT_CYCLES_ON	<p>If set to 1, all MAX_WAIT_CYCLES_* parameters can be used. If set to 0, the MAX_WAIT_CYCLES_* parameters are assigned a value of infinity by using liveness properties. This value means, for instance, if an AWVALID goes high, AWREADY must eventually go high.</p> <div>  Define macro JG_ABVIP_STRONG_SEMANTICS to enable strong semantics for liveness properties when using sv09 and above. </div>
MAX_WAIT_CYCLES_AW	<p>The maximum number of cycles for which AWREADY is low after AWVALID goes high.</p> <p>Default value is 1.</p>
MAX_WAIT_CYCLES_W	<p>The maximum number of cycles for which WREADY is low after WVALID goes high.</p> <p>Default value is 1.</p>
MAX_WAIT_CYCLES_B	<p>The maximum number of cycles for which BREADY is low after BVALID goes high.</p> <p>Default value is 1.</p>
MAX_WAIT_CYCLES_AR	<p>The maximum number of cycles for which ARREADY is low after ARVALID goes high.</p> <p>Default value is 1.</p>
MAX_WAIT_CYCLES_R	<p>The maximum number of cycles for which RREADY is low after RVALID goes high.</p> <p>Default value is 1.</p>
MAX_WAIT_CYCLES_AW_W	<p>The maximum number of cycles for which write signals(WVALID, WREADY) are low after AWVALID goes high</p>
MAX_WAIT_CYCLES_W_AW	<p>The maximum number of cycles for which write address channel signals(AWVALID, AWREADY) are low after WVALID goes high</p>
MAX_WAIT_CYCLES_W_B	<p>The maximum number of cycles for which write response is low after wvalid goes high.</p>
MAX_WAIT_CYCLES_AW_B	<p>The maximum number of cycles for which write response is low after AWVALID goes high</p>

MAX_WAIT_CYCLES_AR_R	The maximum number of cycles for which read response is low after <code>ARVALID</code> goes high
READONLY_INTERFACE	Set to 1 enables Read only channels and disables Write channels
WRITEONLY_INTERFACE	Set to 1 enables Write only channels and disables Read channels
MAXLEN	<p>The maximum number of beats that can be transferred in a burst for read/write requests. Default value is 256. This constrains <code>AXLEN</code> value to be less than <code>MAXLEN</code>.</p> <p>When <code>CONFIG_CSR_INTERFACE</code> is set, the default value of this parameter is 16.</p>
MAXLEN_RD	<p>The maximum number of beats that can be transferred in a burst for read requests. Default value is 256. This constrains <code>ARLEN</code> value to be less than <code>MAXLEN_RD</code>.</p> <p>When <code>CONFIG_CSR_INTERFACE</code> is set, the default value of this parameter is 16.</p>
MAXLEN_WR	<p>The maximum number of beats that can be transferred in a burst for write requests. Default value is 256. This constrains <code>AWLEN</code> value to be less than <code>MAXLEN_WR</code>.</p> <p>When <code>CONFIG_CSR_INTERFACE</code> is set, the default value of this parameter is 16.</p>
RECM_CHKS_ALL_ON	Setting to 1 enables all recommended checks irrespective of the corresponding feature parameter. Default value is 0.
RECM_CHECKS_ON	Setting to 1 enables recommended checks only if the corresponding feature is set to 1. Default value is 0.
RECM_ASSUME_ON	The default value of this parameter is 1'b1. If the parameter is set to 0 then recommended checks are enabled and recommended constraints are disabled. When set to 1, it enables both recommended checks and constraints.
UPDATE_EXCL_TBL_ON_RD_RESP	<p>When this parameter is set to 1, the exclusive access trackers are updated based on the response of the exclusive read. This allows you to be less firm on the previous exclusive write response if an overwriting exclusive read is received.</p> <p>The default value of this parameter is 0.</p>

CDNS_ABVIP_STOP_OVERFLOW	<p>Disable ABVIP assumptions to prevent table overflow.</p> <p>When set to 1, master abvip will not generate *valid signals when MAX_PENDING transactions are outstanding. Also, when set to 1, slave abvip will not generate *ready signals when MAX_PENDING transactions are outstanding.</p> <p>Default value is 1, which means ABVIP overflow assumptions are present.</p>
CDNS_ABVIP_STOP_OVERFLOW_RD	<p>Disable ABVIP assumptions to prevent Read table overflow. Default value is the same as that of CDNS_ABVIP_STOP_OVERFLOW i.e. 1, which means ABVIP overflow assumptions are present.</p>
CDNS_ABVIP_STOP_OVERFLOW_WR	<p>Disable ABVIP assumptions to prevent Write table overflow. Default value is the same as that of CDNS_ABVIP_STOP_OVERFLOW i.e. 1, which means ABVIP overflow assumptions are present.</p>
CDNS_READY_OVFLOW_CHECKS	<p>When set to 1, master or monitor ABVIP will check that slave does not generate *ready signals when MAX_PENDING transactions are outstanding.</p> <p>Default value is 1.</p>
CDNS_VALID_OVFLOW_CHECKS	<p>When set to 1, slave or monitor ABVIP will check that master does not generate *valid signals when MAX_PENDING transactions are outstanding.</p> <p>Default value is 1.</p>
EXCL_ACCESS_WRITE_MATCH_EX_READ	<p>Set to 0 to enable the master to generate the illegal transactions in the exclusive access. By default, it will generate the exclusive write transaction matching with exclusive read</p>
EXCL_ACCESS_SLAVE_ON	<p>By default it is 1. Set to 0 to disable exclusive access of slave</p>
EXCL_ACCESS_WR_RESP_OKAY_ALLOWED	<p>Set to 1 to allow that slave response can send EXOKAY or OKAY response in case of valid exclusive access. By default, It is set to 0 means slave can send only exokay response</p>
CONFIG_CONSISTENT_DECERR	<p>when set to 1, it enables DECERR to be signaled for every beat of read data, or no beats of read data within each cache line of data. Default value is 0.</p>

AXI4 Checks

Compliance Checks

Table B-6 describes the SVA compliance checks for the master and slave in the AXI4 monitor. The properties have been grouped into sub-categories based on different aspects of the AXI4 protocol. The property naming conventions are also shown in the table. You can view the complete hierarchical property name in IFV by selecting the *View - Show Full Path Name* menu command.

Table B-6 AXI4 Monitor Compliance Checks

Property Name	Description	AXI Spec
1.1 Master Stable		
master_ar_arvalid_stable	The ARVALID signal must remain stable until ARREADY goes high.	ARM IHI 0022D, Section A3.2.1 on pg A3-37
master_ar_arid_stable	The ARID signal must remain stable when ARVALID is high until ARREADY goes high.	ARM IHI 0022D, Section A3.2.1 on pg A3-37
master_ar_araddr_stable	The ARADDR signal must remain stable when ARVALID is high until ARREADY goes high.	ARM IHI 0022D, Section A3.2.1 on pg A3-37
master_ar_arlen_stable	The ARLEN signal must remain stable when ARVALID is high until ARREADY goes high.	ARM IHI 0022D, Section A3.2.1 on pg A3-37

master_ar_arsize_stable	The ARSIZE signal must remain stable when ARVALID is high until ARREADY goes high.	ARM IHI 0022D, Section A3.2.1 on pg A3-37
master_ar_arburst_stable	The ARBURST signal must remain stable when ARVALID is high until ARREADY goes high.	ARM IHI 0022D, Section A3.2.1 on pg A3-37
master_ar_arlock_stable	The ARLOCK signal must remain stable when ARVALID is high until ARREADY goes high.	ARM IHI 0022D, Section A3.2.1 on pg A3-37
master_ar_arprot_stable	The ARPROT signal must remain stable when ARVALID is high until ARREADY goes high.	ARM IHI 0022D, Section A3.2.1 on pg A3-37
master_ar_arcache_stable	The ARCACHE signal must remain stable when ARVALID is high until ARREADY goes high.	ARM IHI 0022D, Section A3.2.1 on pg A3-37
master_ar_arqos_stable	The ARQOS signal must remain stable when ARVALID is high until ARREADY goes high.	ARM IHI 0022D, Section A3.2.1 on pg A3-37

master_ar_arregion_stable	The ARREGION signal must remain stable when ARVALID is high until ARREADY goes high.	ARM IHI 0022D, Section A3.2.1 on pg A3-37
master_ar_aruser_stable	The ARUSER signal must remain stable when ARVALID is high until ARREADY goes high.	ARM IHI 0022D, Section A3.2.1 on pg A3-37
master_aw_awvalid_stable	The AWVALID signal must remain stable until AWREADY goes high.	ARM IHI 0022D, Section A3.2.1 on pg A3-37
master_aw_awid_stable	The AWID signal must remain stable when AWVALID is high until AWREADY goes high.	ARM IHI 0022D, Section A3.2.1 on pg A3-37
master_aw_awaddr_stable	The AWADDR signal must remain stable when AWVALID is high until AWREADY goes high.	ARM IHI 0022D, Section A3.2.1 on pg A3-37
master_aw_awlen_stable	The AWLEN signal must remain stable when AWVALID is high until AWREADY goes high.	ARM IHI 0022D, Section A3.2.1 on pg A3-37

master_aw_awsized_stable	The AWSIZE signal must remain stable when AWVALID is high until AWREADY goes high.	ARM IHI 0022D, Section A3.2.1 on pg A3-37
master_aw_awburst_stable	The AWBURST signal must remain stable when AWVALID is high until AWREADY goes high.	ARM IHI 0022D, Section A3.2.1 on pg A3-37
master_aw_awlock_stable	The AWLOCK signal must remain stable when AWVALID is high until AWREADY goes high.	ARM IHI 0022D, Section A3.2.1 on pg A3-37
master_aw_awprot_stable	The AWPROT signal must remain stable when AWVALID is high until AWREADY goes high.	ARM IHI 0022D, Section A3.2.1 on pg A3-37
master_aw_awcache_stable	The AWCACHE signal must remain stable when AWVALID is high until AWREADY goes high.	ARM IHI 0022D, Section A3.2.1 on pg A3-37
master_aw_awqos_stable	The AWQOS signal must remain stable when AWVALID is high until AWREADY goes high.	ARM IHI 0022D, Section A3.2.1 on pg A3-37

master_aw_awregion_stable	The AWREGION signal must remain stable when AWVALID is high AWREADY goes high.	ARM IHI 0022D, Section A3.2.1 on pg A3-37
master_aw_awuser_stable	The AWUSER signal must remain stable when AWVALID is high AWREADY goes high.	ARM IHI 0022D, Section A3.2.1 on pg A3-37
master_w_wvalid_stable	The WVALID signal must remain stable until WREADY goes high.	ARM IHI 0022D, Section A3.2.1 on pg A3-37
master_w_wdata_stable	The WDATA signal must remain stable when WVALID is high WREADY goes high.	ARM IHI 0022D, Section A3.2.1 on pg A3-37
master_w_wstrb_stable	TheWSTRB signal must remain stable when WVALID is high WREADY goes high.	ARM IHI 0022D, Section A3.2.1 on pg A3-37
master_w_wid_stable	The WID signal must remain stable when WVALID is high WREADY goes high.	ARM IHI 0022D, Section A3.2.1 on pg A3-37

master_w_wlast_stable	The WLAST signal must remain stable when WVALID is high WREADY goes high.	ARM IHI 0022D, Section A3.2.1 on pg A3-37
1.2 Master Read		
master_ar_araddr_wrap_aligned	WRAP bursts should be aligned to the transfer size.	ARM IHI 0022D, Section A3.4.1 on pg A3-46
master_ar_arlen_less_than_TRANS_COUNT	ARLEN should be less than MAXLEN value configured	None
master_ar_araddr_wrap_arlen	WRAP bursts should be of length 2,4,8 or 16.	ARM IHI 0022D, Section A3.4.1 on pg A3-46
master_ar_araddr_fixed_arlen	FIXED bursts should be of length up to 16.	ARM IHI 0022D, Section A3.4.1 on pg A3-44
master_ar_araddr_never_cross_4K_boundary	Read burst must not cross 4K boundary.	ARM IHI 0022D, Section A3.4.1 on pg A3-44

master_ar_arburst_no_reserved	Read burst cannot take reserved value 2'b11.	ARM IHI 0022D, Section A3.4.1 on pg A3-46
master_ar_arsize_lte_datawidth	Size of any transfer must not exceed the data bus width of the components in the transaction.	ARM IHI 0022D, Section A3.4.1 on pg A3-45
master_ar_arcache_no_ra_wa_for_uncacheable	Read Allocate(RA) bit and Write Allocate(WA) bit of arcache, arcache[2] and arcache[3] must not be HIGH if modifiable bit arcache[1] is low.	ARM IHI 0022D, Section A4.2 on pg A4-59
master_ar_width_check	AWID, ARID should be of the same width for read and write channels for exclusive transactions	None
1.3 Master Write		
master_aw_awaddr_wrap_aligned	The start address must be aligned to the size of the transfer.	ARM IHI 0022D, Section A3.4.1 on pg A3-46
master_aw_awlen_less_than_TRANS_COUNT	AWLEN should be less than MAXLEN value configured	None
master_aw_awaddr_wrap_awlen	The length of the burst must be 2,4,8 or 16.	ARM IHI 0022D, Section A3.4.1 on pg A3-46

master_aw_awaddr_fixed_awlen	FIXED bursts should be of length up to 16.	ARM IHI 0022D, Section A3.4.1 on pg A3-44
master_aw_awaddr_never_cross_4K_boundary	Bursts must not cross 4KB boundaries to prevent them from crossing boundaries between slaves.	ARM IHI 0022D, Section A3.4.1 on pg A3-44
master_aw_awburst_no_reserved	Burst signal cannot take the reserved value 2'b11.	ARM IHI 0022D, Section A3.4.1 on pg A3-46
master_aw_awsize_lte_datawidth	The size of any transfer must not exceed the data bus width of the components in the transaction.	ARM IHI 0022D, Section A3.4.1 on pg A3-45
master_aw_awcache_no_ra_wa_non_modifiable	Write Cache RA and WA bits must de-asserted if the modifiable bit is not set.	ARM IHI 0022D, Section A4.2 on pg A4-59
master_w_aw_wlast_exact_len	wlast should be asserted in the last beat of a write transaction.	ARM IHI 0022D, Section A3.2.2 on pg A3.39

master_w_aw_wstrb_valid_collision	check strobe for the first beat of a transfer which comes in the same cycle as a control for the same transfer	ARM IHI 0022D, Section A3.4.1 on pg A3.46-A3.47
master_w_aw_wstrb_valid_curr	check strobe received in current cycle whose control is already received	ARM IHI 0022D, Section A3.4.1 on pg A3.46-A3.47
master_w_aw_wstrb_valid_dbc	In case where data beats are received before control, check the strobes of all the previously received beats when control is received.	ARM IHI 0022D, Section A3.4.1 on pg A3.46-A3.47
master_w_aw_wstrb_valid_curr_dbc	In case where data beats are received before control, check the strobes of the beat received in the same cycle as control.	ARM IHI 0022D, Section A3.4.1 on pg A3.46-A3.47
master_w_wstrb_all_active	Checks if the master is using full length data transfer.	None
master_aw_awsizel_all_bytelanes_active	When a master performs full data bus width write transactions, AWSIZE must be equal to WRITE_TRSIZE.	None
master_aw_width_check	AWID, ARID should be of same width for read and write channels for exclusive transactions	None
master_aw_w_awsizel_exact_len_collision	If data phase and control start simultaneously, the control phase must match that data transfer length.	None.

master_w_aw_wvalid_no_dbc	It ensures that wvalid should only come from master if either its control phase has already done or control phase is coming simultaneously with data phase in non-dbc case (i.e. when DATA_BEFORE_CONTROL_ON is off)	None.
master_w_aw_wlast_exact_len_collision	When data and control phase is coming simultaneously then wlast should only be asserted when awlen matches the data beat count.	None.
1.4 Master Atomic Access		
master_ar_excl_araddr_aligned	The address of exclusive access must be aligned to the total number of bytes in the transaction.	ARM IHI 0022D, Section A7.2.4 on pg A7.93
master_ar_excl_arlen_correct	Exclusive accesses are not permitted to use a burst length greater than 16.	ARM IHI 0022D, Section A7.2.4 on pg A7.93
master_ar_excl_arlen_arsize_correct_bytes	The number of bytes in exclusive burst should be 1,2,4,8,16,32,64 or 128.	ARM IHI 0022D, Section A7.2.4 on pg A7.93
master_ar_excl_arcache_low	Exclusive access burst cannot be cacheable and bufferable.	ARM IHI 0022D, Section A7.2.4 on pg A7.93

master_aw_excl_awcache_low	Exclusive access burst cannot be cacheable and bufferable.	ARM IHI 0022F: Section A7.2.4 on pg A7-99
master_ar_aw_excl_exwrite_after_exread	The master must not commence the write portion of exclusive access until the read portion is complete.	ARM IHI 0022D, Section A7.2.2 on pg A7.92
master_aw_excl_awlen_correct	Exclusive accesses are not permitted to use a burst length greater than 16.	ARM IHI 0022D, Section A7.2.4 on pg A7.93
master_aw_excl_awlen_awsizsize_correct_bytes	The number of bytes to be transferred in an exclusive access burst must be a power of 2, that is 1,2,4,8,16,32,64 or 128 bytes.	ARM IHI 0022D, Section A7.2.4 on pg A7.93
master_ar_excl_req_with_wr_ctl_done	The master should not send an exclusive read req when there is already excl write req occurred with the same ID	ARM IHI 0022F: Section A7.2.4 on pg A7.99
1.5 Master Write Data Before Control (DBC)		
master_aw_w_dbc_awlen_exact_len	If data phase finishes before control, the subsequent control phase must match that data transfer length.	None
master_w_aw_wlast_exact_len_dbc	wlast should be asserted in the last beat of a write transaction.	None

1.6 Master Reset

master_rst_arvalid_low_reset	Master should drive arvalid low during reset.	None
master_rst_arvalid_low_reset_released	Master should not drive arvalid at a posedge of aclk after aresetn is high.	None
master_rst_awvalid_low_reset	Master should drive awvalid low during reset.	None
master_rst_awvalid_low_reset_released	Master should not drive awvalid at a posedge of aclk after aresetn is high.	None
master_rst_wvalid_low_reset	Master should drive wvalid low during reset.	None
master_rst_wvalid_low_reset_released	Master should not drive wvalid at a posedge of aclk after aresetn is high.	None
master_rst_csysreq_high_reset_released	Master should drive csysreq high after aresetn is high	None

1.7 Master X-Checks

master_xcheck_arvalid	When reset is high, arvalid should not have X or Z value.	None
master_xcheck_arid	When arvalid is asserted, arid should not have X or Z value on any bit.	None
master_xcheck_araddr	When arvalid is asserted, arid should not have X or Z value on any bit.	None
master_xcheck_arlen	When arvalid is asserted, arlen should not have X or Z value on any bit.	None
master_xcheck_arsize	When arvalid is asserted, arsize should not have X or Z value on any bit.	None
master_xcheck_arburst	When arvalid is asserted, arburst should not have X or Z value on any bit.	None
master_xcheck_arlock	When arvalid is asserted, arlock should not have X or Z value on any bit.	None
master_xcheck_aprot	When arvalid is asserted, aprot should not have X or Z value on any bit.	None
master_xcheck_arcache	When arvalid is asserted, arcache should not have X or Z value on any bit.	None

AXI ABVIP User Guide
Appendix B: AXI4 ABVIP Package Contents--AXI4 Checks

master_xcheck_arqos	When arvalid is asserted, arqos should not have X or Z value on any bit.	None
master_xcheck_arregion	When arvalid is asserted, arregion should not have X or Z value on any bit.	None
master_xcheck_aruser	When arvalid is asserted, aruser should not have X or Z value on any bit.	None
master_xcheck_ready	When rvalid is asserted, rready should not have X or Z value.	None
master_xcheck_awvalid	When reset is asserted, awvalid should not have X or Z value.	None
master_xcheck_awid	When awvalid is asserted, awid should not have X or Z value on any bit.	None
master_xcheck_awaddr	When awvalid is asserted, awaddr should not have X or Z value on any bit.	None
master_xcheck_awlen	When awvalid is asserted, awlen should not have X or Z value on any bit.	None
master_xcheck_awsiz	When awvalid is asserted, awsize should not have X or Z value on any bit.	None
master_xcheck_awburst	When awvalid is asserted, awburst should not have X or Z value on any bit.	None
master_xcheck_awlock	When awvalid is asserted, awlock should not have X or Z value on any bit.	None
master_xcheck_awprot	When awvalid is asserted, awprot should not have X or Z value on any bit.	None
master_xcheck_awcache	When awvalid is asserted, awcache should not have X or Z value on any bit.	None
master_xcheck_awqos	When awqos is asserted, awcache should not have X or Z value on any bit.	None
master_xcheck_awregion	When awregion is asserted, awcache should not have X or Z value on any bit.	None
master_xcheck_awuser	When awuser is asserted, awcache should not have X or Z value on any bit.	None
master_xcheck_wvalid	When reset is asserted, wvalid should not have X or Z value.	None

master_xcheck_wdata	When wvalid is asserted, wdata should not have X or Z value on any bit.	None
master_xcheck_wstrb	When wvalid is asserted, wstrb should not have X or Z value on any bit.	None
master_xcheck_wid	When wvalid is asserted, wid should not have X or Z value on any bit.	None
master_xcheck_wlast	When wvalid is asserted, wlast should not have X or Z value.	None
master_xcheck_bready	When bvalid is asserted, bready should not have X or Z value.	None
1.8 Slave Stable		
slave_r_rvalid_stable	The RVALID signal must remain stable until RREADY goes high.	ARM IHI 0022D, Section A3.2.1 on pg A3-37
slave_r_rlast_stable	The RLAST signal must remain stable when RVALID is high until RREADY goes high.	ARM IHI 0022D, Section A3.2.1 on pg A3-37
slave_r_rdata_stable	The RDATA signal must remain stable when RVALID is high until RREADY goes high.	ARM IHI 0022D, Section A3.2.1 on pg A3-37

slave_r_rid_stable	The RID signal must remain stable when RVALID is high until RREADY goes high.	ARM IHI 0022D, Section A3.2.1 on pg A3-37
slave_r_rresp_stable	The RRESP signal must remain stable when RVALID is high until RREADY goes high.	ARM IHI 0022D, Section A3.2.1 on pg A3-37
slave_r_ruser_stable	The RUSER signal must remain stable when RVALID is high until RREADY goes high.	ARM IHI 0022D, Section A3.2.1 on pg A3-37
slave_b_bvalid_stable	The BVALID signal must remain stable until BREADY goes high.	ARM IHI 0022D, Section A3.2.1 on pg A3-37
slave_b_bid_stable	The BID signal must remain stable when BVALID is high until BREADY goes high.	ARM IHI 0022D, Section A3.2.1 on pg A3-37
slave_b_bresp_stable	The BRESP signal must remain stable when BVALID is high until BREADY goes high.	ARM IHI 0022D, Section A3.2.1 on pg A3-37

slave_b_buser_stable	The BUSER signal must remain stable when BVALID is high until BREADY goes high.	ARM IHI 0022D, Section A3.2.1 on pg A3-37
1.9 Slave Read		
slave_r_ar_rid_match	The Slaves are required to respond with an appropriate RID only after a valid request is received from master	ARM IHI 0022F: Section A5.1 on pg A5-80
slave_r_ar_rid_no_interleave	In a sequence of read transactions with different ARID values, the slave can return read data that is out-of-order with respect to the order in which read requests are received. In case read data interleaving is not supported, RID should not change before end of read data.	ARM IHI 0022D, Section A6.4 on pg A6-87
slave_r_ar_rid_in_order	The slave must return RRESP in the same order that the read request has been issued.	None
slave_r_ar_rlast_exact_len	The slave must assert the rlast signal when it drives the final read transfer in the burst.	ARM IHI 0022D, Section A3.2.2 on pg A3.39
slave_r_ar_resp_consistent_decoderr	DECERR is signaled for every beat of read data, or no beats of read data within each cache line of data	ARM IHI 0022H: Section E1.18.4 on pg E1.401
2.0 Slave Write		

slave_b_aw_bid_match	Slaves are required to respond with an appropriate BID only after a valid request is received from the master.	ARM IHI 0022F: Section A5.1 on pg A5.80
slave_b_aw_bid_order_within_id	All transactions with a given ID must be ordered.	ARM IHI 0022D, Section A6.4 on pg A6-87
slave_b_aw_bid_in_order	The slave must return BRESP in the same order that the write request is issued.	None
slave_w_aw_wready_dbc	If data before control is allowed for slave, it should issue <code>wready</code> with or after <code>awready</code> when <code>DATA_ACCEPT_WITH_OR_AFTER_CONTROL</code> is high	None
2.1 Slave Atomic Access		
slave_r_excl_rresp_no_exokay	The slave must not send an EXOKAY response for a normal read transaction.	ARM IHI 0022D, Section A7.2.2 on pg A7-92
slave_b_excl_bresp_no_exokay	The slave must not send an EXOKAY response till an exclusive write is successful.	ARM IHI 0022D, Section A7.2.2 on pg A7-92

slave_b_excl_bresp_no_exokay_supported	The slave must not send an EXOKAY response for a normal write transaction	ARM IHI 0022D: Section A7.2.3 on pg A7.93
slave_r_excl_rresp_no_exokay_supported	The slave must not send an EXOKAY response for a normal read transaction	ARM IHI 0022D: Section A7.2.2 on pg A7.92
slave_b_aw_excl_resp_overlapping_addr	Exclusive Write fails when same location is updated since the exclusive read has happened	ARM IHI 0022F: Section A7.2.4 on pg A7.99
slave_r_ar_excl_resp_exokay	The slave should not send an OKAY and EXOKAY exclusive read resp in different beat of same transaction	ARM IHI 0022F: Section A7.2.4 on pg A7.99
slave_r_ar_excl_resp_okay	The slave should not send an OKAY and EXOKAY exclusive read resp in different beat of same transaction	ARM IHI 0022F: Section A7.2.4 on pg A7.99
slave_r_excl_rresp_okay	The slave must not send an EXOKAY response when slave does not support exclusive access	ARM IHI 0022F: Section A7.2.4 on pg A7.99

2.2 Slave Reset

slave_rst_rvalid_low_reset	Slave should drive <code>rvalid</code> low during reset.	None
slave_rst_bvalid_low_reset	Slave should drive <code>bvalid</code> low during reset.	None
slave_rst_rvalid_low_reset_released	Slave should not drive <code>rvalid</code> at a posedge of <code>aclk</code> after <code>aresetn</code> is high.	None
slave_rst_bvalid_low_reset_released	Slave should not drive <code>bvalid</code> at a posedge of <code>aclk</code> after <code>aresetn</code> is high.	None
2.3 Slave X-Checks		
slave_xcheck_arready	When <code>arvalid</code> is asserted, <code>arready</code> should not have X or Z value.	None
slave_xcheck_rvalid	When reset is high, <code>rvalid</code> should not have X or Z value.	None
slave_xcheck_rdata	When <code>rvalid</code> is asserted, <code>rdata</code> should not have X or Z value on any bit.	None
slave_xcheck_rid	When <code>rvalid</code> is asserted, <code>rid</code> should not have X or Z value on any bit.	None
slave_xcheck_rresp	When <code>rvalid</code> is asserted, <code>rresp</code> should not have X or Z value on any bit.	None
slave_xcheck_ruser	When <code>rvalid</code> is asserted, <code>ruser</code> should not have X or Z value on any bit.	None
slave_xcheck_rlast	When <code>rvalid</code> is asserted, <code>rlast</code> should not have X or Z value on any bit.	None
slave_xcheck_awready	When <code>awvalid</code> is asserted, <code>awready</code> should not have X or Z value.	None
slave_xcheck_wready	When <code>awvalid</code> is asserted, <code>awready</code> should not have X or Z value.	None
slave_xcheck_bvalid	When reset is high, <code>bvalid</code> should not have X or Z value.	None
slave_xcheck_bid	When <code>bvalid</code> is asserted, <code>bid</code> should not have X or Z value on any bit.	None
slave_xcheck_bresp	When <code>bvalid</code> is asserted, <code>bresp</code> should not have X or Z value on any bit.	None

slave_xcheck_buser	When bvalid is asserted, buser should not have X or Z value on any bit.	None
2.4 Low Power Checks		
slave_lpi_entry_cactive_after_csysreq	cactive can be low only in low power state.	ARM IHI 0022E: Section A9.2.3 on pg A9-108
slave_lpi_csysack_high_when_cactive_low	csysack must not be low until atleast one cycle after cactive is low.	ARM IHI 0022E: Section A9.2.3 on pg A9-108
slave_lpi_csysack_low_when_cactive_high	csysack can be low for as many cycles as are required to complete exit sequence.	ARM IHI 0022E: Section A9.2.5 on pg A9-109
slave_lpi_csysack_low_after_csysreq_low	csysack is low when a request to enter low power state is recognized.	ARM IHI 0022E: Section A9.2.2 on pg A9-107
slave_lpi_csysack_high_after_csysreq_high	csysack is high when a request to exit low power state is recognized.	ARM IHI 0022E: Section A9.2.2 on pg A9-107

master_lpi_csysreq_low	csysreq remains low until handshake with csysack completes.	ARM IHI 0022E: Section A9.2.2 on pg A9-107
master_lpi_csysreq_high	csysreq remains high until handshake with csysack completes.	ARM IHI 0022E: Section A9.2.2 on pg A9-107
slave_lpi_csysack_follows_csysreq_low	csysreq must be acknowledged by csysack.	AXI Spec v1.0, Section 12.2.1, pg 12-4
slave_lpi_csysack_follows_csysreq_high	csysreq must be acknowledged by csysack.	AXI Spec v1.0, Section 12.2.1, pg 12-4
slave_lpi_cactive_high_exit	If csysreq is high then cactive must be high to indicate clk signal is required.	ARM IHI 0022E: Section A9.2.5 on pg A9-109
master_lpi_csysreq_low_to_high_eventually	The system must go from power down to power up mode eventually	None
assert_xcheck_csysreq	When reset is high, csysreq should not have X or Z value.	None
assert_xcheck_csysack	When reset is high, csysack should not have X or Z value.	None

assert_xcheck_cactive	When reset is high, cactive should not have X or Z value.	None
2.5 Internal Table Checks		
assert_ar_rd_tbl_no_overflow	This assertion checks that the number of outstanding read requests should not be greater than MAX_PENDING_RD	None
assert_aw_wr_tbl_no_overflow	This assertion checks that the number of write requests should not be greater than MAX_PENDING_WR	None
assert_w_wr_tbl_no_overflow	This assertion checks that number of outstanding write data transfers should not be greater than MAX_PENDING_WR	None
assert_ar_excl_tbl_no_overflow	This assertion checks that the number of outstanding exclusive requests should not be greater than MAX_PENDING_EXCL.	None
master_ar_rd_tbl_no_overflow	This assertion checks that master should not issue arvalid when number of outstanding read requests is MAX_PENDING_RD.	None
slave_ar_rd_tbl_no_overflow	This assertion checks that slave should not issue arready when number of outstanding read requests is MAX_PENDING_RD.	None
master_aw_wr_tbl_no_overflow	This assertion checks that master should not issue awvalid when number of outstanding write requests is MAX_PENDING_WR.	None
slave_aw_wr_tbl_no_overflow	This assertion checks that slave should not issue awready when number of outstanding write requests is MAX_PENDING_WR.	None
master_w_wr_tbl_no_overflow	This assertion checks that master should not issue wvalid when number of outstanding write requests is MAX_PENDING_WR.	None
slave_w_wr_tbl_no_overflow	This assertion checks that slave should not issue wready when number of outstanding write requests is MAX_PENDING_WR.	None
master_ar_arlock_excl_tbl_no_overflow	This assertion checks that master should not issue arlock when the number of outstanding exclusive requests is MAX_PENDING_EXCL.	None

slave_ar_arready_excl_tbl_no_overflow	This assertion checks that slave should not issue arready when the number of outstanding exclusive requests is MAX_PENDING_EXCL.	None
2.6 Deadlock Checks		
master_aw_awvalid_evntually	All pending write control requests must start eventually	None
master_aw_awvalid_wait_cycles	All pending write control requests must start within MAX_WAIT_CYCLES_W_AW cycles	None
slave_aw_awready_eventually	Slave must assert awready eventually if there are pending write requests	None
master_w_wvalid_eventually	All pending write data must start eventually	None
master_aw_wvalid_wait_cycles	All pending write data requests must start within MAX_WAIT_CYCLES_AW_W cycles	None
slave_w_wready_eventually	Slave must assert wready eventually if there are pending write data	None
slave_aw_wready_wait_cycles	All write data must be accepted within MAX_WAIT_CYCLES_AW_W cycles.	None
slave_aw_wready_wait_cycles	Slave must assert wready within MAX_WAIT_CYCLES_AW_W cycles if there are pending write data	None
slave_b_bvalid_eventually	All pending write response must start eventually	None
slave_aw_bvalid_wait_cycles	All pending responses must be given by slave within MAX_WAIT_CYCLES_AW_B cycles	None
master_b_bready_eventually	Master must assert bready if there are pending write response	None
master_b_bready_wait_cycles	Master must assert bready within MAX_WAIT_CYCLES_AW_B cycles if there are pending write response	None
slave_aw_awvalid_awready_eventually	If awvalid is asserted when awready is low, slave must assert awready eventually	None
slave_w_wvalid_wready_eventually	If wvalid is asserted when wready is low, slave must assert wready eventually	None
master_b_bvalid_bready_eventually	If bvalid is asserted when bready is low, master must assert bready eventually	None

slave_aw_awready_wait_cycles	The maximum number of cycles for which awready remains low after awvalid going high must not exceed the MAX_WAIT_CYCLES_AW specified	None
slave_w_wready_wait_cycles	The maximum number of cycles for which wready remains low after wvalid going high must not exceed the MAX_WAIT_CYCLES_W specified	None
master_b_bready_wait_cycles	The maximum number of cycles for which bready remains low after bvalid going high must not exceed the MAX_WAIT_CYCLES_B specified	None
slave_r_rvalid_eventually	All pending read responses must start eventually	None
slave_ar_rvalid_wait_cycles	All pending read responses must start within MAX_WAIT_CYCLES_AR_R cycles.	None
master_r_rready_eventually	Master must assert rready if there are pending read response	None
slave_ar_arvalid_arready_eventually	If arvalid is asserted when arready is low, slave must assert arready eventually	None
master_r_rvalid_rready_eventually	If rvalid is asserted when rready is low, master must assert rready eventually	None
master_ar_rready_wait_cycles	All read data must be accepted within MAX_WAIT_CYCLES_AR_R cycles.	None
slave_ar_arready_wait_cycles	The maximum number of cycles for which ARREADY remains low after ARVALID going high must not exceed the MAX_WAIT_CYCLES_AR cycles specified	None
master_r_rready_wait_cycles	The maximum number of cycles for which RREADY remains low after RVALID going high must not exceed the MAX_WAIT_CYCLES_R cycles specified	None

Coverage Checks

The monitor provides coverage checks to collect coverage information for possible scenarios in an AXI4-based design. The coverage information is required for complete verification of the DUV.

[Table B-7](#) describes the SVA coverage checks for the AXI4 monitor.

Table B-7 AXI Monitor Coverage Points

Coverage Assertions	Description
1.1 TRANSITION ID COVERS	
cover_arid_0	arid should take id 0
cover_arid_1	arid should take id 1
cover_arid_other	arid should take all other possible values
cover_awid_0	awid should take id 0
cover_awid_1	awid should take id 1
cover_awid_other	awid should take all other possible values
1.2 TRANSACTION LENGTH COVERS	
cover_arlen_len1	arlen, awlen can be upto 256
cover_arlen_len2	arlen, awlen can be upto 256
cover_arlen_len3	arlen, awlen can be upto 256
cover_arlen_len4	arlen, awlen can be upto 256
cover_arlen_short	arlen, awlen can be upto 256
cover_arlen_long	arlen, awlen can be upto 256
cover_arlen_max	arlen, awlen can be upto 256
cover_awlen_len1	arlen, awlen can be upto 256
cover_awlen_len2	arlen, awlen can be upto 256
cover_awlen_len3	arlen, awlen can be upto 256
cover_awlen_len4	arlen, awlen can be upto 256
cover_awlen_short	arlen, awlen can be upto 256
cover_awlen_long	arlen, awlen can be upto 256
cover_awlen_max	arlen, awlen can be upto 256
1.3 TRANSACTION BURST COVERS	

cover_arburst_fixed	arburst can be of fixed type
cover_arburst_incr	arburst can be of increment type
cover_arburst_wrap	arburst can be of wrap type
cover_awburst_fixed	awburst can be of fixed type
cover_awburst_incr	awburst can be of increment type
cover_awburst_wrap	awburst can be of wrap type
cover_arburst_wrap_len_2	If read is of wrap type, the arlen can be 1
cover_arburst_wrap_len_4	If read is of wrap type, the arlen can be 3
cover_arburst_wrap_len_8	If read is of wrap type, the arlen can be 7
cover_arburst_wrap_len_16	If read is of wrap type, the arlen can be 15
cover_awburst_wrap_len_2	If write is of wrap type, the awlen can be 1
cover_awburst_wrap_len_4	If write is of wrap type, the awlen can be 3
cover_awburst_wrap_len_8	If write is of wrap type, the awlen can be 7
cover_awburst_wrap_len_16	If write is of wrap type, the awlen can be 15
cover_arburst_fixed_araddr_unaligned	Unaligned read address during fixed burst
cover_arburst_incr_araddr_unaligned	Unaligned read address during incrementing burst
cover_awburst_fixed_awaddr_unaligned	Unaligned write address during fixed burst
cover_awburst_incr_awaddr_unaligned	Unaligned write address during incrementing burst

1.4 SLAVE RESPONSE SIGNALLING COVERS

cover_rresp_okay	rresp can have okay response
cover_rresp_slvterr	rresp can have slave error response
cover_rresp_decerr	rresp can have decode error response
cover_bresp_okay	bresp can have okay response
cover_bresp_slvterr	bresp can have slave error response
cover_bresp_decerr	bresp can have decode error response

1.5 SEARCHPOINT COVERS

cover_searchpoint_rd_control	Covers for read control
cover_searchpoint_rd_data	Covers for read data
cover_searchpoint_rd_data_last	Covers for read last data
cover_searchpoint_wr_control	Covers for write control
cover_searchpoint_wr_data	Covers for write data
cover_searchpoint_wr_data_last	Covers for write last data
cover_searchpoint_wr_response	Covers for write response

1.6 READ/WRITE ORDER COVER

cover_order_control_before_data	Control can come before data
cover_order_control_with_data	Control can come with data
cover_order_control_during_data	Control can come during data
cover_order_control_after_data	Control can come after data
cover_order_control_dbc_ongoing_no_wvalid	Data before control and no wvalid
cover_order_control_dbc_ongoing_no_wlast	Data before control and no wlast
cover_order_control_dbc_ongoing_wlast	Data before control and an ongoing wlast

cover_order_data_dbc_limit	Last transaction comes for DBC operation.
cover_order_data_dbc_no_limit_pending_ctl	Control information before last transaction of DBC operation.
cover_order_rready_before_rvalid	rready can come before rvalid
cover_order_rready_after_rvalid	rready can come after rvalid
cover_order_rready_with_rvalid	rready can come with rvalid
cover_order_wready_before_wvalid	wready can come before wvalid
cover_order_wready_after_wvalid	wready can come after wvalid
cover_order_wready_with_wvalid	wready can come with wvalid
cover_order_bready_before_bvalid	bready can come before bvalid
cover_order_bready_after_bvalid	bready can come after bvalid
cover_order_bready_with_bvalid	bready can come with bvalid
cover_order_wlast_before_wready	wlast can come before wready
cover_order_wlast_after_wready	wlast can come after wready
cover_order_wlast_with_wready	wlast can come with wready
cover_order_avalid_arvalid_together	avalid can come with arvalid
1.7 Max Wait Cycles Cover	
cover_max_wait_for_wr_request	Maximum number of cycles for which awready is low and awvalid is high
cover_max_wait_for_wr_data	Maximum number of cycles for which wready is low and wvalid is high
cover_max_wait_for_wr_response	Maximum number of cycles for which bready is low and bvalid is high
cover_max_wait_for_rd_request	Maximum number of cycles for which arready is low and arvalid is high

cover_max_wait_for_rd_response	Maximum number of cycles for which rready is low and rvalid is high
1.8 Read/Write Sequence Covers	
cover_seq_rd_len_1	Read sequence with arlen 0
cover_seq_rd_len_2	Read sequence with arlen 1
cover_seq_rd_len_3	Read sequence with arlen 2
cover_seq_rd_len_4	Read sequence with arlen 3
cover_seq_wr_len_1	Write sequence with awlen 0
cover_seq_wr_len_2	Write sequence with awlen 1
cover_seq_wr_len_3	Write sequence with awlen 2
cover_seq_wr_len_4	Write sequence with awlen 3
cover_seq_rd_rresp_out_of_order	Read response out of order possible
cover_seq_wr_bresp_out_of_order	Write response out of order possible
1.9 Exclusive Access Covers	
cover_rresp_exokay	Exokay response for rresp possible
cover_bresp_exokay	Exokay response for bresp possible
cover_ex_arsize_0_bytecount_1	During exclusive read, byte count is 1
cover_ex_arsize_0_bytecount_2	During exclusive read, byte count is 2
cover_ex_arsize_0_bytecount_4	During exclusive read, byte count is 4
cover_ex_arsize_0_bytecount_8	During exclusive read, byte count is 8
cover_ex_arsize_0_bytecount_16	During exclusive read, byte count is 16

cover_ex_arsize_1_bytecount_2	During exclusive read, byte count is 2
cover_ex_arsize_1_bytecount_4	During exclusive read, byte count is 4
cover_ex_arsize_1_bytecount_8	During exclusive read, byte count is 8
cover_ex_arsize_1_bytecount_16	During exclusive read, byte count is 16
cover_ex_arsize_2_bytecount_4	During exclusive read, byte count is 4
cover_ex_arsize_2_bytecount_8	During exclusive read, byte count is 8
cover_ex_arsize_2_bytecount_16	During exclusive read, byte count is 16
cover_ex_arsize_2_bytecount_32	During exclusive read, byte count is 32
cover_ex_arsize_2_bytecount_64	During exclusive read, byte count is 64
cover_ex_arsize_3_bytecount_8	During exclusive read, byte count is 8
cover_ex_arsize_3_bytecount_16	During exclusive read, byte count is 16
cover_ex_arsize_3_bytecount_32	During exclusive read, byte count is 32
cover_ex_arsize_3_bytecount_64	During exclusive read, byte count is 64
cover_ex_arsize_3_bytecount_128	During exclusive read, byte count is 128
cover_ex_awsiz_0_bytecount_1	During exclusive write, byte count is 1
cover_ex_awsiz_0_bytecount_2	During exclusive write, byte count is 2
cover_ex_awsiz_0_bytecount_4	During exclusive write, byte count is 4

cover_ex_awsiz_0_bytecount_8	During exclusive write, byte count is 8
cover_ex_awsiz_0_bytecount_16	During exclusive write, byte count is 16
cover_ex_awsiz_1_bytecount_2	During exclusive write, byte count is 2
cover_ex_awsiz_1_bytecount_4	During exclusive write, byte count is 4
cover_ex_awsiz_1_bytecount_8	During exclusive write, byte count is 8
cover_ex_awsiz_1_bytecount_16	During exclusive write, byte count is 16
cover_ex_awsiz_2_bytecount_4	During exclusive write, byte count is 4
cover_ex_awsiz_2_bytecount_8	During exclusive write, byte count is 8
cover_ex_awsiz_2_bytecount_16	During exclusive write, byte count is 16
cover_ex_awsiz_2_bytecount_32	During exclusive write, byte count is 32
cover_ex_awsiz_2_bytecount_64	During exclusive write, byte count is 64
cover_ex_awsiz_3_bytecount_8	During exclusive write, byte count is 8
cover_ex_awsiz_3_bytecount_16	During exclusive write, byte count is 16
cover_ex_awsiz_3_bytecount_32	During exclusive write, byte count is 32
cover_ex_awsiz_3_bytecount_64	During exclusive write, byte count is 64
cover_ex_awsiz_3_bytecount_128	During exclusive write, byte count is 128

cover_ex_max_number_of_bytes_read	The maximum number of bytes that can be transferred in a read exclusive burst is 128.
cover_ex_max_number_of_bytes_write	The maximum number of bytes that can be transferred in a write exclusive burst is 128.
cover_ex_exread_start	Exclusive read possible
cover_ex_exwrite_start	Exclusive write possible
cover_ex_exread_exwrite_same_cycle	Exclusive read and write at the same cycle
cover_ex_exread_followed_by_exread	Exclusive read followed by another exclusive read
cover_ex_exread_not_exread_exread	Exclusive read sequence
cover_ex_exwrite_followed_by_exwrite	exwrite followed by another exclusive write
cover_ex_wr_is_noncacheable	Exclusive write burst is non-cacheable
cover_ex_rd_is_noncacheable	Exclusive read burst is non-cacheable
cover_ex_exwrite_after_exread	Exclusive write after exclusive read
cover_ex_arlock_normal	Normal access possible during exclusive read
cover_ex_awlock_normal	Normal access possible during exclusive write
cover_ex_tbl_full	Exclusive table full
cover_ex_tbl_full_to_empty	Exclusive table full to empty
2.0 Internal Table Covers	
cover_rd_tbl_full	Read table full
cover_rd_tbl_full_2_empty	Read table full to empty cover
cover_wr_tbl_full	Write table data full cover
cover_wr_tbl_full_2_empty	Write table full to empty cover

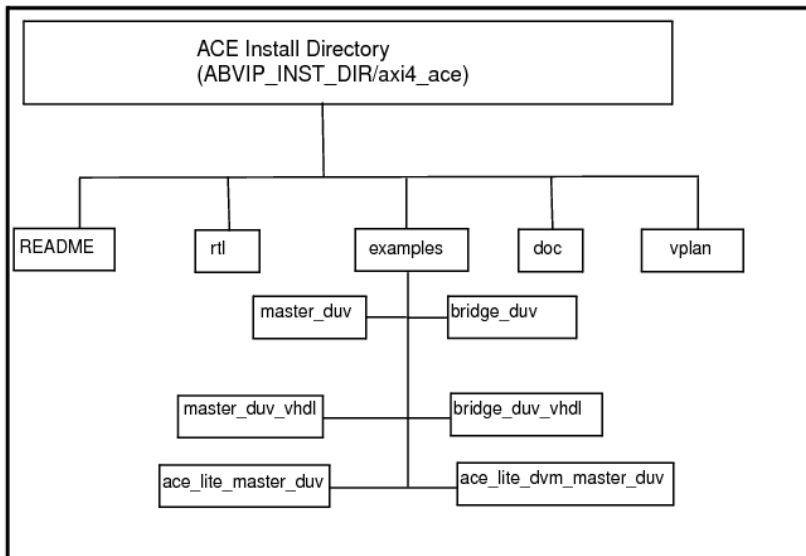
cover_wr_tbl_full_control	Write table control full
cover_wr_tbl_full_data	Write table full
cover_rd_tbl_cnt	Internal read table count cover

Appendix C: ACE ABVIP Package Contents

- [Package Contents](#)
- [ACE Monitor Pin-Level Interface](#)
- [ACE Monitor Parameters](#)
- [ACE Checks](#)
 - [Compliance Checks](#)
- [Coverage Checks](#)

This appendix describes the contents of the ACE ABVIP package. [Figure C-1](#) shows the package directory structure.

Figure C-1 Directory Structure of the ACE ABVIP Package



Package Contents

[Table C-1](#) shows the contents of the ACE installation directory.

Table C-1 Contents of the ACE installation directory

README	Mentions the contents of the package.
rtl	Contains the code for ACE ABVIP.
examples	Contains the examples of ACE ABVIP to be used in formal and simulation based verification.
doc	Contains the AXI ABVIP user guide.
vplan	Contains the ARM IHI 0022D specification based verification plan for ACE ABVIP. This can be used to view check status and coverage collected in the Enterprise Manager run.

[Table C-2](#) shows the files stored in the rtl folder.

Table C-2 Contents of the rtl Folder

Version.v	Prints the version number of the ACE ABVIP.
-----------	---

AXI ABVIP User Guide

Appendix C: ACE ABVIP Package Contents--Package Contents

cdn_abvip_axi4_ace_master.svp	ACE master driving ACE master interface signals and checking ACE slave interface signals.
cdn_abvip_axi4_ace_slave.svp	ACE slave driving ACE slave interface signals and checking ACE master interface signals.
cdn_abvip_axi4_ace_lite_master.svp	ACE-LITE master driving ACE-LITE master interface signals and checking ACE-LITE slave interface signals.
cdn_abvip_axi4_ace_lite_slave.svp	ACE-LITE slave driving ACE-LITE slave interface signals and checking ACE-LITE master interface signals.
cdn_abvip_axi4_ace_lite_dvm_master.svp	ACE-LITE (with DVM interface) master by driving ACE-LITE master interface signals and checking ACE-LITE slave interface signals.
cdn_abvip_axi4_ace_lite_dvm_slave.svp	ACE-LITE (with DVM interface) slave driving ACE-LITE slave interface signals and checking ACE-LITE master interface signals.
cdn_abvip_axi4_ace.sv	Configurable abvip module which instantiates master/slave/monitor based on <code>VERIFY_BLK</code> parameter
axi4_ace_master.sv	Configure <code>cdn_abvip_axi4_ace_master.svp</code> with an old abvip module name. This file is kept for backward compatibility and will be removed in a future release.
axi4_ace_slave.sv	Configure <code>cdn_abvip_axi4_ace_slave.svp</code> with an old abvip module name. This file is kept for backward compatibility and will be removed in a future release.
axi4_ace_lite_master.sv	Configure <code>cdn_abvip_axi4_ace_lite_master.svp</code> with an old abvip module name. This file is kept for backward compatibility and will be removed in a future release.
axi4_ace_lite_slave.sv	Configure <code>cdn_abvip_axi4_ace_lite_slave.svp</code> with an old abvip module name. This file is kept for backward compatibility and will be removed in a future release.
axi4_ace_lite_dvm_master.sv	Configure <code>cdn_abvip_axi4_ace_lite_dvm_master.svp</code> with an old abvip module name. This file is kept for backward compatibility and will be removed in a future release.
axi4_ace_lite_dvm_slave.sv	Configure <code>cdn_abvip_axi4_ace_lite_dvm_slave.svp</code> with an old abvip module name. This file is kept for backward compatibility and will be removed in a future release.
amba4_defines.svh	ACE definition file

The example folder contains examples for running the ACE monitor in different configurations. Each subdirectory contains a README file that describes the steps for running the example and the content of that folder.

Table C-3 Contents of the Example Folder in ACE installation directory

README	Lists the contents of example directory.
CLEAN	Contains a command script that removes all intermediate log files.
demo.sh	Executable for running a simple ABVIP.
bridge_duv	Contains Verilog example for verifying ACE bridge using ACE ABVIP in formal.

bridge_duv_vhdl	Contains VHDL example for verifying ACE bridge using ACE ABVIP in formal.
master_duv	Contains Verilog example for verifying ACE master using ACE ABVIP in formal.
master_duv_vhdl	Contains VHDL example for verifying ACE master using ACE ABVIP in formal.
ace_lite_master_duv	Contains a Verilog example for verifying ACE-LITE master using ACE-LITE ABVIP in formal.
ace_lite_dvm_master_duv	Contains a Verilog example for verifying ACE-LITE master with DVM interface using ACE-LITE ABVIP in formal.
ace_wrapper	Contains Verilog example for verifying AXI master/slave using AXI4 ACE ABVIP configurable module.
xcelium	Contains Verilog example for verifying ACE ABVIP using Xcelium.

ACE Monitor Pin-Level Interface

The pin-level interface is a collection of HDL signals contained in an HDL module and connected to the DUV. A description of the pin-level interface signals for the ACE monitor is listed in [Table C-4](#).

Table C-4 The ACE Monitor Pin Interface

Pin	Source	Description
acclk	Clock source	System clock
aresetn	Reset controller	System reset--active low
awid[ID_WIDTH-1:0]	Master	Write address ID--AWID
awaddr[ADDR_WIDTH-1:0]	Master	Write address--AWADDR
awlen[LEN_WIDTH-1:0]	Master	Burst length--AWLEN
awsize[SIZE_WIDTH-1:0]	Master	Burst size--AWSIZE
awburst[BURST_WIDTH-1:0]	Master	Burst type--AWBURST
awlock	Master	Lock type--AWLOCK
awcache[CACHE_WIDTH-1:0]	Master	Cache type--AWCACHE
awprot[PROT_WIDTH-1:0]	Master	Protection type--AWPROT
awqos[QOS_WIDTH-1:0]	Master	Quality of service--AWQOS
awregion[REGION_WIDTH-1:0]	Master	Address region--AWREGION
awuser[AWUSER_WIDTH-1:0]	Master	User signal--AWUSER
awdomain[DOMAIN_WIDTH-1:0]	Master	Domain signal--AWDOMAIN
awsnoop[AWSNOOP_WIDTH-1:0]	Master	Snoop signal--AWSNOOP
awbar[BAR_WIDTH-1:0]	Master	Bar signal--AWBAR
awvalid	Master	Write address valid--AWVALID

awready	Slave	Write address ready--AWREADY
wdata[DATA_WIDTH-1:0]	Master	Write data--WDATA
wstrb[DATA_WIDTH/8-1:0]	Master	Write strobes--WSTRB
wuser[WUSER_WIDTH-1:0]	Master	User signal- wuser
wlast	Master	Write last--WLAST
wack	Master	Write acknowledgement- WACK
wvalid	Master	Write valid--WVALID
wready	Slave	Write ready--WREADY
bid[ID_WIDTH-1:0]	Slave	Response ID--BID
bresp[BRESP_WIDTH-1:0]	Slave	Write response--BRESP
buser[BUSER_WIDTH-1:0]	Slave	User signal--buser
bvalid	Slave	Write response valid--BVALID
bready	Master	Response ready--BREADY
arid[ID_WIDTH-1:0]	Master	Read address ID--ARID
araddr[ADDR_WIDTH-1:0]	Master	Read address--ARADDR
arlen[LEN_WIDTH-1:0]	Master	Burst length--ARLEN
arsize[SIZE_WIDTH-1:0]	Master	Burst size--ARSIZE
arburst[BURST_WIDTH-1:0]	Master	Burst type--ARBURST
arlock	Master	Lock type--ARLOCK
arcache[CACHE_WIDTH-1:0]	Master	Cache type--ARCACHE
arprot[PROT_WIDTH-1:0]	Master	Protection type--ARPROT
arqos[QOS_WIDTH-1:0]	Master	Quality of service-ARQOS
aruser[ARUSER_WIDTH-1:0]	Master	User signal -ARUSER
arregion[REGION_WIDTH-1:0]	Master	Address region-ARREGION
ardomain[DOMAIN_WIDTH-1:0]	Master	Domain signal-ARDOMAIN
arsnoop[SNOOP_WIDTH-1:0]	Master	Snoop signal-ARSNOOP
arbar[BAR_WIDTH-1:0]	Master	Master Bar signal- ARBAR
arvalid	Master	Read address valid--ARVALID
aready	Slave	Read address ready-ARREADY
rid[ID_WIDTH-1:0]	Slave	Read ID tag--RID
rdata[DATA_WIDTH-1:0]	Slave	Read data--RDATA
rresp[RRESP_WIDTH-1:0]	Slave	Read response--RRESP
rlast	Slave	Read last--RLAST
ruser[USER_WIDTH-1:0]	Slave	User signal-RUSER

rack	Master	Read acknowledgement- RACK
rvalid	Slave	Read valid--RVALID
rready	Slave	Read ready--RREADY
acaddr[ADDR_WIDTH-1:0]	Interconnect	Snoop address- ACADDR
acprot[PROT_WIDTH-1:0]	Interconnect	Snoop protection signal- ACPROT
acsnoop[SNOOP_WIDTH-1:0]	Interconnect	Snoop signal- ACSNOOP
acvalid	Interconnect	Snoop valid- ACVALID
aready	Interconnect	Snoop ready- AREADY
crvalid	Master	Snoop response valid
cready	Master	Snoop response ready
cresp	Master	Snoop response
cdvalid	Master	Snoop data valid
cdready	Master	Snoop data ready
cddata	Master	Snoop data
cdlast	Master	Snoop data last
csysreq	Clock controller	System low-power request--CSYSREQ
csysack	Peripheral device	Low-power request acknowledgment--CSYSACK
cactive	Peripheral device	Clock active--CACTIVE

ACE Monitor Parameters

The ACE ABVIP is highly configurable. The monitor uses some parameters (Verilog `localparam`) for defining burst types, sizes, and access types. These parameters are internal to the ACE monitor and are not supposed to change. However, you can change the parameters listed in [Table C-5](#).

Table C-5 ACE Monitor Parameters


Parameter	Description
ADDR_WIDTH	Address bus width. Default value is 32. The minimum value of ADDR_WIDTH during byte strobe checking (BYTE_STROBE_ON is 1) must be 8.
DATA_WIDTH	Data bus width. Default value is 64.
ID_WIDTH	The number of bits needed to capture the number of IDs supported. Default value is 2.
LEN_WIDTH	The number of bits needed to capture the length of data per request. Default value is 4.

AXI ABVIP User Guide
Appendix C: ACE ABVIP Package Contents--ACE Monitor Parameters

ALL_STROBES_HIGH_ON	<p>Whether all strobes are set to high.</p> <p>The default value is 0, which means this parameter is not set.</p>
ALLOW_SPARSE_STROBE	<p>When this parameter is set, master ABVIP can drive <code>wstrb</code> bits to 0 for one or more valid byte lanes.</p> <p>The default value of this parameter is 1. This parameter is effective only if parameter <code>BYTE_STROBE_ON</code> is set to 1.</p>
BYTE_STROBE_ON	<p>Whether write byte strobe (WSTRB) checking is required.</p> <p>The default value is 0, which means byte strobe checking is not required.</p>
COVERAGE_ON	<p>Whether coverage generation is set.</p> <p>The default value is 0, which is no coverage.</p>
DATA_BEFORE_CONTROL_ON	<p>Whether data before control is supported.</p> <p>The default value is 1, which means data before control is supported.</p>
DATA_ACCEPT_WITH_OR_AFTER_CONTROL	<p>Whether slave can accept data with or after control.</p> <p>The default value is 0, which means data can be accepted before control.</p>
ERROR_CHECK_ON	<p>Set this parameter to 1 to disable the max address boundary properties of ABVIP. The address will be allowed to cross the max boundary range.</p> <p>The default value is 0.</p>
EXCL_ACCESS_ON	<p>Whether exclusive access is supported.</p> <p>The default value is 0, which means no exclusive access.</p>
BARRIER_ON	<p>Whether barrier transaction is supported.</p> <p>The default is 1, which means barrier transaction is supported.</p>
DVM_ON	<p>Whether DVM transaction is supported.</p> <p>The default is 0, which means no DVM transaction is allowed. This is not present in ACE-LITE.</p>
READ_INTERLEAVE_ON	<p>Whether read data interleaving is supported.</p> <p>The default value is 1, which means read interleave mode is supported.</p>
WRITE_INTERLEAVE_ON	<p>Whether write data interleaving is supported.</p> <p>The default value is 1, which means write interleave mode is supported.</p>
READ_RESP_IN_ORDER_ON	<p>Whether read response in order is supported.</p> <p>The default value is 0, which means read response can come in any order.</p>
WRITE_RESP_IN_ORDER_ON	<p>Whether write response in order is supported.</p> <p>The default value is 0, which means write response can come in any order.</p>

ADDR_HAZARD_CHECKS_ON	<p>Whether address hazard check is supported.</p> <p>The default is 0, which means no hazard checks.</p>
RST_CHECKS_ON	<p>Whether reset checks are supported.</p> <p>The default value is 0, which means no reset checks.</p>
XCHECKS_ON	<p>Support of X-checks. The default value is 0, meaning no X-checks.</p> <div style="border: 1px solid #f0e68c; padding: 10px; margin-top: 10px;"> <p>⚠ Use the Jasper elaborate switch “<code>enable_sva_isunknown</code>” for X-Prop analysis when this parameter is enabled.</p> </div>
CONFIG_PARAM_CHECKS	<p>Set to 1 will enable the ABVIP parameter checks.</p> <p>Default value is 1.</p>
CONFIG_LIVENESS	<p>Liveness properties will be created for the scenarios whose <code>MAX_WAIT_CYCLES_*</code> parameter is 0. Setting this parameter to 0 disables all liveness properties.</p> <p>Default value is 1.</p> <p>When enabled, all liveness assumptions and assertions are enabled.</p> <p>Also, if <code>CONFIG_LIVENESS</code> and <code>DEADLOCK_CHKS_ON</code>, both parameters are enabled then <code>DEADLOCK_CHKS_ON</code> will take priority. It will disable all liveness properties and will enable only deadlock checks.</p> <div style="border: 1px solid #f0e68c; padding: 10px; margin-top: 10px;"> <p>⚠ Define macro <code>JG_ABVIP_STRONG_SEMANTICS</code> to enable strong semantics for liveness properties when using sv09 and above.</p> </div>
DEADLOCK_CHKS_ON	<p>Liveness properties will be created for the scenarios whose <code>MAX_WAIT_CYCLES_*</code> parameter is 0. Setting this parameter to 1 enables all deadlock checks.</p> <p>Default value is 0.</p> <p>When enabled, then only liveness asserts are enabled and constraints are disabled.</p> <p>Also, if <code>CONFIG_LIVENESS</code> and <code>DEADLOCK_CHKS_ON</code>, both parameters are enabled then <code>DEADLOCK_CHKS_ON</code> will take priority. It will disable all liveness properties and will enable only deadlock checks.</p> <div style="border: 1px solid #f0e68c; padding: 10px; margin-top: 10px;"> <p>⚠ Define macro <code>JG_ABVIP_STRONG_SEMANTICS</code> to enable strong semantics for deadlock checks when using sv09 and above.</p> </div>

AXI ABVIP User Guide
Appendix C: ACE ABVIP Package Contents--ACE Monitor Parameters

CONFIG_STABLE_CHECKS	<p>Set to 1 will enable the ABVIP stable property checks for signals.</p> <p>Default value is 1.</p>
CONFIG_RDATA_MASKED	<p>Set to 0 will enable the X-propagation and stability properties applied to full Read data.</p> <p>Set to 1 will enable the X-propagation and stability properties applied to valid byte lane of Read data.</p> <p>Default value is 1.</p>
CONFIG_RD_FULL_STRB	<p>Set to 0 will enable full read strobe not required for the read transaction.</p> <p>Set to 1 will enable full read strobe required for the read transaction.</p> <p>Default value is 0.</p>
CONFIG_CSR_INTERFACE	<p>This parameter is used to connect ABVIP with Jasper CSR App. The default value of this parameter is 0. You can set the parameter value to 1 to enable ABVIP with CSR checkers.</p>
CONFIG_CSR_FEED_EARLY_W	<p>Tune internal CSR logic behavior:</p> <ul style="list-style-type: none"> • Set to 0 will feed write data to CSR checker on write response transaction • Set to 1 will feed write data to CSR checker on write data transaction <p>Default value is 0.</p> <div style="border: 1px solid #f9e79f; padding: 10px; margin-top: 10px;"> <p> Arrays of size <code>MAX_PENDING_WR*WLAXLEN*DATA_WIDTH</code> will be created in ABVIP when this parameter is 0.</p> <p>-disable_auto_bbox or -no_bbox_m or -no_bbox_i or -bbox_a switch needs to be added in elaborate command to avoid black-boxing of these arrays in ABVIP.</p> <p>Elaboration time will increase in this mode.</p> </div>
CONFIG_CSR_FEED_ON_BVALID	<p>Tune internal CSR logic behavior when <code>CONFIG_CSR_FEED_EARLY_W = 0</code>,</p> <ul style="list-style-type: none"> • Set to 0 will feed write data to CSR checker on write response channel "bvalid & bready" assertion • Set to 1 will feed write data to CSR checker on write response channel "bvalid" assertion. <p>Default value is 0.</p>

AXI ABVIP User Guide
Appendix C: ACE ABVIP Package Contents--ACE Monitor Parameters

CONFIG_CSR_FEED_ON_RVALID	<p>Tune internal CSR logic behavior,</p> <p>Set to 0 will feed read data to CSR checker on read response channel <code>rvalid</code> and <code>rready</code> assertion.</p> <p>Set to 1 will feed read data to CSR checker on read response channel <code>rvalid</code> assertion.</p> <p>Default value is 0.</p>
LOW_POWER_ON	<p>Whether low power interface checking is required.</p> <p>The default value is 0, which means no low power interface checking. If low power mode is not supported, low power interface signals such as <code>CSYSREQ</code>, <code>CACTIVE</code>, and <code>CSYSACK</code> must be tied to 1'b1.</p>
MAX_WAIT_CYCLES_ON	<p>If set to 1, all <code>MAX_WAIT_CYCLES_*</code> parameters can be used. If set to 0, the <code>MAX_WAIT_CYCLES_*</code> parameters are assigned a value of infinity by using liveness properties. This value means, for instance, if an <code>AWVALID</code> goes high, <code>AWREADY</code> must eventually go high.</p> <div style="border: 1px solid #f0e68c; padding: 10px; margin-top: 10px;"> <p>⚠ Define macro <code>JG_ABVIP_STRONG_SEMANTICS</code> to enable strong semantics for liveness properties when using sv09 and above.</p> </div>
MAX_WAIT_CYCLES_AW	The maximum number of cycles for which <code>AWREADY</code> is low after <code>AWVALID</code> goes high.
MAX_WAIT_CYCLES_W	The maximum number of cycles for which <code>WREADY</code> is low after <code>WVALID</code> goes high.
MAX_WAIT_CYCLES_B	The maximum number of cycles for which <code>BREADY</code> is low after <code>BVALID</code> goes high.
MAX_WAIT_CYCLES_AR	The maximum number of cycles for which <code>ARREADY</code> is low after <code>ARVALID</code> goes high.
MAX_WAIT_CYCLES_R	The maximum number of cycles for which <code>RREADY</code> is low after <code>RVALID</code> goes high.
MAX_WAIT_CYCLES_AW_W	The maximum number of cycles for which write signals(<code>WVALID</code> , <code>WREADY</code>) are low after <code>AWVALID</code> goes high
MAX_WAIT_CYCLES_W_AW	The maximum number of cycles for which write address channel signals(<code>AWVALID</code> , <code>AWREADY</code>) are low after <code>WVALID</code> goes high
MAX_WAIT_CYCLES_W_B	The maximum number of cycles for which write response is low after <code>wvalid</code> goes high.
MAX_WAIT_CYCLES_AW_B	The maximum number of cycles for which write response is low after <code>awvalid</code> goes high
MAX_WAIT_CYCLES_AR_R	The maximum number of cycles for which read response is low after <code>arvalid</code> goes high
MAX_PENDING_SNOOP	<p>The maximum number of pending snoop requests for which the response or data is incomplete. This is not present in ACE-LITE.</p> <p>This parameter should never be set to 0.</p>

AXI ABVIP User Guide
Appendix C: ACE ABVIP Package Contents--ACE Monitor Parameters

MAX_PENDING_BAR	<p>The maximum number of pending barrier requests for which the response or data is incomplete.</p> <p>This parameter should never be set to 0 if BARRIER_ON is set to 1.</p>
MAX_PENDING_DVM_SYNC	<p>The maximum number of pending DVM Sync requests allowed. This is not present in ACE-LITE.</p> <p>This parameter should never be set to 0.</p>
MAXLEN	<p>The maximum number of beats that can be transferred in a burst for read/write requests. Default value is 256. This will constrain <code>axlen</code> value to be less than <code>MAXLEN</code>.</p> <p>When <code>CONFIG_CSR_INTERFACE</code> is set, the default value of this parameter is 16.</p>
MAXLEN_RD	<p>The maximum number of beats that can be transferred in a burst for read requests. Default value is 256. This constrains <code>arlen</code> value to be less than <code>MAXLEN_RD</code>.</p> <p>When <code>CONFIG_CSR_INTERFACE</code> is set, the default value of this parameter is 16.</p>
MAXLEN_WR	<p>The maximum number of beats that can be transferred in a burst for write requests. Default value is 256. This constrains <code>awlen</code> value to be less than <code>MAXLEN_WR</code>.</p> <p>When <code>CONFIG_CSR_INTERFACE</code> is set, the default value of this parameter is 16.</p>
MAX_WAIT_CYCLES_AC	<p>The maximum number of cycles for which <code>ACREADY</code> is low after <code>ACVALID</code> goes high.</p>
MAX_WAIT_CYCLES_CR	<p>The maximum number of cycles for which <code>CRREADY</code> is low after <code>CRVALID</code> goes high.</p>
MAX_WAIT_CYCLES_CD	<p>The maximum number of cycles for which <code>CDREADY</code> is low after <code>CDVALID</code> goes high.</p>
MAX_WAIT_CYCLES_RACK	<p>The maximum number of cycles for which <code>RACK</code> is low after the read response.</p>
MAX_WAIT_CYCLES_WACK	<p>The maximum number of cycles for which <code>WACK</code> is low after the write response.</p>
MAX_WAIT_CYCLES_AC_CR	<p>Maximum wait cycles when <code>CRVALID</code> goes high after <code>ACVALID</code></p>
MAX_WAIT_CYCLES_CR_CD	<p>Maximum wait cycles when <code>CDVALID</code> goes high after <code>CRVALID</code> with <code>CRRESP[0]</code></p>
HINT_MULTIPART	<p>DVM multipart Hint message. The default value is 1, which means DVM hint multipart is supported.</p>
SHAREABLE_START_ADDR	<p>Shareable start address. Default value is 0.</p>
SHAREABLE_END_ADDR	<p>Shareable end address. The default value is $(10 * \text{CACHE_LINE_SIZE} - 1)$</p>
NONSHAREABLE_START_ADDR	<p>Non-Shareable start address. The default value is $(10 * \text{CACHE_LINE_SIZE})$</p>
NONSHAREABLE_END_ADDR	<p>Non-Shareable end address. The default value is $(10 * \text{CACHE_LINE_SIZE}) + 5120$</p>

AXI ABVIP User Guide
Appendix C: ACE ABVIP Package Contents--ACE Monitor Parameters

SYSTEM_START_ADDR	System start address. The default value is $(10 * \text{CACHE_LINE_SIZE}) + 5121$
CLK_CNTL_PRESENT	Clock controller present. The default value is 0, which means the clock controller is not present.
PROT_IN_OVERLAP	Prot value in overlapping address calculation. The default value is 1, which means prot value is used in address overlapping calculation.
CDNS_ABVIP_STOP_OVERFLOW	Disable ABVIP assumptions to prevent table overflow. When set to 1, master ABVIP will not generate new *valid signals when MAX_PENDING transactions are outstanding. Also, when set to 1, slave ABVIP will not generate *ready signals when MAX_PENDING transactions are outstanding. The default value is 1, which means ABVIP overflow assumptions are present.
CDNS_ABVIP_STOP_OVERFLOW_RD	Disable ABVIP assumptions to prevent Read table overflow. Default value is same as that of CDNS_ABVIP_STOP_OVERFLOW i.e. 1, which means ABVIP overflow assumptions are present.
CDNS_ABVIP_STOP_OVERFLOW_WR	Disable ABVIP assumptions to prevent Write table overflow. Default value is same as that of CDNS_ABVIP_STOP_OVERFLOW i.e. 1, which means ABVIP overflow assumptions are present.
CDNS_READY_OVERFLOW_CHECKS	When set to 1, master or monitor ABVIP will check that slave does not generate *ready signals when MAX_PENDING transactions are outstanding. Default value is 1.
CDNS_VALID_OVERFLOW_CHECKS	When set to 1, slave or monitor ABVIP will check that master does not generate *valid signals when MAX_PENDING transactions are outstanding. Default value is 1.
CDNS_ABVIP_STOP_OVERFLOW_SNP	Disable ABVIP assumptions to prevent Snoop table overflow. Default value is same as that of CDNS_ABVIP_STOP_OVERFLOW i.e. 1, which means ABVIP overflow assumptions are present.
CDNS_ABVIP_SOFT_CONS_ON	Disable assumptions required for assertion driven simulation (ADS) mode. The default value is 1, which means assumptions for ADS are active.
SNOOP_FILTER_PRESENT	Snoop filter present. The default value is 0, which means the snoop filter is not present.
CCI400_MULTI_CACHE_LINE	Parameter to apply CCI400 related assumptions.
WREVICT_ON	WriteEvict support. The default value is 0, which means WriteEvict is not supported.
RECM_CHKS_ALL_ON	Setting to 1 enables all recommended checks irrespective of the corresponding feature parameter. Default value is 0.
RECM_CHECKS_ON	Setting to 1 enables recommended checks only if the corresponding feature is set to 1. Default value is 0.

RECM_ASSUME_ON	The default value of this parameter is 1'b1. If the parameter is set to 0 then recommended checks are enabled and recommended constraints are disabled. When set to 1, it enables both recommended checks and constraints.
DVM_V8	Set to 1 to enable DVM properties complaint to DVMv8 message format.
CONFIG_SN_MI_NO_DAT	This parameter is used to enable the recommended behavior of no data transfer for <code>MakeInvalid</code> snoop.
UPDATE_EXCL_TBL_ON_RD_RESP	When this parameter is set to 1, the exclusive access trackers are updated based on the response of the exclusive read. This allows you to be less firm on the previous exclusive write response if an overwriting exclusive read is received. The default value of this parameter is 0.
CONFIG_ACCELERATOR_PORT	This enables support for AMBA4 Accelerator Coherency Port (ACP) for ACE4-Lite as defined in ARM IHI 0022E. The default value of this parameter is 0.
EXCL_ACCESS_WRITE_MATCH_EX_READ	Set to 0 to enable the master to generate the illegal transactions in the exclusive access. By default, it generates the exclusive write transaction matching with exclusive read.
EXCL_ACCESS_SLAVE_ON	Set to 0 to disable exclusive access of slave. The default value of this parameter is 1.
EXCL_ACCESS_WR_RESP_OKAY_ALLOWED	Set to 1 to allow that slave response can send EXOKAY or OKAY response in case of valid exclusive access. By default, It is set to 0 means slave can send only exokay response.
CONFIG_CONSISTENT_DECERR	Set to 1 to enable DECERR to be signaled for every beat of read data or no beats of read data within each cache line of data. The default value of this parameter is 0.
CONFIG_SHAREABLE_TRANSACTION	Set to 1 to define whether a master issues Inner Shareable or Outer Shareable transactions and whether a slave supports them. The default value of this parameter is 1.

ACE Checks

Compliance Checks

[Table C-6](#) describes the SVA compliance checks for the master and slave in the ACE monitor. The properties have been grouped into sub-categories based on different aspects of the ACE protocol. The property naming conventions are also shown in the table. You can view the complete hierarchical property name in IFV by selecting the *View - Show Full Path Name* menu command.

To view AXI4 monitor compliance checks, refer to [AXI4 Monitor Compliance Checks](#). To view additional ACE checks, refer to [Table C-6](#).

Table C-6 ACE Monitor Compliance Checks

Property Name	Description	AXI Spec	Applicable for ACE-LITE
---------------	-------------	----------	-------------------------

1.1 X-Checks

master_xcheck_ardomain	No AXI 4 signal should have X or Z value when the corresponding valid signal is asserted.	ARM IHI 0022D, Section A3.2.2 on pg A3-38	Yes
master_xcheck_arsnoop	No AXI 4 signal should have X or Z value when the corresponding valid signal is asserted.	ARM IHI 0022D, Section A3.2.2 on pg A3-38	Yes
master_xcheck_arbar	No AXI 4 signal should have X or Z value when the corresponding valid signal is asserted.	ARM IHI 0022D, Section A3.2.2 on pg A3-38	Yes
master_xcheck_rack	No AXI 4 signal should have X or Z value when the corresponding valid signal is asserted.	ARM IHI 0022D, Section A3.2.2 on pg A3-38	No
master_xcheck_awdomain	No AXI 4 signal should have X or Z value when the corresponding valid signal is asserted.	ARM IHI 0022D, Section A3.2.2 on pg A3-38	Yes
master_xcheck_awsnoop	No AXI 4 signal should have X or Z value when the corresponding valid signal is asserted.	ARM IHI 0022D, Section A3.2.2 on pg A3-38	Yes
master_xcheck_awbar	No AXI 4 signal should have X or Z value when the corresponding valid signal is asserted.	ARM IHI 0022D, Section A3.2.2 on pg A3-38	Yes

AXI ABVIP User Guide
Appendix C: ACE ABVIP Package Contents--ACE Checks

master_xcheck_wack	No AXI 4 signal should have X or Z value when the corresponding valid signal is asserted.	ARM IHI 0022D, Section A3.2.2 on pg A3-38	No
master_xcheck_acready	No AXI 4 signal should have X or Z value when the corresponding valid signal is asserted.	ARM IHI 0022D, Section A3.2.2 on pg A3-38	Yes
master_xcheck_crvalid	No AXI 4 signal should have X or Z value when the corresponding valid signal is asserted.	ARM IHI 0022D, Section A3.2.2 on pg A3-38	Yes
master_xcheck_cresp	No AXI 4 signal should have X or Z value when the corresponding valid signal is asserted.	ARM IHI 0022D, Section A3.2.2 on pg A3-38	Yes
master_xcheck_cdvalid	No AXI 4 signal should have X or Z value when the corresponding valid signal is asserted.	ARM IHI 0022D, Section A3.2.2 on pg A3-38	No
master_xcheck_cddata	No AXI 4 signal should have X or Z value when the corresponding valid signal is asserted.	ARM IHI 0022D, Section A3.2.2 on pg A3-38	No
master_xcheck_cdlast	No AXI 4 signal should have X or Z value when the corresponding valid signal is asserted.	ARM IHI 0022D, Section A3.2.2 on pg A3-38	No
slave_xcheck_acvalid	No AXI 4 signal should have X or Z value when the corresponding valid signal is asserted.	ARM IHI 0022D, Section A3.2.2 on pg A3-38	Yes

AXI ABVIP User Guide
Appendix C: ACE ABVIP Package Contents--ACE Checks

slave_xcheck_acaddr	No AXI 4 signal should have X or Z value when the corresponding valid signal is asserted.	ARM IHI 0022D, Section A3.2.2 on pg A3-38	Yes
slave_xcheck_acsnoop	No AXI 4 signal should have X or Z value when the corresponding valid signal is asserted.	ARM IHI 0022D, Section A3.2.2 on pg A3-38	Yes
slave_xcheck_acprot	No AXI 4 signal should have X or Z value when the corresponding valid signal is asserted.	ARM IHI 0022D, Section A3.2.2 on pg A3-38	Yes
slave_xcheck_cready	No AXI 4 signal should have X or Z value when the corresponding valid signal is asserted.	ARM IHI 0022D, Section A3.2.2 on pg A3-38	Yes
slave_xcheck_cdready	No AXI 4 signal should have X or Z value when the corresponding valid signal is asserted.	ARM IHI 0022D, Section A3.2.2 on pg A3-38	No

1.2 Stability Checks

master_ar_ardomain_stable	ardomain must remain stable while arready is low.	ARM IHI 0022D, Section A3.2.1 on pg A3-37	Yes
master_ar_arsnoop_stable	arsnoop must remain stable while arready is low.	ARM IHI 0022D, Section A3.2.1 on pg A3-37	Yes

AXI ABVIP User Guide
Appendix C: ACE ABVIP Package Contents--ACE Checks

master_ar_arbar_stable	arbar must remain stable while arready is low.	ARM IHI 0022D, Section A3.2.1 on pg A3-37	Yes
master_aw_awdomain_stable	awdomain must remain stable while awready is low.	ARM IHI 0022D, Section A3.2.1 on pg A3-37	Yes
master_aw_awsnoop_stable	awsnoop must remain stable while awready is low.	ARM IHI 0022D, Section A3.2.1 on pg A3-37	Yes
master_aw_awbar_stable	awbar must remain stable while awready is low.	ARM IHI 0022D, Section A3.2.1 on pg A3-37	Yes
slave_ac_acvalid_stable	acvalid must remain stable while acready is low.	ARM IHI 0022D, Section C3.6.2 on pg C3-174	Yes
slave_ac_acaddr_stable	acaddr must remain stable while acvalid is high and acready low.	ARM IHI 0022D, Section C3.6.2 on pg C3-174	Yes
slave_ac_acsnoop_stable	acsnoop must remain stable while acvalid is high and acready low.	ARM IHI 0022D, Section C3.6.2 on pg C3-174	Yes
slave_ac_acprot_stable	acprot must remain stable while acvalid is high and acready low.	ARM IHI 0022D, Section C3.6.2 on pg C3-174	Yes

AXI ABVIP User Guide
Appendix C: ACE ABVIP Package Contents--ACE Checks

master_cr_crvalid_stable	crvalid must remain stable while cready is low.	ARM IHI 0022D, Section C3.7 on pg C3-176	Yes
master_cr_crresp_stable	crresp must remain stable while crvalid is high and cready low.	ARM IHI 0022D, Section C3.7 on pg C3-176	Yes
master_cd_cdvalid_stable	cdvalid must remain stable while cready is low.	ARM IHI 0022D, Section C3.8 on pg C3-180	No
master_cd_cddata_stable	cddata must remain stable while cdvalid is high and cready low.	ARM IHI 0022D, Section C3.8 on pg C3-180	No
master_cd_cdlast_stable	cdlast must remain stable while cdvalid is high and cready low.	ARM IHI 0022D, Section C3.8 on pg C3-180	No
1.3 Reset Checks			
master_rst_crvalid_low_reset	Master should drive crvalid low during reset.	ARM IHI 0022D, Section A3.1.2 on pg A3-36	Yes
master_rst_crvalid_low_reset_released	Master should not drive crvalid at a posedge of aclk after aresetn is high.	ARM IHI 0022D, Section A3.1.2 on pg A3-36	Yes

AXI ABVIP User Guide
Appendix C: ACE ABVIP Package Contents--ACE Checks

master_rst_cdvalid_low_reset	Master should drive cdvalid low during reset.	ARM IHI 0022D, Section A3.1.2 on pg A3-36	No
master_rst_cdvalid_low_reset_released	Master should not drive cdvalid at a posedge of aclk after aresetn is high.	ARM IHI 0022D, Section A3.1.2 on pg A3-36	No
slave_rst_acvalid_low_reset	Slave should drive acvalid low during reset.	ARM IHI 0022D, Section A3.1.2 on pg A3-36	Yes
slave_rst_acvalid_low_reset_released	Slave should not drive acvalid at a posedge of aclk after aresetn is high	ARM IHI 0022D, Section A3.1.2 on pg A3-36	Yes
1.4 Read Checks			
master_ar_arsize_for_coherent	arsize must be equal to data bus width for Cache Line Size, Barrier and DVM Transactions, arsize must be equal to or less than data bus width for ReadOnce Transactions	ARM IHI 0022D, Section C3.1.5 on pg C3-163, C3-166, section C12.6 on C12-287	Yes
master_ar_arlen_less_than_TRANS_COUNT	ARLEN should be less than the MAXLEN value configured	None	Yes
master_ar_arsize_arlen	Transaction size must be equal to cache line size for Cache Line Size Transactions	ARM IHI 0022D, Section C3.1.5 on pg C3-163	Yes

AXI ABVIP User Guide
Appendix C: ACE ABVIP Package Contents--ACE Checks

master_ar_arlen_for_coherent	arlen must be less than 16 for Cache Line Size Transactions arlen must be all zeros for Barrier/DVM Transactions	ARM IHI 0022D, Section C3.1.5 on pg C3-163, C3-166, section C12.6 on C12-287	Yes
master_ar_arburst_for_coherent	arburst must be INCR for all Barrier/DVM Transactions arburst must be INCR or WRAP for all other coherent Transactions	ARM IHI 0022D, Section C3.1.5 on pg C3-163, C3-166, section C12.6 on C12-287	Yes
master_ar_araddr_for_barrier	araddr must be all zeros for Barrier	ARM IHI 0022D, Section C3.1.5 on pg C3-166	Yes
master_ar_araddr_for_cachelinesize	When ARBURST is INCR, the address must be aligned to the cache line size	ARM IHI 0022D, Section C3.1.5 on pg C3-163	Yes
master_ar_arcache_for_coherent	arcache must be normal non-cacheable (AxCACHE=0010) for Barrier/DVM Transactions. arcache must be modifiable (AxCACHE[1]=1) for all other coherent Transactions	ARM IHI 0022D, Section C3.1.5 on pg C3-163 - C3-166, section C12.6 on C12-287	Yes

AXI ABVIP User Guide
Appendix C: ACE ABVIP Package Contents--ACE Checks

master_ar_arlock_for_coherent	arlock must be normal access (AxLOCK=0) for all coherent Read Transactions except ReadClean, ReadShared, CleanUnique.	ARM IHI 0022D, Section C3.1.5 on pg C3-163 - C3-166, section C12.6 on C12-287	Yes
master_ar_ardomain_by_arcache	Device transactions, as indicated by AxCACHE[1] = 0, must only use AxDOMAIN = 11, System. Cacheable transactions, as indicated by AxCACHE[3:2] != 00, must not use AxDOMAIN = 11, System.	ARM IHI 0022D, Section C3.1.1 on C3-158	Yes
master_ar_ardomain_for_coherent	ardomain must be INNER, OUTER for coherent and dvm transaction groups. ardomain must be INNER, OUTER, NON for cache maintenance transaction group	ARM IHI 0022D, Section C3.1.3 on pg C3-161, section C3.1.5 on pg C3-163, C3-164, C12.6 on pg C12-287	Yes
master_ar_arbar_for_coherent	arbar must be normal access for all Transactions except Barrier	ARM IHI 0022D, Section C3.1.5 on pg C3-163 - C3-166	Yes
master_ar_arsnoop_legal_values	Legal Read address channel combinations of arsnoop, arbar, ardomain. All unused encodings are reserved.	ARM IHI 0022D, Section C3.1.3 on pg C3-161	Yes

AXI ABVIP User Guide
Appendix C: ACE ABVIP Package Contents--ACE Checks

master_ar_arsnoop_for_barrier	All barrier transactions, must have ARSNOOP[3:0] = 0000 arsnoop must be all zeroes for Barrier Transactions	ARM IHI 0022D, Section C3.1.3 on pg C3-161, section C3.1.5 on pg C3-166	Yes
master_ar_arbar_for_dvm	All DVM transactions, arbar should have the value of 2'b00	ARM IHI 0022D, Section C12.6 on pg C12-287	No
master_ar_aw_no_overlapping_addr	The current read request address should not overlap with any of the outstanding write requests and current write request(if any).	None	Yes
master_ar_r_arcache_no_read_while_maintenance_scl	The master must not issue any further shareable transactions to the same cache line until the cache maintenance transaction is complete.	ARM IHI 0022D, Section C7.2.1 on pg C7-243	Yes
slave_r_ar_resp	ReadNoSnoop, ReadUnique, CleanUnique, MakeUnique, CleanInvalid, and MakeInvalid transaction must have a response with IsShared RRESP[3] de-asserted (IsUnique). ReadNoSnoop, ReadOnce, ReadClean, CleanUnique, MakeUnique, CleanShared, CleanInvalid, and MakeInvalid transactions must have a response with PassDirty RRESP[2] de-asserted (PassClean). A barrier transaction and a DVM transaction response must use RRESP[3:2] = "00". A ReadNotSharedDirty transaction response must not use RRESP[3:2] = "11".	ARM IHI 0022D, Section C3.2.1 on pg C3-167, C3-168	No
master_r_rack_eventually	All pending read transactions will be acknowledged eventually.	None	No
master_r_rack_wait_cycles	All pending read transactions will be acknowledged within MAX_WAIT_CYCLES_RACK cycles	None	No
master_ar_width_check	AWID, ARID should be of the same width for read and write channels for exclusive transactions	None	Yes
master_ar_shareable_tr_ardomain	The must must not send Inner Shareable and Outer Shareable transactions	ARM IHI 0022H: Section E1.18.2 on pg E1-399	Yes

AXI ABVIP User Guide
Appendix C: ACE ABVIP Package Contents--ACE Checks

Slave_r_ar_resp_consistent_decoderr	DECERR is signaled for every beat of read data, or no beats of read data within each cache line of data	ARM IHI 0022H: Section E1.18.4 on pg E1.401	Yes
1.5 Write Control Checks			
master_aw_no_evict	An Evict transaction is only used by a master that supports a snoop filter.	ARM IHI 0022D, Section C4.9.1 on pg C4-213	No
master_aw_awunique_for_writeclean	AWUNIQUE must be deasserted for WriteClean transaction	ARM IHI 0022D, Section C3.1.4 on pg C3-167	No
master_aw_awunique_for_writeevict	AWUNIQUE must be asserted for WriteEvict transaction	ARM IHI 0022D, Section C3.1.4 on pg C3-167	No
master_aw_no_writeevict	No WriteEvict transaction, if WREVICT_ON parameter is off.	None	No
master_ar_r_aw_b_awsnoop_no_maintenance_while_pending_scl	The master must complete any outstanding shareable transactions to the same cache line before issuing the cache maintenance transaction.	ARM IHI 0022D, Section C7.2.1 on pg C7-243	No
master_aw_awsiz_for_coherent	awsiz must be equal to data bus width for Cache Line Size and Barrier Transactions awsiz must be equal to or less than data bus width for WriteBack/Clean Transactions	ARM IHI 0022D, Section C3.1.5 on pg C3-163 - C3-166	Yes

AXI ABVIP User Guide
Appendix C: ACE ABVIP Package Contents--ACE Checks

master_aw_awsiz_awlen	Transaction size must be equal to cache line size for Cache Line Size Transactions transaction size must be equal to or less than cache line size for WriteBack/Clean Transactions	ARM IHI 0022D, Section C3.1.5 on pg C3-163 - C3-165	Yes
master_aw_awlen_less_than_TRANS_COUNT	AWLEN should be less than the MAXLEN value configure	None	Yes
master_aw_awlen_for_coherent	awlen must be 1, 2, 4, 8, 16 for Cache Line Size Transactions, awlen must be equal to or less than 16 for WriteBack/Clean Transactions, awlen must be all zeros for Barrier/DVM Transactions	ARM IHI 0022D, Section C3.1.5 on pg C3-163, C3-166	Yes
master_aw_awburst_for_coherent	awburst must be INCR for all Barrier/DVM Transactions, awburst must be INCR or WRAP for all other coherent Transactions	ARM IHI 0022D, Section C3.1.5 on pg C3-163, C3-165	Yes
master_aw_awaddr_for_barrier	awaddr must be all zeros for Barrier	ARM IHI 0022D, Section C3.1.5 on pg C3-166	Yes
master_aw_awaddr_for_cachelinesize	When AWBURST is INCR, the address must be aligned to the cache line size for Cache Line Size Transactions	ARM IHI 0022D, Section C3.1.5 on pg C3-163	Yes
master_aw_awaddr_for_wrbkclean	When AWBURST is INCR, the start and end address must be in the same cache line, for WriteBack/Clean Transactions	ARM IHI 0022D, Section C3.1.5 on pg C3-165	No

AXI ABVIP User Guide
Appendix C: ACE ABVIP Package Contents--ACE Checks

master_aw_awcache_for_coherent	awcache must be normal non-cacheable (AxCACHE=0010) for Barrier/DVM Transactions awcache must be modifiable (AxCACHE[1]=1) for all other coherent Transactions awcache must be modifiable (AxCACHE[1]=1) for all other coherent Transactions	ARM IHI 0022D, Section C3.1.5 on pg C3-163 - C3-166	Yes
master_aw_awlock_for_coherent	awlock must be normal access (AxLOCK=0) for all coherent Write Transactions	ARM IHI 0022D, Section C3.1.5 on pg C3-163 - C3-166	Yes
master_aw_awdomain_by_awcache	Device transactions, as indicated by AxCACHE[1] = 0, must only use AxDOMAIN = 11, System. Cacheable transactions, as indicated by AxCACHE[3:2] != 00, must not use AxDOMAIN = 11, System	ARM IHI 0022D, Section C3.1.1 on C3-158	Yes
master_aw_awdomain_for_coherent	awdomain must be INNER, OUTER for coherent transaction group and Evict awdomain must be INNER, OUTER, NON for memory update transaction group.	ARM IHI 0022D, Section C3.1.3 on pg C3-162, section C3.1.5 on pg C3-165	Yes
master_aw_awbar_for_coherent	awbar must be normal access for all Transactions except Barrier	ARM IHI 0022D, Section C3.1.5 on pg C3-163 - C3-166	Yes
master_aw_awsnoop_legal_values	Legal Write address channel combinations of awsnoop, awbar, awdomain, all unused encodings are reserved	ARM IHI 0022D, Section C3.1.3 on pg C3-162	Yes

AXI ABVIP User Guide
Appendix C: ACE ABVIP Package Contents--ACE Checks

master_aw_awsnoop_for_barrier	All barrier transactions, must have AWSNOOP[2:0] = 000	ARM IHI 0022D, Section C3.1.5 on pg C3-166	Yes
master_aw_ar_no_overlapping_addr	Current write request address should not overlap with any one of the outstanding read and write requests	None	Yes
master_aw_b_awsnoop_no_wrunique_wrbkclean	Complete any outstanding WriteBack or WriteClean transactions before issuing a WriteUnique or WriteLineUnique transaction	ARM IHI 0022D, Section C4.8.6 on pg C4-205	No
master_aw_b_awsnoop_no_wrbkclean_wrunique	No additional WriteBack or WriteClean transactions can be issued until all outstanding WriteUnique or WriteLineUnique transactions are completed.	ARM IHI 0022D, Section C4.8.6 on pg C4-205	No
master_aw_b_ar_r_awcache_no_write_while_maintenance_scl	The master must not issue any further shareable transactions to the same cache line until the cache maintenance transaction is complete.	ARM IHI 0022D, Section C7.2.1 on pg C7-243	Yes
slave_r_ar_ac_cr_rvalid_no_rresp_before_sn_finished_scl	If the interconnect sends a snoop transaction to a master, it must not provide the same master with a read response to a transaction to the same cache line, until it has received a snoop response on <code>CRRESP</code> to the snoop transaction.	ARM IHI 0022D, Section C7.2.1 on pg C7-243	No
slave_r_ar_ac_cr_rvalid_no_ns_rresp_before_sn_finished_scl	If the interconnect sends a snoop transaction to a master, it must not provide the same master with a non shareable read response to a transaction to the same cache line, until it has received a snoop response on <code>CRRESP</code> to the snoop transaction. To enable this check, set parameter <code>RECM_CHECKS_ON</code> to 1.	None	No
assert_aw_wr_tbl_no_overflow	This assertion checks that the number of write requests should not be greater than <code>MAX_PENDING_WR</code>	None	Yes
master_aw_wr_tbl_no_overflow_ace	This assertion checks that the number of write requests should not be greater than <code>MAX_PENDING_WR</code>	None	Yes

AXI ABVIP User Guide
Appendix C: ACE ABVIP Package Contents--ACE Checks

slave_aw_wr_tbl_no_overflow_ace	This assertion checks that the number of write requests should not be greater than MAX_PENDING_WR Note: Removed this property starting from 11.30.046 as the scenario already covered in other property.	None	Yes
master_aw_width_check	AWID, ARID should be of the same width for read and write channels for exclusive transactions	None	Yes
master_aw_w_awlen_exact_len_collision	If data phase and control start simultaneously, the control phase must match that data transfer length.	None	Yes
master_w_aw_wvalid_no_dbc	It ensures that wvalid should only come from master if either its control phase has already done or control phase is coming simultaneously with data phase in non-dbc case (i.e. when DATA_BEFORE_CONTROL_ON is off)	None	Yes
master_w_aw_wlast_exact_len_collision	When data and control phase are coming simultaneously then wlast should only be asserted when awlen matches the data beat count.	None	Yes
master_aw_shareable_tr_awdomain	It must not send Inner Shareable and Outer Shareable transactions	ARM IHI 0022H: Section E1.18.2 on pg E1-399	Yes
1.6 Write Data Checks			
master_w_aw_wstrb_for_coherent	WriteLineUnique transactions are not permitted to use sparse write data strobes.	ARM IHI 0022D, Section C3.1.5 on pg C3-164	Yes
master_w_aw_wstrb_valid_collision	Check strobe for the first beat of a transfer which comes in the same cycle as control for the same transfer.	ARM IHI 0022D, Section A3.4.1 on pg A3.46-A3.47	Yes
master_w_aw_wstrb_valid_curr	Check strobe received in current cycle whose control is already received	ARM IHI 0022D, Section A3.4.1 on pg A3.46-A3.47	Yes

AXI ABVIP User Guide
Appendix C: ACE ABVIP Package Contents--ACE Checks

master_w_aw_wstrb_valid_dbc	In case where data beats are received before control, check the strobes of all the previously received beats when control is received.	ARM IHI 0022D, Section A3.4.1 on pg A3.46-A3.47	Yes
master_w_aw_wstrb_valid_curr_dbc	In case where data beats are received before control, check the strobes of the beat received in the same cycle as control.	ARM IHI 0022D, Section A3.4.1 on pg A3.46-A3.47	Yes
master_aw_awsizes_all_bytelanes_active	This assertion checks that the table never overflows. This ensures protocol verification results are valid. To be used as an ASSERTION only.	None	Yes
assert_w_wr_tbl_no_overflow	This assertion checks that the number of write requests should not be greater than MAX_PENDING_WR. To be used as an ASSERTION only.	None	Yes
slave_b_aw_ac_cr_bvalid_no_bresp_before_sn_finished_scl	If the interconnect sends a snoop transaction to a master, it must not provide the same master with a response to a transaction to the same cache line, until it has received a snoop response on CRRESP to the snoop transaction (indicated by snoop_in_prg_scl_b).	ARM IHI 0022D, Section C6.2 on pg C6-227	No
slave_b_aw_ac_cr_bvalid_no_wns_bresp_before_sn_finished_scl	If the interconnect sends a snoop transaction to a master, it must not provide the same master with a response to a transaction to the same cache line, until it has received a snoop response on CRRESP to the snoop transaction. Set RECM_CHECKS_ON to enable this property.	None	No
master_b_wack_only_after_write	The WACK signal is asserted by a master to indicate that it has completed a write transaction. WACK is asserted for a single cycle and the interconnect is required to accept the WACK signal in a single cycle. The WACK signal must be asserted the cycle after the associated handshake or later.	ARM IHI 0022D, Section C3.5 on pg C3-172	No
master_b_wack_eventually	All pending write transactions will be acknowledged eventually.	None	No
master_w_wr_tbl_no_overflow_ace	This assertion checks that the number of write requests should not be greater than MAX_PENDING_WR Note: Removed this property starting from 11.30.046 as the scenario already covered in other property.	None	Yes

AXI ABVIP User Guide
Appendix C: ACE ABVIP Package Contents--ACE Checks

slave_w_wr_tbl_no_overflow_ace	This assertion checks that the number of write requests should not be greater than MAX_PENDING_WR	None	Yes
slave_w_aw_wready_dbc	If data before control is allowed for slave, it should issue <code>wready</code> with or after <code>awready</code> when DATA_ACCEPT_WITH_OR_AFTER_CONTROL is high	None	Yes
master_w_wack_wait_cycles	All pending write transactions will be acknowledged within MAX_WAIT_CYCLES_WACK cycles	None	No
slave_b_bresp_while_evict	An evict transaction does not propagate downstream and the interconnect is required to generate an <code>OKAY BRESP[1:0] = 0b00</code> write response	ARM IHI 0022D: Section C6.4 on pg C6-242	Yes
1.7 Atomic Access			
master_ar_excl_arcache_low	Exclusive access burst cannot be cacheable and bufferable.	ARM IHI 0022D, Section A7.2.4 on pg A7-97	Yes
master_aw_excl_awcache_low	Exclusive access burst cannot be cacheable and bufferable.	ARM IHI 0022F: Section A7.2.4 on pg A7-99	Yes
slave_b_excl_bresp_no_exokay_supported	The slave must not send an <code>EXOKAY</code> response for a normal write transaction.	ARM IHI 0022D, Section A7.2.3 on pg A7.93	Yes
master_ar_excl_arlen_correct	Exclusive accesses are not permitted to use a burst.	ARM IHI 0022D, Section A7.2.4 on pg A7.93	Yes
master_ar_excl_araddr_aligned	The address of exclusive access must be aligned to the total number of bytes in the transaction.	ARM IHI 0022D, Section A7.2.4 on pg A7.93	Yes

AXI ABVIP User Guide
Appendix C: ACE ABVIP Package Contents--ACE Checks

master_ar_excl_arlen_arsize_correct_bytes	The number of bytes in an exclusive burst.	ARM IHI 0022D, Section A7.2.4 on pg A7.93	Yes
master_ar_excl_arlen_correct	Exclusive accesses are not permitted to use a burst length greater than 16.	ARM IHI 0022D, Section A7.2.4 on pg A7.93	Yes
master_ar_excl_araddr_aligned	The address of exclusive access must be aligned to the total number of bytes in the transaction.	ARM IHI 0022D, Section A7.2.4 on pg A7.93	Yes
master_ar_excl_arlen_arsize_correct_bytes	The number of bytes in an exclusive burst should be 1,2,4,8,16,32,64 or 128.	ARM IHI 0022D, Section A7.2.4 on pg A7.93	Yes
master_ar_excl_load	An exclusive load must use either <code>ReadClean</code> or <code>ReadShared</code> transaction type An exclusive store must be a <code>CleanUnique</code> transaction.	ARM IHI 0022D, Section C9.6 on C9-268	No
master_aw_excl_write	The domain should be always System from write channel for exclusive transactions.	ARM IHI 0022D, Section C9.6 on C9-268	No
slave_r_excl_resp_no_exokay	The slave must not send an EXOKAY response for a normal read transaction.	ARM IHI 0022D, Section A7.2.2 on pg A7.92	Yes
master_ar_excl_cleanunique_no_pending_lock	For clean and exclusive transaction, previous <code>ReadClean</code> or <code>ReadShared</code> transaction must be completed Note: Removing this property starting from 11.30.046, as this feature is no longer supported. ABVIP supports exclusive transactions with domain NOT_SHARED and SYSTEM	None	Yes
master_aw_excl_no_lock_trans_for_sharable	There should not be any lock transaction for sharable domain from the write channel	None	Yes

AXI ABVIP User Guide
Appendix C: ACE ABVIP Package Contents--ACE Checks

master_ar_excl_arsnoop	Exclusive accesses in case of shared domain is possible with ReadClean/ReadShared/CleanUnique	ARM IHI 0022F: Section D9.2.4 on pg D9.99	No
master_excl_ar_excl_store_when_excl_seq_prog	A master must not permit an Exclusive Store transaction at the same time as any transaction performing an Exclusive seq for ACE	ARM IHI 0022F: Section D9.2.3 on pg A7.99	No
master_ar_arlock_excl_shared_tbl_no_overflow	Master should not issue arlock when exclusive access table is full	ARM IHI 0022F	No
master_ar_arlock_excl_shared_domain_tbl_no_overflow	Master should not issue arlock when exclusive access table is full	ARM IHI 0022F	No
slave_r_excl_resp_shared_domain	The EXOKAY response is only permitted for a ReadNoSnoop, ReadClean, ReadShared or CleanUnique transaction	ARM IHI 0022F: Section D9.2.4 on pg D9.99	No
slave_excl_r_wr_resp_no_decerr	The slave can send an EXOKAY/OKAY/SLVERR response only when there is valid exclusive store request for ACE	ARM IHI 0022F: Section A7.2.2 on pg A7.98	No
slave_excl_r_shared_domain_resp_overlapping_addr	Exclusive Write fails when same location is updated since the exclusive read has happened for ACE	ARM IHI 0022F: Section A7.2.4 on pg A7.99	No
slave_ar_arready_excl_shared_domain_tbl_no_overflow	Slave should not accept exclusive access transaction when exclusive access table is full for shared domain	ARM IHI 0022F	No
1.8 Snoop Transaction			
slave_ac_aw_w_no_snoop_before_bresp_wack_scl	If an interconnect provides a master with a response to a write transaction, it should not send the same master a snoop transaction to the same cache line until it has received an acknowledgment of the transaction response on WACK.	ARM IHI 0022D, Section C6.2 on pg C6-227	No

AXI ABVIP User Guide
Appendix C: ACE ABVIP Package Contents--ACE Checks

slave_ac_aw_w_no_snoop_before_wns_bresp_wack_scl	If the interconnect provides a master with a response to a write transaction including <code>writenosnoop</code> , it must not send the same master a snoop transaction to the same cache line until it has received an acknowledgment of the transaction response. Set parameter <code>RECM_CHECK_ON</code> to 1, to enable this property.	None	No
slave_ac_acsnoop_legal_values	Legal Snoop address channel values of <code>acsnoop</code> , all unused encodings are reserved.	ARM IHI 0022D, Section C3.7 on pg C3-176	No
slave_ac_acaddr_wrap_aligned	The snoop address, <code>ACADDR</code> , must be aligned to the data transfer size, which is determined by the width of the snoop data bus in bytes.	ARM IHI 0022D, Section C3.6.2 on pg C3-174	No
slave_ac_ar_r_no_snoop_before_rresp_rack_scl	If the interconnect provides a master with a response to a read transaction, it must not send the same master a snoop transaction to the same cache line until it has received an acknowledgment of the transaction response on <code>RACK</code> .	ARM IHI 0022D, Section C6.2 on pg C6-227	No
slave_ac_ar_r_no_snoop_before_ns_rresp_rack_scl	If the interconnect provides a master with a response to a non-shareable read transaction, it must not send the same master a snoop transaction to the same cache line until it has received an acknowledgment of the transaction response on <code>RACK</code> . Set <code>RECM_CHECKS_ON</code> to enable this property.	ARM IHI 0022D, Section C6.2 on pg C6-227	No
assert_ac_sn_tbl_no_overflow	This assertion checks that the number of snoop requests should not be greater than <code>MAX_PENDING_SNOOP</code>	None	No
master_cr_ac_crvalid	The master must wait for both <code>ACVALID</code> and <code>ACREADY</code> to be asserted before asserting <code>CRVALID</code> .	ARM IHI 0022D, Section C3.9 on pg C3-182	No
master_cr_ac_cd_cresp_for_dbr	Data Transfer must be asserted if: - there is already a snoop data ongoing for the snoop burst (DBR). - there is already snoop data asserted for the same snoop burst (DWR).	ARM IHI 0022D, Section C3.9 on pg C3-182	No

AXI ABVIP User Guide
Appendix C: ACE ABVIP Package Contents--ACE Checks

master_cr_cd_crrsp0_low_with_cdvalid	If Data Transfer is low and cdvalid is high, there must be two snoop transactions pending for response.	ARM IHI 0022D, Section C3.9 on pg C3 182	No
master_cr_ac_cd_crrsp_legal_for_invalidate	Illegal responses are: - IsShared = 1 for a ReadUnique, CleanInvalid, MakeInvalid transaction.	ARM IHI 0022D, Section C3.7 on pg C3 179	No
master_cr_crrsp_legal	Illegal responses are: - DataTransfer = 0 and PassDirty = 1 for any transaction.	ARM IHI 0022D, Section C3.7 on pg C3 179	No
master_cr_aw_b_ac_crrsp_wrbkclean_scl	If a snooped master has a memory update in progress, using either a WriteBack or WriteClean transaction, when it receives a snoop transaction to the same line, then the snooped master must ensure that no other master can perform a memory update at the same time. This is done by giving a snoop response with PassDirty (CRRESP[2]) = 0 and IsShared (CRRESP[3]) = 1, which does not pass the permission to store to the line and does not pass responsibility for updating memory	ARM IHI 0022D, Section C5.2.3 on pg C5-216	No
assert_cr_sn_tbl_no_overflow	This assertion checks that the number of snoop requests should not be greater than MAX_PENDING_SNOOP	None	No
master_cd_ac_cr_cdvalid	The master must wait for both ACVALID and ACREADY to be asserted before asserting CDVALID.	ARM IHI 0022D, Section C3.9 on pg C3 182	No
master_cd_ac_cdlast_exact_len	CDLAST is asserted for the last data transfer of a snoop transaction.	ARM IHI 0022D, Section C3.8 on pg C3 180	No
assert_cd_sn_tbl_no_overflow	This assertion checks that the number of snoop requests should not be greater than MAX_PENDING_SNOOP	None	No
master_ac_acready_eventually	All snoop control requests must be eventually accepted.	None	No

AXI ABVIP User Guide
Appendix C: ACE ABVIP Package Contents--ACE Checks

slave_cr_cready_eventually	All snoop response requests must be eventually given.	None	No
slave_cd_cready_eventually	All snoop data requests must be eventually.	None	No
master_cr_crvalid_eventually	For every snoop address that is presented to a cached master on the snoop address channel, a corresponding response is required on the snoop response channel.	ARM IHI 0022D, Section C3.7 on pg C3 176	No
master_cd_cdvalid_eventually	All snoop request with data beats should come eventually.	None	No
master_cd_cdlast_eventually	All snoop request with data beats should finish eventually.	ARM IHI 0022D, Section C3.8 on pg C3 180	No
master_ac_sn_tbl_no_overflow	This assertion checks that the number of snoop requests should not be greater than MAX_PENDING_SNOOP	None	No
slave_ac_sn_tbl_no_overflow	This assertion checks that the number of snoop requests should not be greater than MAX_PENDING_SNOOP	None	No
master_cr_sn_tbl_no_overflow	This assertion checks that the number of snoop requests should not be greater than MAX_PENDING_SNOOP	None	No
slave_cr_sn_tbl_no_overflow	This assertion checks that the number of snoop requests should not be greater than MAX_PENDING_SNOOP	None	No
master_cd_sn_tbl_no_overflow	This assertion checks that the number of snoop requests should not be greater than MAX_PENDING_SNOOP	None	Yes
slave_cd_sn_tbl_no_overflow	This assertion checks that the number of snoop requests should not be greater than MAX_PENDING_SNOOP	None	No
slave_ac_acready_wait_cycles	The maximum number of cycles for which <code>acready</code> remains low after <code>acvalid</code> going high must not exceed the <code>MAX_WAIT_CYCLES_AC</code> specified.	None	No
slave_cr_crready_wait_cycles	The maximum number of cycles for which <code>crready</code> remains low after <code>crvalid</code> going high must not exceed the <code>MAX_WAIT_CYCLES_CR</code> specified.	None	No
slave_cd_cdready_wait_cycles	The maximum number of cycles for which <code>cdready</code> remains low after <code>cdvalid</code> going high must not exceed the <code>MAX_WAIT_CYCLES_CD</code> specified.	None	No

1.9 Barrier Transaction

AXI ABVIP User Guide
Appendix C: ACE ABVIP Package Contents--ACE Checks

master_ar_bar_check_aruser	ARUSER cannot be reliably transported along with barrier transactions. Therefore, it is recommended that user defined signals be zero for barrier transactions.	ARM IHI 0022E, Section C8.2.3 on pg C8-260	Yes
master_aw_bar_check_awuser	AWUSER cannot be reliably transported along with barrier transactions. Therefore, it is recommended that user defined signals be zero for barrier transactions.	ARM IHI 0022E, Section C8.2.3 on pg C8-260	Yes
slave_r_bar_check_ruser	RUSER cannot be reliably transported along with barrier transactions. Therefore, it is recommended that user defined signals be zero for barrier transactions.	ARM IHI 0022E, Section C8.2.3 on pg C8-260	Yes
master_w_bar_check_wuser	WUSER cannot be reliably transported along with barrier transactions. Therefore, it is recommended that user defined signals be zero for barrier transactions. Note: Removing this property starting from 11.30.046 release, as it is incorrect.	ARM IHI 0022E, Section C8.2.3 on pg C8-260	Yes
slave_b_bar_check_buser	BUSER cannot be reliably transported along with barrier transactions. Therefore, it is recommended that user defined signals be zero for barrier transactions.	ARM IHI 0022E, Section C8.2.3 on pg C8-260	Yes
slave_r_ar_bar_rlast_for_barrier	Barrier responses must have RLAST asserted.	ARM IHI 0022E, Section C8.2.3 on pg C8-260	Yes
master_aw_ar_bar_write_barrier_match	All barrier transactions must be issued in pairs, with a barrier transaction on both the read address and write address channels. This is referred to as issuing a "barrier pair". The two barrier transactions in a barrier pair must have the same AXID, AXBAR, AXDOMAIN, and AXPROT values.	ARM IHI 0022D, Section C8.4.1 on pg C8-254	Yes

AXI ABVIP User Guide
Appendix C: ACE ABVIP Package Contents--ACE Checks

master_ar_aw_bar_read_barrier_match	All barrier transactions must be issued in pairs, with a barrier transaction on both the read address and write address channels. This is referred to as issuing a "barrier pair". The two barrier transactions in a barrier pair must have the same AXID, AXBAR, AXDOMAIN and, AXPROT values.	ARM IHI 0022D, Section C8.4.1 on pg C8-254	Yes
master_ar_aw_bar_read_barrier_id_unused	Barrier transactions are required to use different ID values than are in use for non-barrier transactions. It is permissible for barrier transactions and non-barrier transactions to use the same AXI ID value if one has fully completed before the other is issued.	ARM IHI 0022D, Section C8.4.1 on pg C8-254	Yes
master_ar_aw_bar_read_normal_id_unused	Barrier transactions are required to use different ID values than are in use for non-barrier transactions. It is permissible for barrier transactions and non-barrier transactions to use the same AXI ID value if one has fully completed before the other is issued.	ARM IHI 0022D, Section C8.4.1 on pg C8-254	Yes
master_aw_ar_bar_write_barrier_id_unused	Barrier transactions are required to use different ID values than are in use for non-barrier transactions. It is permissible for barrier transactions and non-barrier transactions to use the same AXI ID value if one has fully completed before the other is issued.	ARM IHI 0022D, Section C8.4.1 on pg C8-254	Yes
master_aw_ar_bar_write_normal_id_unused	Barrier transactions are required to use different ID values than are in use for non-barrier transactions. It is permissible for barrier transactions and non-barrier transactions to use the same AXI ID value if one has fully completed before the other is issued.	ARM IHI 0022D, Section C8.4.1 on pg C8-254	Yes
master_ar_aw_bar_read_barrier_id_match	Barrier pairs must be issued in the same sequence on the read address and write address channels.	ARM IHI 0022D, Section C8.4.1 on pg C8-254	Yes
master_aw_ar_bar_write_barrier_id_match	Barrier pairs must be issued in the same sequence on the read address and write address channels.	ARM IHI 0022D, Section C8.4.1 on pg C8-254	Yes

AXI ABVIP User Guide
Appendix C: ACE ABVIP Package Contents--ACE Checks

master_aw_ar_bar_start_mem_updt	Shareable transactions are limited to memory update transactions like <code>WriteBack</code> , <code>WriteClean</code> , and <code>Evict</code> operations during a barrier.	ARM IHI 0022D, Section C8.4.1 on pg C8-255	No
master_ar_aw_bar_read_barrier_eventually	A master interface that has issued a write barrier on the AW channel, must eventually issue the corresponding read barrier on the AR channel if all other transactions on the AR channel are progressed.	ARM IHI 0022D, Section C8.4.1 on pg C8-255	Yes
master_aw_ar_bar_write_barrier_eventually	Barrier responses must have <code>RLAST</code> asserted.	ARM IHI 0022D, Section C8.2.3 on pg C8-250	Yes
slave_b_aw_bar_bresp_for_barrier	Barrier responses must have <code>BRESP</code> = 'b00.	ARM IHI 0022D, Section C8.2.3 on pg C8-250	Yes
slave_r_ar_bar_rresp_for_barrier	Barrier responses must have <code>RRESP</code> = 'b0000.	ARM IHI 0022D, Section C8.2.3 on pg C8-250	Yes
master_aw_ar_bar_tbl_no_overflow	This assertion checks that the number of barrier requests should not be greater than <code>MAX_PENDING_BAR</code>	None	Yes
master_ar_aw_bar_tbl_no_overflow	This assertion checks that the number of barrier requests should not be greater than <code>MAX_PENDING_BAR</code>	None	Yes
slave_aw_ar_bar_tbl_no_overflow	This assertion checks that the number of barrier requests should not be greater than <code>MAX_PENDING_BAR</code>	None	Yes
slave_ar_aw_bar_tbl_no_overflow	This assertion checks that the number of barrier requests should not be greater than <code>MAX_PENDING_BAR</code> .	None	Yes
assert_param_MAX_PENDING_BAR_legal	ACE master interface must not issue more than 256 outstanding barrier transactions	None	Yes
assert_param_MAX_PENDING_BAR	<code>MAX_PENDING_BAR</code> should be greater than 0	None	Yes
assert_param_MAX_PENDING_SNOOP	<code>MAX_PENDING_SNOOP</code> should be greater than 0	None	Yes

2.0 DVM Transaction

AXI ABVIP User Guide
Appendix C: ACE ABVIP Package Contents--ACE Checks

master_ar_dvm_reserved	DVM all reserved bit format for initial DVM request.	ARM IHI 0022D, Section C12.5 on pg C12-286	No
master_ar_dvm_msg_type_reserved	DVM message type reserved bits for initial DVM request.	ARM IHI 0022D, Section C12.5 on pg C12-286	No
master_ar_addl_dvm_msg_type_reserved	Reserved bits for additional DVM message.	ARM IHI 0022D, Section C12.5 on pg C12-286	No
master_ar_dvm_tlb_inv_msg_format	Supported msg for TLB invalidate.	ARM IHI 0022D, Section C12.7.1 on pg C12-289	No
master_ar_dvm_brn_prd_inv_msg_format	Supported msg for branch predictor invalidate.	ARM IHI 0022D, Section C12.7.2 on pg C12-290	No
master_ar_dvm_phy_inst_cache_inv_msg_format	Supported msg for physical instruction cache invalidate.	ARM IHI 0022D, Section C12.7.3 on pg C12-291	No
master_ar_dvm_virtual_inst_cache_inv_msg_format	Supported msg for virtual instruction cache invalidate	ARM IHI 0022D, Section C12.7.4 on pg C12-292	No

AXI ABVIP User Guide
Appendix C: ACE ABVIP Package Contents--ACE Checks

master_ar_dvm_sync_msg_format	Supported msg for DVM synchronization	ARM IHI 0022D, Section C12.7.5 on pg C12-293	No
master_ar_dvm_hint_msg_format	Supported msg for hint	ARM IHI 0022D, Section C12.7.6 on pg C12-293	No
master_ar_araddr_dvm_complete	For a DVM complete message, ARADDR is defined to be all zeros.	ARM IHI 0022D, Section C12.6 on pg C12-287	Yes
master_ar_dvm_addl_msg_follows_dvm_msg	Multipart DVM messages are always sent as successive transactions.	ARM IHI 0022D, Section C12.6 on pg C12-287	No
master_cr_dvm_addl_msg_resp	For multi-part DVM transactions, a response is provided for each transaction. For such transactions, it is required that the component must provide the same response to each transaction.	ARM IHI 0022D, Section C12.3.4 on pg C12-283	No
slave_ar_r_dvm_mesg_resp_format	RRESP can take value of 0000 or 0010 during DVM message.	ARM IHI 0022D, Section C12.3.4 on pg C12-283	No

AXI ABVIP User Guide
Appendix C: ACE ABVIP Package Contents--ACE Checks

slave_ar_r_dvm_sync_comp_resp_format	An error response is not permitted for a DVM sync or a DVM complete.	ARM IHI 0022D, Section C12.3.4 on pg C12-283, Section C12.7.6 on pg C12-294	No
slave_ar_r_resp_same_for_addl_dvm_msg	For multi-part DVM messages, a response is provided for each transaction. It is required that the response given to all parts of a multiple transaction message are the same.	ARM IHI 0022D, Section C12.3.4 on pg C12-283	No
master_ar_dvm_sync_msg	This is a synchronization transaction that the component issues to check that all previous DVM Operations that it has issued completely.	ARM IHI 0022D, Section C12.1 on pg C12-280	No
master_ar_r_dvm_id_unused	DVM messages are required to use AXI ID values that are not in use by any normal (non-DVM) transactions issued on the AR channel.	ARM IHI 0022D, Section C12.3.5 on pg C12-283, C8.4.1, page C8-256	No
master_ar_r_dvm_sync_eventually	This is a synchronization transaction that the component issues to check that all previous DVM Operations that it has issued completely.	ARM IHI 0022D, Section C12.2 on pg C12-281	No
slave_ar_r_dvm_complete_follows_dvm_sync_eventually	A DVM Sync must complete in a timely manner.	ARM IHI 0022D, Section C12.2 on pg C12-281	No

AXI ABVIP User Guide
Appendix C: ACE ABVIP Package Contents--ACE Checks

slave_ac_dvm_reserved	DVM all reserved bit format for initial DVM requests.	ARM IHI 0022D, Section C12.5 on pg C12-286	Yes
slave_ac_dvm_msg_type_reserved	DVM reserved bit for message types for initial DVM requests.	ARM IHI 0022D, Section C12.5 on pg C12-286	Yes
slave_ac_addl_dvm_msg_type_reserved	Reserved bits for additional DVM messages.	ARM IHI 0022D, Section C12.5 on pg C12-286.	Yes
slave_ac_dvm_tlb_inv_msg_format	Supported msg for TLB invalidate.	ARM IHI 0022D, Section C12.7.1 on pg C12-289	Yes
slave_ac_dvm_brn_prd_inv_msg_format	Supported msg for branch predictor invalidate.	ARM IHI 0022D, Section C12.7.2 on pg C12-290	Yes
slave_ac_dvm_phy_inst_cache_inv_msg_format	Supported msg for physical instruction cache invalidate.	ARM IHI 0022D, Section C12.7.3 on pg C12-291	Yes
slave_ac_dvm_virtual_inst_cache_inv_msg_format	Supported msg for virtual instruction cache invalidate.	ARM IHI 0022D, Section C12.7.4 on pg C12-292	Yes

AXI ABVIP User Guide
Appendix C: ACE ABVIP Package Contents--ACE Checks

slave_ac_dvm_sync_msg_format	Supported msg for DVM Synchronization.	ARM IHI 0022D, Section C12.7.5 on pg C12-293	Yes
slave_ac_dvm_hint_msg_format	Supported msg for hint.	ARM IHI 0022D, Section C12.7.6 on pg C12-293	Yes
slave_ac_acaddr_dvm_complete	For a DVM complete message, ACADDR is defined to be all zeros.	ARM IHI 0022D, Section C12.6 on pg C12-287	Yes
slave_ac_dvm_addl_msg_follows_dvm_msg	Multipart DVM messages are always sent as successive transactions.	ARM IHI 0022D, Section C12.3.3 on pg C12-282	Yes
master_cr_ac_dvm_cresp	<p>If the component can perform the requested DVM action, it must respond by setting <code>CRRRESP</code> = "00000". If the component is unable to perform the requested DVM action, it must respond by setting <code>CRRRESP</code> = "00010" to indicate that it is an unsupported message. A component is not permitted to set <code>CRRRESP</code> to 00010 in response to DVM Sync or a DVM Complete</p> <p>No data transfer is associated with DVM transactions (<code>CRRRESP</code>[0] = 0) All Hint Messages (<code>acaddr</code>[15]=0) must respond with response value <code>CRRRESP</code> set to 0.</p>	ARM IHI 0022D, Section C12.3.4 on pg C12-283, C12.7.6 on pg C12-293	Yes
master_ar_ac_dvm_complete_after_handshake	A DVM Complete on the AR channel must only be issued after the handshake of the associated DVM Sync on the same master's AC channel.	ARM IHI 0022D, Section C12.3.2 on pg C12-282	No

master_ar_ac_dvm_complete_after_dvm_sync	The component should not send DVM complete if it has not received any DVM Sync. Note: Removing this property starting from 11.30.046 release, as it is incorrect	ARM IHI 0022D, Section C12.2 on pg C12-281	No
master_ar_ac_dvm_complete_after_dvm_sync_eventually	DVM complete is issued by a component that has received a number of DVM operations and a DVM Sync.	ARM IHI 0022D, Section C12.2 on pg C12-281	No
master_ar_dvm_araddr_23_16	If ARADDR[5] of the first transaction is de-asserted, ARADDR[23:16] of the first transaction must be all zeros for all defined message types except Hint.	ARM IHI 0022D, Section C12.6 on pg C12-298	Yes
master_ar_dvm_araddr_39_32	If ARADDR[5] of the first transaction is de-asserted, ARADDR[39:32] of the first transaction must be all zeros for all defined message types except Hint.	ARM IHI 0022D, Section C12.6 on pg C12-298	Yes
master_ar_araddr_31_0	If ARADDR[6] of the first transaction is de-asserted, ARADDR[31:24] of the first transaction must be all zeros for all defined message types except Hint.	ARM IHI 0022D, Section C12.6 on pg C12-298	Yes
master_ar_r_dvm_sync_wait_cycles	Pending DVM Sync transactions will be acknowledged within MAX_WAIT_CYCLES_DVM_SYNC cycles.	None	Yes
master_aw_no_bar_trn	Property expects that master will not launch a barrier transaction when parameter BARRIER_ON=0	None	Yes
master_ar_no_bar_trn	Property expects that master will not launch a barrier transaction when parameter BARRIER_ON=0	None	Yes

Coverage Checks

The monitor provides coverage checks to collect coverage information for possible scenarios in an ACE-based design. The coverage information is required for complete verification of the DUV.

To view AXI4 coverage checks, refer to [AXI Monitor Coverage Points](#). To view additional ACE checks, refer to [Table C-7](#).

Table C-7 ACE Monitor Coverage Points

Coverage Assertions	Description
cover_awdomain_nonshared	awdomain can be of non-shared type
cover_awdomain_inner	awdomain can be of inner type
cover_awdomain_outer	awdomain can be of outer type
cover_awdomain_system	awdomain can be of system type
cover_ardomain_nonshared	ardomain can be of non-shared type
cover_ardomain_inner	ardomain can be of inner type
cover_ardomain_outer	ardomain can be of outer type
cover_ardomain_system	ardomain can be of system type
cover_awcache_x_awdomain_device_system	Cover proper awdomain and awcache combination
cover_awcache_x_awdomain_cacheable_notshared	Cover proper awdomain and awcache combination
cover_awcache_x_awdomain_cacheable_inner	Cover proper awdomain and awcache combination
cover_awcache_x_awdomain_cacheable_outer	Cover proper awdomain and awcache combination
cover_awcache_x_awdomain_cacheable_system	Cover proper awdomain and awcache combination
cover_awcache_x_awdomain_writeThBck_notshared	Cover proper awdomain and awcache combination
cover_awcache_x_awdomain_writeThBck_inner	Cover proper awdomain and awcache combination
cover_awcache_x_awdomain_writeThBck_outer	Cover proper awdomain and awcache combination
cover_arcache_x_ardomain_device_system	Cover proper awdomain and awcache combination
cover_arcache_x_ardomain_cacheable_notshared	Cover proper awdomain and awcache combination
cover_arcache_x_ardomain_cacheable_inner	Cover proper awdomain and awcache combination
cover_arcache_x_ardomain_cacheable_outer	Cover proper awdomain and awcache combination
cover_arcache_x_ardomain_cacheable_system	Cover proper awdomain and awcache combination
cover_arcache_x_ardomain_writeThBck_notshared	Cover proper awdomain and awcache combination
cover_arcache_x_ardomain_writeThBck_inner	Cover proper awdomain and awcache combination

AXI ABVIP User Guide
Appendix C: ACE ABVIP Package Contents--Coverage Checks

cover_arcache_x_ardomain_writeThBck_outer	Cover proper awdomain and awcache combination
cover_ar_trgrp_NonSnoop	Cover read non snoop transaction
cover_ar_trgrp_Coherent Cover	read coherent transaction
cover_ar_trgrp_Maintenance	Cover read maintenance transaction
cover_ar_trgrp_DVM	Cover read DVM transaction
cover_ar_trgrp_Barrier	Cover read Barrier transaction
cover_aw_trgrp_NonSnoop	Cover write non snoop transaction
cover_aw_trgrp_Coherent	Cover write coherent transaction
cover_aw_trgrp_Memory	Cover write maintenance transaction
cover_aw_trgrp_Barrier	Cover write DVM transaction
cover_arsnoop_ReadNoSnoop	Cover ReadNoSnoop transaction
cover_arsnoop_ReadOnce	Cover ReadOnce transaction
cover_arsnoop_ReadShared	Cover ReadShared transaction
cover_arsnoop_ReadClean	Cover ReadClean transaction
cover_arsnoop_ReadNotSharedDirty	Cover ReadNotSharedDirty transaction
cover_arsnoop_ReadUnique	Cover ReadUnique transaction
cover_arsnoop_CleanUnique	Cover CleanUnique transaction
cover_arsnoop_MakeUnique	Cover MakeUnique transaction
cover_arsnoop_CleanShared	Cover CleanShared transaction
cover_arsnoop_CleanInvalid	Cover CleanInvalid transaction
cover_arsnoop_MakeInvalid	Cover MakeInvalid transaction
cover_arsnoop_DVM_Complete	Cover DVM Complete transaction
cover_arsnoop_DVM_Message	Cover DVM Message transaction
cover_arsnoop_Barrier	Cover Barrier transaction
cover_rsnoop_ReadNoSnoop_finished	Cover ReadNoSnoop transaction finished
cover_rsnoop_ReadOnce_finished	Cover ReadOnce transaction finished
cover_rsnoop_ReadShared_finished	Cover ReadShared transaction finished

AXI ABVIP User Guide
Appendix C: ACE ABVIP Package Contents--Coverage Checks

cover_rsnoop_ReadClean_finished	Cover ReadClean transaction finished
cover_rsnoop_ReadNotSharedDirty_finished	Cover ReadNotSharedDirty transaction finished
cover_rsnoop_ReadUnique_finished	Cover ReadUnique transaction finished
cover_rsnoop_CleanUnique_finished	Cover CleanUnique transaction finished
cover_rsnoop_MakeUnique_finished	Cover MakeUnique transaction finished
cover_rsnoop_CleanShared_finished	Cover CleanShared transaction finished
cover_rsnoop_CleanInvalid_finished	Cover CleanInvalid transaction finished
cover_rsnoop_MakeInvalid_finished	Cover MakeInvalid transaction finished
cover_rsnoop_DVM_Complete_finished	Cover DVM Complete transaction finished
cover_rsnoop_DVM_Message_finished	Cover DVM Message transaction finished
cover_rsnoop_Barrier_finished	Cover Barrier transaction finished
cover_awsnoop_WriteNoSnoop	Cover WriteNoSnoop transaction
cover_awsnoop_WriteUnique	Cover WriteUnique transaction
cover_awsnoop_WriteLineunique	Cover WriteLineUnique transaction
cover_awsnoop_WriteClean	Cover WriteClean transaction
cover_awsnoop_WriteBack	Cover WriteBack transaction
cover_awsnoop_Evict	Cover Evict transaction
cover_awsnoop_Barrier	Cover Barrier transaction
cover_bsnoop_WriteNoSnoop_finished	Cover WriteNoSnoop transaction finished
cover_bsnoop_WriteUnique_finished	Cover WriteUnique transaction finished
cover_bsnoop_WriteLineunique_finished	Cover WriteLineUnique transaction finished
cover_bsnoop_WriteClean_finished	Cover WriteClean transaction finished
cover_bsnoop_WriteBack_finished	Cover WriteBack transaction finished
cover_bsnoop_WriteBack_finished	Cover WriteBack transaction finished

AXI ABVIP User Guide
Appendix C: ACE ABVIP Package Contents--Coverage Checks

cover_bsnoop_Barrier_finished	Cover Barrier transaction finished
cover_arbar_NormalRespect	Cover proper read barrier combination
cover_arbar_MemoryBarrier	Cover proper read barrier combination
cover_arbar_NormalIgnore	Cover proper read barrier combination
cover_arbar_SyncBarrier	Cover proper read barrier combination
cover_awbar_NormalRespect	Cover proper write barrier combination
cover_awbar_MemoryBarrier	Cover proper write barrier combination
cover_awbar_NormalIgnore	Cover proper write barrier combination
cover_awbar_SyncBarrier	Cover proper write barrier combination
cover_rresp_PassDirty	Cover rresp with PassDirty bit set
cover_rresp_PassClean	Cover rresp with PassClean bit set
cover_rresp_IsShared	Cover rresp with IsShared bit set
cover_rresp_IsUnique	Cover rresp with IsUnique bit set
cover_acprot_privileged	Cover proper acprot combination
cover_acprot_normal	Cover proper acprot combination
cover_acprot_nonsecure	Cover proper acprot combination
cover_acprot_secure	Cover proper acprot combination
cover_acprot_instruction	Cover proper acprot combination
cover_acprot_data	Cover proper acprot combination
cover_acsnoop_ReadOnce	Cover ReadOnce snoop transaction
cover_acsnoop_ReadShared	Cover ReadShared snoop transaction
cover_acsnoop_ReadClean	Cover ReadClean snoop transaction
cover_acsnoop_ReadNotSharedDirty	Cover ReadNotSharedDirty snoop transaction
cover_acsnoop_ReadUnique	Cover ReadUnique snoop transaction
cover_acsnoop_CleanShared	Cover CleanShared snoop transaction

AXI ABVIP User Guide
Appendix C: ACE ABVIP Package Contents--Coverage Checks

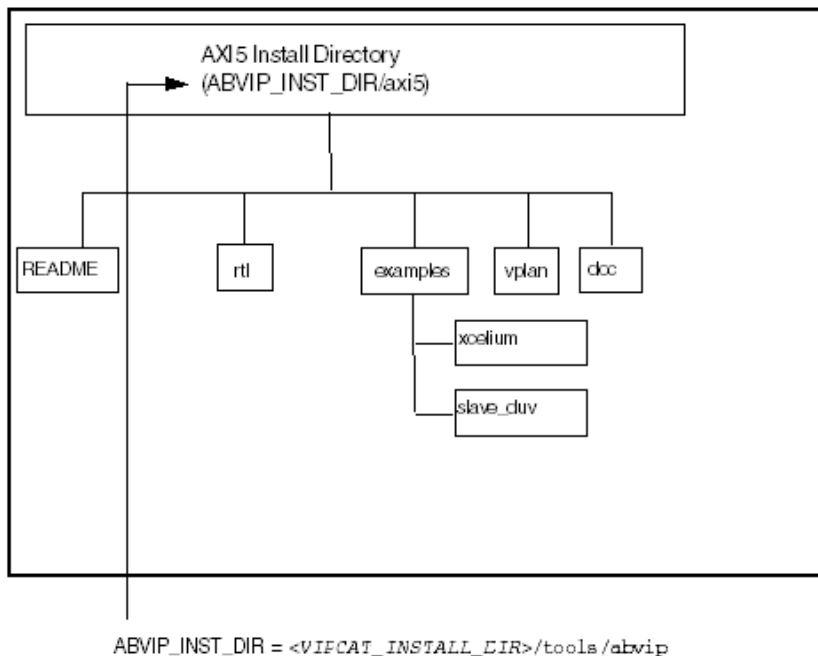
cover_acsnoop_CleanInvalid	Cover CleanInvalid snoop transaction
cover_acsnoop_MakeInvalid	Cover MakeInvalid snoop transaction
cover_acsnoop_DVM_Complete	Cover DVM Complete snoop transaction
cover_acsnoop_DVM_Message	Cover DVM Message snoop transaction
cover_crresp_DataTransfer	Cover crresp with data transfer bit set
cover_crresp_NoDataTransfer	Cover crresp with no data transfer
cover_crresp_Error	Cover crresp with error
cover_crresp_NoError	Cover crresp with no error
cover_crresp_PassDirty	Cover crresp with PassDirty bit set
cover_crresp_PassClean	Cover crresp with PassClean bit set
cover_crresp_IsShared	Cover crresp with IsShared bit set
cover_crresp_IsUnique	Cover crresp with IsUnique bit set
cover_crresp_WasUnique	Cover crresp with WasUnique bit set
cover_crresp_WasShared	Cover crresp with WasShared bit set
cover_rack	Cover rack
cover_wack	Cover wack
cover_seq_bar_bresp_out_of_order	Cover barrier transaction with out of order response
cover_seq_evict_bresp_in_order	Cover evict transaction with in order response
cover_seq_normal_evict_pair	Cover normal and evict transaction pair with response in order
cover_seq_normal_barrier_pair	Cover barrier and normal transaction with out of order response (<i>awlen</i> =0)
cover_seq_normal_len2_barrier_pair	Cover barrier and normal transaction with out of order response (<i>awlen</i> =1)
cover_seq_snoop_dbr_order	Cover snoop data before request
cover_seq_dbc_barrier_collision	Cover DBC barrier collision
cover_seq_dbc_len2_barrier_pair	Cover DBC barrier pair with out of order response

Appendix D: AXI5 ABVIP Package Contents

- [Package Contents](#)
- [AXI5 ABVIP Pin-Level Interface](#)
- [AXI5 ABVIP Parameters](#)
- [AXI5 Checks](#)
 - [Compliance Checks](#)
- [Coverage Checks](#)

This appendix describes the contents of the AXI5 ABVIP package. [Figure B-1](#) shows the package directory structure.

Figure B-1 Directory Structure of the AXI5 ABVIP Package



Package Contents

[Table B-1](#) shows the contents of the AXI5 installation directory.

Table B-1 Contents of the AXI5 installation directory

README	Mentions the contents of the package.
rtl	Contains the code for AXI5 ABVIP.
examples	Contains the examples of AXI5 ABVIP to be used in formal and simulation based verification.
doc	Contains the AXI ABVIP user guide.

[Table B-2](#) shows the files stored in the rtl folder.

Table B-2 Contents of the rtl Folder

Version.v	Prints the version number of the ABVIP.
cdn_abvip_axi5_monitor.svp	Encrypted SVA based monitor with protocol compliance checks and functional covers.

cdn_abvip_axi5_master.svp	Encrypted AXI4 master driving AXI5 master interface signals and checking AXI5 slave interface signals.
cdn_abvip_axi5_slave.svp	Encrypted AXI4 slave driving AXI5 slave interface signals and checking AXI5 master interface signals.
cdn_abvip_amba5_g_master.svp	Included in the cdn_abvip_axi5_master.svp. Drives the AXI5-G version of master interface signal and checks slave interface signals
cdn_abvip_amba5_g_slave.svp	Included in the cdn_abvip_axi5_slave.svp. Drives the AXI5-G version of slave interface signal and checks master interface signals
cdn_abvip_amba5_g_monitor.svp	Included in the cdn_abvip_axi5_monitor.svp. Checks the AXI5-G version of AXI5 master and AXI5 slave interface signals
cdn_abvip_amba5_h_master.svp	Included in the cdn_abvip_axi5_master.svp. Drives the AXI5-H version of master interface signal and checks slave interface signals
cdn_abvip_amba5_h_slave.svp	Included in the cdn_abvip_axi5_slave.svp. Drives the AXI5-H version of slave interface signal and checks master interface signals
cdn_abvip_amba5_h_monitor.svp	Included in the cdn_abvip_axi5_monitor.svp. Checks the AXI5-H version of AXI5 master and AXI5 slave interface signals
axi4_defines.svh	AXI definition file

The example folder contains examples for running AXI5 ABVIP in different configurations. Each subdirectory contains a README file that describes the steps for running the example and the content of that folder.

Table B-3 Contents of the Example Folder in AXI5 installation directory

README	Lists the contents of example directory.
slave_duv	Contains Verilog example for verifying AXI5 slave using AXI5 ABVIP in formal using Jasper
xcelium	Contains Verilog example for verifying AXI5 slave using AXI5 ABVIP in simulation using Xcelium

AXI5 ABVIP Pin-Level Interface

The pin-level interface is a collection of HDL signals contained in an HDL module and connected to the DUV. A description of the pin-level interface signals for the AXI5 ABVIP is listed in [Table B-4](#).

Table B-4 The AXI5 ABVIP Pin Interface

Pin	Source	Description
acclk	Clock source	System clock
aresetn	Reset controller	System reset--active low
awid[ID_WIDTH-1:0]	Master	Write address ID--AWID
awidunq	Master	Write Unique ID Indicator - AWIDUNQ
awaddr[ADDR_WIDTH-1:0]	Master	Write address--AWADDR
awlen[LEN_WIDTH-1:0]	Master	Burst length--AWLEN

awsize[SIZE_WIDTH-1:0]	Master	Burst size--AWSIZE
awburst[BURST_WIDTH-1:0]	Master	Burst type--AWBURST
awlock	Master	Lock type--AWLOCK
awmpam[10:0]	Master	Write Channel MPAM information – AWMPAM
awcache[CACHE_WIDTH-1:0]	Master	Cache type--AWCACHE
awprot[PROT_WIDTH-1:0]	Master	Protection type--AWPROT
awqos[QOS_WIDTH-1:0]	Master	Quality of service-AWQOS
awregion[REGION_WIDTH-1:0]	Master	Address region-AWREGION
awuser[AUSER_WIDTH-1:0]	Master	User signal--AWUSER
awvalid	Master	Write address valid--AWVALID
awready	Slave	Write address ready--AWREADY
wdata[DATA_WIDTH-1:0]	Master	Write data--WDATA
wstrb[DATA_WIDTH/8-1:0]	Master	Write strobes--WSTRB
wuser[WUSER_WIDTH-1:0]	Master	User signal- wuser
wlast	Master	Write last--WLAST
wvalid	Master	Write valid--WVALID
wready	Slave	Write ready--WREADY
bid[ID_WIDTH-1:0]	Slave	Response ID--BID
bidunq	Slave	Write Response Unique ID Indicator - BIDUNQ
bresp[BRESP_WIDTH-1:0]	Slave	Write response--BRESP
buser[BUSER_WIDTH-1:0]	Slave	User signal--buser
bvalid	Slave	Write response valid--BVALID
bready	Master	Response ready--BREADY
arid[ID_WIDTH-1:0]	Master	Read address ID--ARID
aridunq	Master	Read Unique ID Indicator--ARIDUNQ
araddr[ADDR_WIDTH-1:0]	Master	Read address--ARADDR
arlen[LEN_WIDTH-1:0]	Master	Burst length--ARLEN
arsize[SIZE_WIDTH-1:0]	Master	Burst size--ARSIZE
arburst[BURST_WIDTH-1:0]	Master	Burst type--ARBURST
arlock	Master	Lock type--ARLOCK
archunken	Master	Read Data Chunking Enable-- ARCHUNKEN
armpam[10:0]	Master	Read Channel MPAM information – ARMPAM
arcache[CACHE_WIDTH-1:0]	Master	Cache type--ARCACHE
arprot[PROT_WIDTH-1:0]	Master	Protection type--ARPROT

arqos[QOS_WIDTH-1:0]	Master	Quality of service-ARQOS
aruser[ARUSER_WIDTH-1:0]	Master	User signal -ARUSER
arregion[REGION_WIDTH-1:0]	Master	Address region-ARREGION
arvalid	Master	Read address valid--ARVALID
aready	Slave	Read address ready-ARREADY
rid[ID_WIDTH-1:0]	Slave	Read ID tag--RID
ridunq	Slave	Read Response Unique ID Indicator--RIDUNQ
rdata[DATA_WIDTH-1:0]	Slave	Read data--RDATA
rresp[RRESP_WIDTH-1:0]	Slave	Read response--RRESP
rlast	Slave	Read last--RLAST
ruser[USER_WIDTH-1:0]	Slave	User signal-RUSER
rvalid	Slave	Read valid--RVALID
rready	Slave	Read ready--RREADY
rchunkv	Slave	Valid signal of rchunknum and rchunkstrb – RCHUNKV
rchunknum [CHUNKNUM_WIDTH-1:0]	Slave	Read Data Chunk Number--RCHUNKNUM
rchunkstrb [CHUNKSTB_WIDTH-1:0]	Slave	Read Data Chunk Strobe --RCHUNKSTRB
csysreq	Clock controller	System low-power request--CSYSREQ
csysack	Peripheral device	Low-power request acknowledgment--CSYSACK
cactive	Peripheral device	Clock active--CACTIVE
armusecsid	Master	Read Address secure stream identifier
armmusid	Master	Read Address stream identifier
armmussidv	Master	Read Address substream identifier valid
armmussid	Master	Read Address substream identifier
armmuatst	Master	Read Address address translated
awmmusecsid	Master	Write Address secure stream identifier
awmmusid	Master	Write Address stream identifier
awmmussidv	Master	Write Address substream identifier valid
awmmussid	Master	Write Address substream identifier
awmmuatst	Master	Write Address address translated
rpoison	Slave	Read Data Poison
wpoison	Master	Write Data Poison
rdatachk	Slave	Read Data Check
wdatachk	Master	Write Data Check

varqosaccept	Slave	Read Address QoS Accept
vawqosaccept	Slave	Write Address QoS Accept
artrace	Master	Read address trace
rtrace	Slave	Read data trace
awtrace	Master	Write address trace
wtrace	Slave	Write data trace
btrace	Slave	Write response trace
arloop	Master	Read address loopback
rloop	Slave	Read data loopback
awloop	Master	Write address loopback
bloop	Slave	Write response loopback
awakeup	Master	low power wakeup signal
arnsaid	Master	Read address non-secure access identifier
awnsaid	Master	Write address non-secure access identifier
awvalidchk	Master	Check signal for AWVALID
awreadychk	Slave	Check signal for AWREADY
awidchk [IDWRCHK_WIDTH-1:0]	Master	Check signal for AWID
awaddrchk [ADDRCHK_WIDTH-1:0]	Master	Check signal for AWADDR
awlenchk	Master	Check signal for AWLEN
awctlchk0	Master	Check signal for AWSIZE, AWBURST, AWLOCK, AWPROT
awctlchk1	Master	Check signal for AWREGION, AWCACHE, AWQOS
awuserchk [AWUSERCHK_WIDTH-1:0]	Master	Check signal for AWUSER
awtracechk	Master	Check signal for AWTRACE
awloopchk	Master	Check signal for AWLOOP
awmmuchk	Master	Check signal for AWMMUATST, AWMMUSECSID, AWMMUSSIDV
awmmusidchk [MMUSIDCHK_WIDTH-1:0]	Master	Check signal for AWMMUSID
awmmussidchk [MMUSSIDCHK_WIDTH-1:0]	Master	Check signal for AWMMUSSID
awnsaidchk	Master	Check signal for AWNSAID
awmpamchk	Master	Check signal for AWMPAM
awidunqchk	Master	Check signal for AWIDUNQ
wvalidchk	Master	Check signal for WVALID
wreadychk	Slave	Check signal for WREADY

wdatachk [DATACHK_WIDTH-1:0]	Master	Check signal for WDATA
wstrbchk [WSTRBCHK_WIDTH-1:0]	Master	Check signal forWSTRB
wlastchk	Master	Check signal for WLAST
wuserchk [WUSERCHK_WIDTH-1:0]	Master	Check signal for WUSER
wpoisonchk [WPOISONCHK_WIDTH-1:0]	Master	Check signal for WPOISON
wtracechk	Master	Check signal for WTRACE
bvalidchk	Slave	Check signal for BVALID
breaychk	Master	Check signal for BREADY
bidchk [IDWRCHK_WIDTH-1:0]	Slave	Check signal for BID
brespchk	Slave	Check signal for BRESP
btracechk	Slave	Check signal for BTRACE
bloopchk	Slave	Check signal for BLOOP
buserchk [BUSERCHK_WIDTH-1:0]	Slave	Check signal for BUSER
arreadychk	Slave	Check signal for ARREADY
arvalidchk	Master	Check signal for ARVALID
aridchk [IDRDCHK_WIDTH-1:0]	Master	Check signal for ARID
araddrchk [ADDRCHK_WIDTH-1:0]	Master	Check signal for ARADDR
arlenchk	Master	Check signal for ARLEN
arctlchk0	Master	Check signal for ARSIZE, ARBURST, ARLOCK, ARPROT
arctlchk1	Master	Check signal for ARREGION, ARCACHE, ARQOS
aruserchk [ARUSERCHK_WIDTH-1:0]	Master	Check signal for ARUSER
artracechk	Master	Check signal for ARTRACE
arloopchk	Master	Check signal for ARLOOP
armmuchk	Master	Check signal for ARMMUATST, ARMMUSECSID, ARMMUSSIDV
armmusidchk [MMUSIDCHK_WIDTH-1:0]	Master	Check signal for ARMMUSID
armmuSSIDchk [MMUSSIDCHK_WIDTH-1:0]	Master	Check signal for ARMMUSSID
arnsaidchk	Master	Check signal for ARNSAID
aridunqchk	Master	Check signal for ARIDUNQ
armpamchk	Master	Check signal for ARMPAM
rvalidchk	Slave	Check signal for RVALID
rreadychk	Master	Check signal for RREADY

ridchk [IDRDCHK_WIDTH-1:0]	Slave	Check signal for RID
rdatachk [DATACHK_WIDTH - 1:0]	Slave	Check signal for RDATA
rrespchk	Slave	Check signal for RRESP
rloopchk	Slave	Check signal for RLOOP
ruserchk [RUSERCHK_WIDTH-1:0]	Slave	Check signal for RUSER
rtracechk	Slave	Check signal for RTRACE
rchunkchk	Slave	Check signal for RCHUNKEN, RCHUNKSTRB, RCHUNKNUM
rlastchk	Slave	Check signal for RLAST
rpoisonchk [POISONCHK_WIDTH-1:0]	Slave	Check signal for RPOISON
vawqosacceptchk	Slave	Check signal for VAWQOSACCEPT
varqosacceptchk	Slave	Check signal for VARQOSACCEPT
awakeupchk	Master	Check signal for AWAKEUP
awtagop	Master	Write request tag operation
artagop	Master	Read request tag operation
wtag	Master	The tag that is associated with write data
wtagupdate	Master	Indicates which tags must be written to memory in an Update operation
rtag	Slave	The tag that is associated with read data
btagmatch	Slave	Indicates the result of a tag comparison on a write transaction
bcomp	Slave	Response flag that indicates that a write is observable
wtagchk	Master	Parity Check signal for write tag
rtagchk	Slave	Parity Check signal for read tag
armmuflow	Master	Indicates the SMMU flow for managing translation faults ARMMUFLOW
awmmuflow	Master	Indicates the SMMU flow for managing translation faults AWMMUFLOW

AXI5 ABVIP Parameters

The AXI5 ABVIP is highly configurable. You can configure the parameters listed in [Table B-5](#) as required.




Table B-5 AXI5 Monitor Parameters

Parameter	Description
ADDR_WIDTH	Address bus width. The default value is 32. The minimum value of ADDR_WIDTH during byte strobe checking (BYTE_STROBE_ON is 1) must be 8.

AXI ABVIP User Guide
Appendix D: AXI5 ABVIP Package Contents--AXI5 ABVIP Parameters

DATA_WIDTH	Data bus width. The default value is 64.
ID_WIDTH	The number of bits needed to capture the number of IDs supported. The default value is 2.
LEN_WIDTH	The number of bits needed to capture the length of data per request. The default value is 4.
AXI5_LITE	If set to 1, the AXI5-Lite mode functionality is enabled. The default value is 0, which means AXI5.
ALL_STROBES_HIGH_ON	Whether all strobes are set to high. The default value is 0, which means this parameter is not set.
ALLOW_SPARSE_STROBE	When this parameter is set, master ABVIP can drive <code>wstrb</code> bits to 0 for one or more valid byte lanes. The default value of this parameter is 1. This parameter is effective only if parameter <code>BYTE_STROBE_ON</code> is set to 1.
BYTE_STROBE_ON	Whether write byte strobe (WSTRB) checking is required. The default value is 0, which means byte strobe checking is not required.
COVERAGE_ON	Whether coverage generation is set. The default value is 0, which is no coverage.
DATA_BEFORE_CONTROL_ON	Whether data before control is supported. The default value is 1, which means data before control is supported.
DATA_ACCEPT_WITH_OR_AFTER_CONTROL	Whether slave can accept data with or after control. The default value is 0, which means data can be accepted before control.
ERROR_CHECK_ON	Set to 1 disable the ABVIP's max address boundary properties. The address will be allowed to cross the max boundary range. The default value is 0.
EXCL_ACCESS_ON	Whether exclusive access is supported. The default value is 0, which means no exclusive access.
READ_INTERLEAVE_ON	Whether read data interleaving is supported. The default value is 1, which means read interleave mode is supported.
WRITE_INTERLEAVE_ON	Whether write data interleaving is supported. The default value is 1, which means write interleave mode is supported.
READ_RESP_IN_ORDER_ON	Whether read response in order is supported. The default value is 0, which means the read response can come in any order.
WRITE_RESP_IN_ORDER_ON	Whether write response in order is supported. The default value is 0, which means the write response can come in any order.


AXI ABVIP User Guide
Appendix D: AXI5 ABVIP Package Contents--AXI5 ABVIP Parameters

RST_CHECKS_ON	<p>Whether reset checks are supported.</p> <p>The default value is 0, which means no reset checks.</p>
XCHECKS_ON	<p>Support of X-checks. The default value is 0, meaning no X-checks.</p> <div>  Use the Jasper elaborate switch “<code>enable_sva_isunknown</code>” for X-Prop analysis when this parameter is enabled. </div>
CONFIG_PARAM_CHECKS	<p>Set to 1 will enable the ABVIP parameter checks.</p> <p>The default value is 1.</p>
CONFIG_LIVENESS	<p>Liveness properties will be created for the scenarios whose <code>MAX_WAIT_CYCLES_*</code> parameter is 0. Setting this parameter to 0 disables all liveness properties.</p> <p>The default value is 1.</p> <p>When enabled, all liveness assumptions and assertions are enabled.</p> <p>Also, if <code>CONFIG_LIVENESS</code> and <code>DEADLOCK_CHKS_ON</code>, both parameters are enabled then <code>DEADLOCK_CHKS_ON</code> will take priority. It will disable all liveness properties and will enable only deadlock checks.</p> <div>  Define macro <code>JG_ABVIP_STRONG_SEMANTICS</code> to enable strong semantics for liveness properties when using sv09 and above. </div>
DEADLOCK_CHKS_ON	<p>Liveness properties will be created for the scenarios whose <code>MAX_WAIT_CYCLES_*</code> parameter is 0. Setting this parameter to 1 enables all deadlock checks.</p> <p>The default value is 0.</p> <p>When enabled, then only liveness asserts are enabled and constraints are disabled.</p> <p>Also, if <code>CONFIG_LIVENESS</code> and <code>DEADLOCK_CHKS_ON</code>, both parameters are enabled then <code>DEADLOCK_CHKS_ON</code> will take priority. It will disable all liveness properties and will enable only deadlock checks.</p> <div>  Define macro <code>JG_ABVIP_STRONG_SEMANTICS</code> to enable strong semantics for deadlock checks when using sv09 and above. </div>
CONFIG_STABLE_CHECKS	<p>Set to 1 will enable the ABVIP property checks for signals.</p> <p>The default value is 1</p>
CONFIG_RDATA_MASKED	<p>Set to 0 will enable the X-propagation and stability properties to apply to full Read data.</p> <p>Set to 1 will enable the X-propagation and stability properties to apply to valid byte lane of Read data .</p> <p>The default value is 1.</p>

AXI ABVIP User Guide




Appendix D: AXI5 ABVIP Package Contents--AXI5 ABVIP Parameters

CONFIG_RD_FULL_STRB	<p>Set to 0 will enable full read strobe not required for the read transaction.</p> <p>Set to 1 will enable full read strobe required for the read transaction.</p> <p>The default value is 0.</p>
CONFIG_CSR_INTERFACE	<p>This parameter is used to connect ABVIP with Jasper CSR App. The default value of this parameter is 0. You can set the parameter value to 1 to enable ABVIP with CSR checkers.</p>
CONFIG_CSR_FEED_EARLY_W	<p>Tune internal CSR logic behavior:</p> <ul style="list-style-type: none"> Set to 0 will feed write data to CSR checker on write response transaction. Set to 1 will feed write data to CSR checker on write data transaction. <p>The default value is 0.</p> <div> <p> Arrays of size MAX_PENDING_WR*WLAXLEN*DATA_WIDTH will be created in ABVIP when this parameter is 0.</p> <p>-disable_auto_bbox or -no_bbox_m or -no_bbox_i or -bbox_a switch needs to be added in elaborate command to avoid black-boxing of these arrays in ABVIP.</p> <p>Elaboration time will increase in this mode.</p> </div>
CONFIG_CSR_FEED_ON_BVALID	<p>Tune internal CSR logic behavior when CONFIG_CSR_FEED_EARLY_W = 0,</p> <ul style="list-style-type: none"> Set to 0 will feed write data to CSR checker on write response channel <code>bvalid</code> & <code>breedy</code> assertion. Set to 1 will feed write data to CSR checker on write response channel <code>bvalid</code> assertion. <p>The default value is 0.</p>
CONFIG_CSR_FEED_ON_RVALID	<p>Tune internal CSR logic behavior,</p> <p>Set to 0 will feed read data to CSR checker on the read response channel <code>rvalid</code> and <code>rready</code> assertion.</p> <p>Set to 1 will feed read data to CSR checker on the read response channel <code>rvalid</code> assertion.</p> <p>The default value is 0.</p>



MAX_WAIT_CYCLES_ON	<p>If set to 1, all MAX_WAIT_CYCLES_* parameters can be used. If set to 0, the MAX_WAIT_CYCLES_* parameters are assigned a value of infinity by using liveness properties. This value means, for instance, if an AWVALID goes high, AWREADY must eventually go high.</p> <div>  Define macro JG_ABVIP_STRONG_SEMANTICS to enable strong semantics for liveness properties when using sv09 and above. </div>
MAX_WAIT_CYCLES_AW	The maximum number of cycles for which AWREADY is low after AWVALID goes high.
MAX_WAIT_CYCLES_W	The maximum number of cycles for which WREADY is low after WVALID goes high.
MAX_WAIT_CYCLES_B	The maximum number of cycles for which BREADY is low after BVALID goes high.
MAX_WAIT_CYCLES_AR	The maximum number of cycles for which ARREADY is low after ARVALID goes high.
MAX_WAIT_CYCLES_R	The maximum number of cycles for which RREADY is low after RVALID goes high.
MAX_WAIT_CYCLES_AW_W	The maximum number of cycles for which write signals(WVALID, WREADY) are low after AWVALID goes high
MAX_WAIT_CYCLES_W_AW	The maximum number of cycles for which write address channel signals(AWVALID, AWREADY) are low after WVALID goes high
MAX_WAIT_CYCLES_W_B	The maximum number of cycles for which write response is low after wvalid goes high.
MAX_WAIT_CYCLES_AW_B	The maximum number of cycles for which write response is low after AWVALID goes high
MAX_WAIT_CYCLES_AR_R	The maximum number of cycles for which read response is low after ARVALID goes high
READONLY_INTERFACE	Set to 1 will enable Read only channels and disable Write channels
WRITEONLY_INTERFACE	Set to 1 will enable Write only channels and disable Read channels
MAXLEN	<p>The maximum number of beats that can be transferred in a burst for the read/write requests. The default value is 256. This will constrain AXLLEN value to be less than MAXLEN.</p> <p>When CONFIG_CSR_INTERFACE is set, the default value of this parameter is 16.</p>
MAXLEN_RD	<p>The maximum number of beats that can be transferred in a burst for read requests. Default value is 256. This constrains ARLEN value to be less than MAXLEN_RD.</p> <p>When CONFIG_CSR_INTERFACE is set, the default value of this parameter is 16.</p>
MAXLEN_WR	<p>The maximum number of beats that can be transferred in a burst for write requests. Default value is 256. This constrains AWLEN value to be less than MAXLEN_WR.</p> <p>When CONFIG_CSR_INTERFACE is set, the default value of this parameter is 16.</p>

AXI ABVIP User Guide
Appendix D: AXI5 ABVIP Package Contents--AXI5 ABVIP Parameters

CDNS_READY_OVFLOW_CHECKS	This parameter is used to enable/disable properties that prevent the assertion of ready signal when table/tracker is full. The default value is 1. It is recommended to set it to 0 when doing verification with cci400.
RECM_CHECKS_EXCL_ON	Enable recommended exclusive checks. Default value is 1.
RECM_CHKS_ALL_ON	Setting to 1 enables all recommended checks irrespective of the corresponding feature parameter. Default value is 0.
RECM_CHECKS_ON	Setting to 1 enables recommended checks only if the corresponding feature is set to 1. Default value is 0.
RECM_ASSUME_ON	The default value of this parameter is 1'b1. If the parameter is set to 0 then recommended checks are enabled and recommended constraints are disabled. When set to 1, it enables both recommended checks and constraints.
CONFIG_SIMULATION DYNAMIC_DATA_WDTH_EN DATA_WIDTH_inB CDDATA_WIDTH_inB	These parameters should be left to their default values.
CONFIG_UNTRANSLATED_TRN	<p>This parameter is used to enable/disable untranslated transaction properties. Default value is version 1. The required signals are supported for version 2 of untranslated transactions if the value of this parameter is 2</p> <div>  CDNS_AMBA5_H parameter should be set to 1, to enable version 2 properties </div>
CONFIG_ATOMIC_TRN	This parameter is used to enable/disable Atomic transaction properties. Default value is 1.
POISON_WIDTH	Poison bit width. The default value is DATA_WIDTH/64
CONFIG_POISON_SIGNAL_CHKS	This parameter is used to enable/disable poison signal checks. Default value is 1.
CONFIG_DATA_CHK_SIGNAL_CHKS	This parameter is used to enable/disable data to check signal properties. Default value is 1.
CONFIG_QOS_SIGNAL_CHKS	This parameter is used to enable/disable QoS signal properties. Default value is 1.
CONFIG_TRACE_SIGNAL_CHKS	This parameter is used to enable/disable trace signal properties. Default value is 1.
CONFIG_USER_LOOPBACK_CHKS	This parameter is used to enable/disable loopback signal properties. Default value is 1.
CONFIG_LP_WAKEUP_SIG_CHKS	<p>This parameter is used to enable/disable low power wakeup signal properties. Default value is 1.</p> <div>  LOW_POWER_ON should also be set to 1, to enable wakeup signal properties. </div>
LOW_POWER_ON	This parameter is used to enable/disable low power properties. Default value is 0.
UPDATE_EXCL_TBL_ON_RD_RESP	<p>When this parameter is set to 1, the exclusive access trackers are updated based on the response of the exclusive read. This allows you to be less firm on the previous exclusive write response if an overwriting exclusive read is received.</p> <p>The default value of this parameter is 0.</p>

CDNS_ABVIP_STOP_OVERFLOW	<p>Disable ABVIP assumptions to prevent table overflow.</p> <p>When set to 1, master ABVIP will not generate new *valid signals when MAX_PENDING transactions are outstanding. Also, when set to 1, slave ABVIP will not generate *ready signals when MAX_PENDING transactions are outstanding.</p> <p>Default value is 1, which means ABVIP overflow assumptions are present.</p>
CDNS_ABVIP_STOP_OVERFLOW_RD	<p>Disable ABVIP assumptions to prevent Read table overflow. Default value is the same as that of CDNS_ABVIP_STOP_OVERFLOW i.e. 1, which means ABVIP overflow assumptions are present.</p>
CDNS_ABVIP_STOP_OVERFLOW_WR	<p>Disable ABVIP assumptions to prevent Write table overflow. Default value is the same as that of CDNS_ABVIP_STOP_OVERFLOW i.e. 1, which means ABVIP overflow assumptions are present.</p>
CDNS_READY_OVFLOW_CHECKS	<p>When set to 1, master or monitor ABVIP will check that slave does not generate *ready signals when MAX_PENDING transactions are outstanding.</p> <p>Default value is 1.</p>
CDNS_VALID_OVFLOW_CHECKS	<p>When set to 1, slave or monitor ABVIP will check that master does not generate *valid signals when MAX_PENDING transactions are outstanding.</p> <p>Default value is 1.</p>
CDNS_AMBA5_G	<p>This parameter enables the AMBA5-G features.</p> <p>The default value of this parameter is 0.</p>
CDNS_AMBA5_H	<p>This parameter enables the AMBA5-H features.</p> <p>The default value of this parameter is 0.</p>
CONFIG_UNQ_ID_INDICATOR	<p>When set to 1, it enables the Unique ID Support feature. The default value of this parameter is 0.</p> <div>  CDNS_AMBA5_G parameter should be set to 1, to enable the unique ID signal properties. </div>
CONFIG_MPAM_SUPPORT	<p>Parameter <i>CONFIG_MPAM_SUPPORT</i> enables the interface to support Memory Partitioning and Mapping technology.</p> <p>The default value of this parameter is 0.</p> <div>  CDNS_AMBA5_G parameter should be set to 1 to enable the mpam signal properties. </div>
CONFIG_READ_DATA_CHUNKING	<p>It enables the slave interface to send the read data in the chunk's forms.</p> <p>The default value of this parameter is 0.</p> <div>  CDNS_AMBA5_G parameter should be set to 1 to enable the read data to chunk signal properties. </div>
CHUNKNUM_WIDTH	width of the rchunknum signal
CHUNKSTB_WIDTH	width of the rchunkstrb signal

AXI ABVIP User Guide
Appendix D: AXI5 ABVIP Package Contents--AXI5 ABVIP Parameters

CONFIG_PARITY_CHECK_TYPE	<p>It enables the Parity feature in the interface</p> <p>Set to 0: Disable parity Check scheme, Set to 1: to enable data parity bit, Set to 2: enable all Parity bit</p> <div>  CDNS_AMBA5_G parameter should be set to 1 to enable the read data to chunk signal properties. </div>
CONFIG_ODD_PARITY_BYTE_DATA	<p>Set to 0: Disable byte data parity check</p> <div>  CDNS_AMBA5_G parameter should be set to 1 to enable the read data to chunk signal properties. </div>
CONFIG_ODD_PARITY_BYTE_ALL	<p>Set to 0: Disable byte all parity check</p> <div>  CDNS_AMBA5_G parameter should be set to 1 to enable the read data to chunk signal properties. </div>
ADDRCHK_WIDTH	Width of the Address check signal
ARUSERCHK_WIDTH	Width of Read Address User check signal
AWUSERCHK_WIDTH	Width of Write Address User check signal
IDRDCHK_WIDTH	Width of Read ID check signal
DATACHK_WIDTH	Width of Data check signal
MMUSIDCHK_WIDTH	Width of mmusid check signal
MMUSSIDCHK_WIDTH	Width of mmussid check signal
IDWRCHK_WIDTH	Width of Write ID check signal
WSTRBCHK_WIDTH	Width of Write Strb check signal
WUSERCHK_WIDTH	Width of Write user check signal
BUSERCHK_WIDTH	Width of B user check signal
POISONCHK_WIDTH	Width of Poison check signal
RUSERCHK_WIDTH	Width of Read user check signal
EXCL_ACCESS_WRITE_MATCH_EX_READ	<p>Set to 0 to enable the master to generate the illegal transactions in the exclusive access. By default, it will generate the exclusive write transaction matching with exclusive read</p>
EXCL_ACCESS_SLAVE_ON	By default it is 1. Set to 0 to disable exclusive access of slave
EXCL_ACCESS_WR_RESP_OKAY_ALLOWED	<p>Set to 1 to allow that slave response can send EXOKAY or OKAY response in case of valid exclusive access. By default, It is set to 0 means slave can send only exokay response</p>
CDNS_AMBA5_H	<p>This parameter enables the AMBA5-H features.</p> <p>The default value of this parameter is 0.</p>
CONFIG_REGULAR_TRANSACTION	<p>When set to 1, it enables the Regular Transactions feature.</p> <p>The default value of this parameter is 0.</p> <div>  CDNS_AMBA5_H parameter should be set to 1, to enable the properties of the regular transaction. </div>

<i>CONFIG_CONSISTENT_DECERR</i>	when set to 1, it enables DECERR to be signaled for every beat of read data, or no beats of read data within each cache line of data. Default value is 0
<i>MAX_TRANSACTION_BYTES</i>	defines the maximum size of a transaction in bytes. Default value is 4096
<i>USER_REQ_WIDTH</i>	new width of REQ User defined signals. Default value is 32 <div>⚠ CDNS_AMBA5_H parameter should be set to 1, to enable this parameter.</div>
<i>USER_DATA_WIDTH</i>	new width of DATA User defined signals. Default value is 32 <div>⚠ CDNS_AMBA5_H parameter should be set to 1, to enable this parameter.</div>
<i>USER_RESP_WIDTH</i>	new width of RESP User defined signals. Default value is 32 <div>⚠ CDNS_AMBA5_H parameter should be set to 1, to enable this parameter.</div>
<i>LOOP_R_WIDTH</i>	defines the width of Read address loop signals. Default value is 8 <div>⚠ CDNS_AMBA5_H parameter should be set to 1, to enable this parameter.</div>
<i>LOOP_W_WIDTH</i>	defines the width of Write address loop signals. Default value is 8 <div>⚠ CDNS_AMBA5_H parameter should be set to 1, to enable the properties of regular transaction.</div>
<i>CONFIG_MTE_SUPPORT</i>	Define the Support of Memory tagging. Set to 1: Value of MTE is Standard, Set to 2: the value of MTE is Basic <div>⚠ CDNS_AMBA5_H parameter should be set to 1, to enable the memory tagging properties</div>
<i>TAGUPDATE_WIDTH</i>	Define the width of the wtagupdate signal, the value depends on Data_width
<i>TAG_WIDTH</i>	Define the width of tag signal in read and write channel, the value depends on Data_width
<i>TAGCHK_WIDTH</i>	Define the width of the tagchk parity signal

AXI5 Checks

Compliance Checks

Table B-6 describes the SVA compliance checks for the master and slave in the AXI5 ABVIP. The properties have been grouped into sub-categories based on different aspects of the AXI5 protocol. The property naming conventions are also shown in the table.

Table B-6 AXI5 Monitor Compliance Checks

AXI ABVIP User Guide
Appendix D: AXI5 ABVIP Package Contents--AXI5 Checks

Property Name	Description	AXI Spec
1.1 Master Stable		
master_ar_arvalid_stable	The ARVALID signal must remain stable until ARREADY goes high.	ARM IHI 0022f Section A3.2.1 A3-43
master_ar_arid_stable	The ARID signal must remain stable when ARVALID is high until ARREADY goes high.	ARM IHI 0022f Section A3.2.1 A3-41
master_ar_araddr_stable	The ARADDR signal must remain stable when ARVALID is high until ARREADY goes high.	ARM IHI 0022f Section A3.2.1 A3-43
master_ar_arlen_stable	The ARLEN signal must remain stable when ARVALID is high until ARREADY goes high.	ARM IHI 0022f Section A3.2.1 A3-41
master_ar_arsize_stable	The ARSIZE signal must remain stable when ARVALID is high until ARREADY goes high.	ARM IHI 0022f Section A3.2.1 A3-41
master_ar_arburst_stable	The ARBURST signal must remain stable when ARVALID is high until ARREADY goes high.	ARM IHI 0022f Section A3.2.1 A3-41
master_ar_arlock_stable	The ARLOCK signal must remain stable when ARVALID is high until ARREADY goes high.	ARM IHI 0022f Section A3.2.1 A3-41
master_ar_arprot_stable	The ARPROT signal must remain stable when ARVALID is high until ARREADY goes high.	ARM IHI 0022f Section A3.2.1 A3-43
master_ar_arcache_stable	The ARCACHE signal must remain stable when ARVALID is high until ARREADY goes high.	ARM IHI 0022f Section A3.2.1 A3-41
master_ar_arqos_stable	The ARQOS signal must remain stable when ARVALID is high until ARREADY goes high.	ARM IHI 0022f Section A3.2.1 A3-41
master_ar_arregion_stable	The ARREGION signal must remain stable when ARVALID is high until ARREADY goes high.	ARM IHI 0022f Section A3.2.1 A3-41
master_ar_aruser_stable	The ARUSER signal must remain stable when ARVALID is high until ARREADY goes high.	ARM IHI 0022f Section A3.2.1 A3-41
master_aw_awvalid_stable	The AWVALID signal must remain stable until AWREADY goes high.	ARM IHI 0022f Section A3.2.1 A3-41
master_aw_awid_stable	The AWID signal must remain stable when AWVALID is high until AWREADY goes high.	ARM IHI 0022f Section A3.2.1 A3-41
master_aw_awaddr_stable	The AWADDR signal must remain stable when AWVALID is high until AWREADY goes high.	ARM IHI 0022f Section A3.2.1 A3-41
master_aw_awlen_stable	The AWLEN signal must remain stable when AWVALID is high until AWREADY goes high.	ARM IHI 0022f Section A3.2.1 A3-41
master_aw_awsized_stable	The AWSIZE signal must remain stable when AWVALID is high until AWREADY goes high.	ARM IHI 0022f Section A3.2.1 A3-41

AXI ABVIP User Guide
Appendix D: AXI5 ABVIP Package Contents--AXI5 Checks

master_aw_awburst_stable	The AWBURST signal must remain stable when AWVALID is high until AWREADY goes high.	ARM IHI 0022f Section A3.2.1 A3-41
master_aw_awlock_stable	The AWLOCK signal must remain stable when AWVALID is high until AWREADY goes high.	ARM IHI 0022f Section A3.2.1 A3-41
master_aw_awprot_stable	The AWPROT signal must remain stable when AWVALID is high until AWREADY goes high.	ARM IHI 0022f Section A3.2.1 A3-41
master_aw_awcache_stable	The AWCACHE signal must remain stable when AWVALID is high until AWREADY goes high.	ARM IHI 0022f Section A3.2.1 A3-41
master_aw_awqos_stable	The AWQOS signal must remain stable when AWVALID is high until AWREADY goes high.	ARM IHI 0022f Section A3.2.1 A3-41
master_aw_awregion_stable	The AWREGION signal must remain stable when AWVALID is high until AWREADY goes high.	ARM IHI 0022f Section A3.2.1 A3-41
master_aw_awuser_stable	The AWUSER signal must remain stable when AWVALID is high until AWREADY goes high.	ARM IHI 0022f Section A3.2.1 A3-41
master_w_wvalid_stable	The WVALID signal must remain stable until WREADY goes high.	ARM IHI 0022f Section A3.2.1 A3-41
master_w_wdata_stable	The WDATA signal must remain stable when WVALID is high until WREADY goes high.	ARM IHI 0022f Section A3.2.1 A3-41
master_w_wstrb_stable	The WSTRB signal must remain stable when WVALID is high until WREADY goes high.	ARM IHI 0022f Section A3.2.1 A3-41
master_w_wlast_stable	The WLAST signal must remain stable when WVALID is high until WREADY goes high.	ARM IHI 0022f Section A3.2.1 A3-41
master_armmusecsid_stable	The <code>armmusecsid</code> signal must remain stable when ARVALID is high until ARREADY goes high	ARM IHI 0022f Section E2.12 E2-370
master_armmusid_stable	The <code>armmusid</code> signal must remain stable when ARVALID is high until ARREADY goes high	ARM IHI 0022f Section E2.12 E2-370
master_armmuSSIDv_stable	The <code>armmuSSIDv</code> signal must remain stable when ARVALID is high until ARREADY goes high	ARM IHI 0022f Section E2.12 E2-370
master_armmuSSID_stable	The <code>armmuSSID</code> signal must remain stable when ARVALID is high until ARREADY goes high	ARM IHI 0022f Section E2.12 E2-370
master_armmuatst_stable	The <code>armmuatst</code> signal must remain stable when ARVALID is high until ARREADY goes high	ARM IHI 0022f Section E2.12 E2-370
master_artrace_stable	The <code>artrace</code> signal must remain stable when ARVALID is high until ARREADY goes high	ARM IHI 0022f Section E2.6 o E2-355

AXI ABVIP User Guide
Appendix D: AXI5 ABVIP Package Contents--AXI5 Checks

master_arloop_stable	The arloop signal must remain stable when ARVALID is high until ARREADY goes high	ARM IHI 0022f Section E2.7 o E2-357
master_awmmusecsid_stable	The awmmusecsid signal must remain stable when AWVALID is high until AWREADY goes high	ARM IHI 0022f Section E2.12 E2-370
master_awmmusid_stable	The awmmusid signal must remain stable when AWVALID is high until AWREADY goes high	ARM IHI 0022f Section E2.12 E2-370
master_awmmussidv_stable	The awmmussidv signal must remain stable when AWVALID is high until AWREADY goes high	ARM IHI 0022f Section E2.12 E2-370
master_awmmussid_stable	The awmmussidv signal must remain stable when AWVALID is high until AWREADY goes high	ARM IHI 0022f Section E2.12 E2-370
master_awmmuatst_stable	The awmmuatst signal must remain stable when AWVALID is high until AWREADY goes high	ARM IHI 0022f Section E2.12 E2-370
master_awatop_stable	The awatop signal must remain stable when AWVALID is high until AWREADY goes high	ARM IHI 0022f Section E2.1 o E2-336
master_wpoison_stable	The wpoison signal must remain stable when WVALID is high until WREADY goes high	ARM IHI 0022f Section E2.5.2 E2-352
master_wdatachk_stable	The wdatachk signal must remain stable when WVALID is high until WREADY goes high	ARM IHI 0022f Section E2.5.2 E2-352
master_awtrace_stable	The awtrace signal must remain stable when AWVALID is high until AWREADY goes high	ARM IHI 0022f Section E2.6 o E2-355
master_wtrace_stable	The wtrace signal must remain stable when WVALID is high until WREADY goes high	ARM IHI 0022f Section E2.6 o E2-355
master_awloop_stable	The awloop signal must remain stable when AWVALID is high until AWREADY goes high	ARM IHI 0022f Section E2.7 o E2-357
master_ar_arpam_stable	arpam should remain stable if arvalid && !arready	ARM IHI 0022f pg E1.16-381
master_archunken_stable	archunken should remain stable if arvalid && !arready	ARM IHI 0022f pg E1.16-381
master_aridunq_stable	aridunq should remain stable if arvalid && !arready	ARM IHI 0022f pg E1.16-381
master_aw_awmpam_stable	awmpam should remain stable if awvalid && !awready	ARM IHI 0022f pg E1.16-381
master_awidunq_stable	awidunq should remain stable if awvalid && !awready	ARM IHI 0022f pg E1.16-381
master_ar_araddrchk_stable	araddrchk should remain stable if arvalid && !arready	ARM IHI 0022f pg E2.4-389
master_ar_aridchk_stable	aridchk should remain stable if arvalid && !arready	ARM IHI 0022f pg E2.4-389
master_ar_artracechk_stable	artracechk should remain stable if arvalid && !arready	ARM IHI 0022f pg E2.4-389

AXI ABVIP User Guide
Appendix D: AXI5 ABVIP Package Contents--AXI5 Checks

master_ar_arloopchk_stable	arloopchk should remain stable if arvalid && !arready	ARM IHI 0022(pg E2.4-389
master_ar_armpamchk_stable	armpamchk should remain stable if arvalid && !arready	ARM IHI 0022(pg E2.4-389
master_ar_aruserchk_stable	aruserchk should remain stable if arvalid && !arready	ARM IHI 0022(pg E2.4-389
master_ar_arlenchk_stable	arlenchk should remain stable if arvalid && !arready	ARM IHI 0022(pg E2.4-389
master_ar_arctlchk1_stable	arctlchk0 should remain stable if arvalid && !arready	ARM IHI 0022(pg E2.4-389
master_ar_arctlchk0_stable	arctlchk1 should remain stable if arvalid && !arready	ARM IHI 0022(pg E2.4-389
master_ar_arnsaidchk_stable	arnsaidchk should remain stable if arvalid && !arready	ARM IHI 0022(pg E2.4-389
master_ar_armmuchk_stable	armmuchk should remain stable if arvalid && !arready	ARM IHI 0022(pg E2.4-389
master_ar_armmusidchk_stable	armmusidchk should remain stable if arvalid && !arready	ARM IHI 0022(pg E2.4-389
master_ar_armmussidchk_stable	armmussidchk should remain stable if arvalid && !arready	ARM IHI 0022(pg E2.4-389
master_aw_awaddrchk_stable	awaddrchk should remain stable if awvalid && !awready	ARM IHI 0022(pg E2.4-389
master_aw_awidchk_stable	awidchk should remain stable if awvalid && !awready	ARM IHI 0022(pg E2.4-389
master_aw_awlenchk_stable	awlenchk should remain stable if awvalid && !awready	ARM IHI 0022(pg E2.4-389
master_aw_awtracechk_stable	awtracechk should remain stable if awvalid && !awready	ARM IHI 0022(pg E2.4-389
master_aw_awloopchk_stable	awloopchk should remain stable if awvalid && !awready	ARM IHI 0022(pg E2.4-389
master_aw_awmpamchk_stable	awmpamchk should remain stable if awvalid && !awready	ARM IHI 0022(pg E2.4-389
master_aw_awuserchk_stable	awuserchk should remain stable if awvalid && !awready	ARM IHI 0022(pg E2.4-389
master_aw_awnsaidchk_stable	awnsaidchk should remain stable if awvalid && !awready	ARM IHI 0022(pg E2.4-389
master_aw_awmmuchk_stable	awmmuchk should remain stable if awvalid && !awready	ARM IHI 0022(pg E2.4-389
master_aw_awmmusidchk_stable	awmmusidchk should remain stable if awvalid && !awready	ARM IHI 0022(pg E2.4-389
master_aw_awmmussidchk_stable	awmmussidchk should remain stable if awvalid && !awready	ARM IHI 0022(pg E2.4-389
master_aw_awctlchk0_stable	awctlchk0 should remain stable if awvalid && !awready	ARM IHI 0022(pg E2.4-389
master_aw_awctlchk1_stable	awctlchk1 should remain stable if awvalid && !awready	ARM IHI 0022(pg E2.4-389
master_w_wdatachk_stable	wdatachk should remain stable if wvalid && !wready	ARM IHI 0022(pg E2.4-389
master_w_wlastchk_stable	wlastchk should remain stable if wvalid && !wready	

AXI ABVIP User Guide
Appendix D: AXI5 ABVIP Package Contents--AXI5 Checks

master_w_wstrbchk_stable	wstrbchk should remain stable if wvalid && !wready	ARM IHI 0022f pg E2.4-389
master_w_wuserchk_stable	wuserchk should remain stable if wvalid && !wready	ARM IHI 0022f pg E2.4-389
master_w_wpoisonchk_stable	wpoisonchk should remain stable if wvalid && !wready	ARM IHI 0022f pg E2.4-389
master_w_wtracechk_stable	wtracechk should remain stable if wvalid && !wready	ARM IHI 0022f pg E2.4-389
master_aw_awtagop_stable	awtagop should remain stable if awvalid and !awready	ARM IHI 0022f Section E2.6 o 391
master_ar_artagop_stable	artagop should remain stable if arvalid and !arready	ARM IHI 0022f Section E2.6 o 391
master_w_wtag_stable	wtag should remain stable if wvalid and !wready	ARM IHI 0022f Section E2.6 o 391
master_w_wtagchk_stable	wtagchk should remain stable if wvalid and !wready	ARM IHI 0022f Section E2.6 o 391
master_w_wtagupdate_stable	wtagupdate should remain stable if wvalid and !wready	ARM IHI 0022f Section E2.6 o 391
master_aw_awmmuflow_stable	awmmuflow should remain stable if awvalid and !awready	ARM IHI 0022f Section E2.6 o 391
master_ar_armmuflow_stable	armmuflow should remain stable if arvalid and !arready	ARM IHI 0022f Section E2.6 o 391

1.2 Master Read

master_ar_araddr_wrap_aligned	WRAP bursts should be aligned to the transfer size	ARM IHI 0022f Section A3.4.1 A3-50
master_ar_arlen_less_than_TRANS_COUNT	arlen should be less than TRANS_COUNT	ARM IHI 0022f Section A3.4.1 A3-50
master_ar_araddr_wrap_arlen	WRAP bursts should be of length 2,4,8 or 16.	ARM IHI 0022f Section A3.4.1 A3-50
master_ar_araddr_fixed_arlen	FIXED bursts should be of length up to 16.	ARM IHI 0022f Section A3.4.1 A3-48
master_ar_araddr_never_cross_4K_boundary	Read burst must not cross 4K boundary.	ARM IHI 0022f Section A3.4.1 A3-48
master_ar_arburst_no_reserved	Read burst cannot take reserved value 2'b11.	ARM IHI 0022f Section A3.4.1 A3-50
master_ar_arsize_lte_datawidth	Size of any transfer must not exceed the data bus width of the components in the transaction.	ARM IHI 0022f Section A3.4.1 A3-49

AXI ABVIP User Guide
Appendix D: AXI5 ABVIP Package Contents--AXI5 Checks

master_ar_arcache_no_ra_wa_for_uncacheable	Read Allocate(RA) bit and Write Allocate(WA) bit of arcache, arcache[2] and arcache[3] must not be HIGH if modifiable bit arcache[1] is low.	ARM IHI 0022f Section A4.2 o A4-63
master_ar_armmussid_valid	For transactions which do not specify a sub-stream identifier, AXMMUSSID must be driven to 0	ARM IHI 0022f Section E2.12. pg E2-371
master_ar_armmuatst_asserted	For transactions that have already undergone translation as indicated by armmuatst asserted, AXMMUSECSID/AXMMUSSIDV must be driven to 0	ARM IHI 0022f Section E2.12. pg E2-371
master_ar_armmusecsid_deasserted	Master must drive arprot[1] high indicating non-sec transaction, when armmusecsid is deasserted	ARM IHI 0022f Section E2.12. pg E2-371
master_ar_width_check	AWID, ARID should be of same width for read and write channels for exclusive transactions	None
master_ar_mpam_combination	If NS is 0, MPAN_NS cannot be 1, it is illegal combination for read address channel	ARM IHI 0022f section 6.4.2
master_aridunq_no_outstanding_rd_trans	When ARIDUNQ is asserted, there must be no outstanding read transactions from this master with the same ARID value	ARM IHI 0022f pg E1.16-381
master_no_rd_req_with_same_arid	A master must not issue a read request with the same ARID as an outstanding read transaction that had ARIDUNQ asserted	ARM IHI 0022f pg E1.16-381
master_aridunq_ridunq_match	If ARIDUNQ is deasserted or asserted for a request, the corresponding RIDUNQ signals must be deasserted or asserted for all response beats for that transaction	ARM IHI 0022f pg E1.16-381
master_ar_arvalidchk_gen	arvalidchk should be generated whenever aresetn is asserted	ARM IHI 0022f pg E2.6-392
master_ar_arreadychk_gen	arreadychk should be generated whenever aresetn is asserted	ARM IHI 0022f pg E2.6-392
master_ar_aridchk_gen	aridchk should be generated whenever arvalid is asserted	ARM IHI 0022f pg E2.6-392
master_ar_araddrchk_gen	araddrchk should be generated whenever arvalid is asserted	ARM IHI 0022f pg E2.6-392
master_ar_arlenchk_gen	arlenchk should be generated whenever arvalid is asserted	ARM IHI 0022f pg E2.6-392
master_ar_aruserchk_gen	aruserchk should be generated whenever arvalid is asserted	ARM IHI 0022f pg E2.6-392
master_ar_arloopchk_gen	arloopchk should be generated whenever arvalid is asserted	ARM IHI 0022f pg E2.6-392
master_ar_artracechk_gen	artracechk should be generated whenever arvalid is asserted	ARM IHI 0022f pg E2.6-392
master_ar_arctlchk0_gen	arctlchk0 should be generated whenever arvalid is asserted	ARM IHI 0022f pg E2.6-392
master_ar_arctlchk1_gen	arctlchk1 should be generated whenever arvalid is asserted	ARM IHI 0022f pg E2.6-392
master_ar_arnsaidchk_gen	arnsaidchk should be generated whenever arvalid is asserted	ARM IHI 0022f pg E2.6-392
master_ar_armmuchk_gen	armmuchk should be generated whenever arvalid is asserted	ARM IHI 0022f pg E2.6-392
master_ar_armmusidchk_gen	armmusidchk should be generated whenever arvalid is asserted	ARM IHI 0022f pg E2.6-392

AXI ABVIP User Guide
Appendix D: AXI5 ABVIP Package Contents--AXI5 Checks

master_ar_armmussidchk_gen	armmussidchk should be generated whenever arvalid is asserted	ARM IHI 0022F pg E2.6-392
master_ar_armpamchk_gen	armpamchk should be generated whenever arvalid is asserted	ARM IHI 0022F pg E2.6-392
master_ar_trsize_never_cross_max_bytes	Read transaction must not cross maximum transaction bytes	ARM IHI 0022F Section E1.18. pg E1-400
master_ar_trsize_never_cross_max_boundary	Read transaction must not cross maximum transaction boundary	ARM IHI 0022F Section E1.18. pg E1-400
1.3 Master Write		
master_aw_awaddr_wrap_aligned	The start address must be aligned to the size of the transfer.	ARM IHI 0022F Section A3.4.1 A3-50
master_aw_awaddr_wrap_awlen	The length of the burst must be 2,4,8 or 16.	ARM IHI 0022F Section A3.4.1 A3-50
master_aw_awaddr_fixed_awlen	FIXED bursts should be of length up to 16.	ARM IHI 0022F Section A3.4.1 A3-48
master_aw_awaddr_never_cross_4K_boundary	Bursts must not cross 4KB boundaries to prevent them from crossing boundaries between slaves.	ARM IHI 0022F Section A3.4.1 A3-48
master_aw_awburst_no_reserved	Burst signal cannot take the reserved value 2'b11.	ARM IHI 0022F Section A3.4.1 A3-50
master_aw_awsizelte_datawidth	The size of any transfer must not exceed the data bus width of the components in the transaction.	ARM IHI 0022F Section A3.4.1 A3-49
master_aw_awcache_no_ra_wa_non_modifiable	Write Cache RA and WA bits must de-asserted if the modifiable bit is not set.	ARM IHI 0022F Section A4.2 o A4-63
master_w_aw_wlast_exact_len	wlast should be asserted in the last beat of a write transaction.	ARM IHI 0022F Section A3.2.2 A3-43
master_w_aw_wstrb_valid_collision	Master should drive the valid write data strobe in case of simultaneous write control information	ARM IHI 0022F section A3.3 or A3-44
master_w_aw_wstrb_valid_curr	Master should drive the valid write data strobe in case of already received corresponding write control information	ARM IHI 0022F section A3.3 or A3-44
master_w_aw_wstrb_valid_dbc	Master should drive valid write data strobe in case of write data before corresponding write control information	ARM IHI 0022F section A3.3 or A3-44
master_w_aw_wstrb_valid_curr_dbc	Master should drive valid write data strobe in case of simultaneous corresponding write control information	ARM IHI 0022F section A3.3 or A3-44
master_w_wstrb_all_active	If Master always performs full data bus width write transactions. The default value for write strobes is all signals asserted	ARM IHI 0022F section A9.3.4 A9-115
master_aw_awsizel_all_bytelanes_active	When a master performs full data bus width write transactions, AWSIZE must be equal to WRITE_TRSIZE.	NoneARM IHI 0022F: section on pg A9-111

AXI ABVIP User Guide
Appendix D: AXI5 ABVIP Package Contents--AXI5 Checks

master_aw_width_check	AWID, ARID should be of the same width for read and write channels for exclusive transactions	None
master_aw_atomicID_width_check	AWID, ARID should be of the same width for read and write channels for atomic transactions	None
master_aw_w_awlen_exact_len_collision	If data phase and control starts simultaneously, the control phase must match that data transfer length.	None
master_aw_w_wlast_awlen_exact_len_overflow	If data phase comes with last, then subsequent control phase must match that data transfer length exactly.	ARM IHI 0022f Section A3.3 o A3-44
master_aw_w_notwlast_awlen_exact_len_overflow	If data phase doesn't come with last, then subsequent control phase must have data transfer length greater than 0.	ARM IHI 0022f Section A3.3 o A3-44
master_w_aw_wvalid_no_dbc	It ensures that wvalid should only come from master if either its control phase has already done or control phase is coming simultaneously with data phase in non-dbc case (i.e. when DATA_BEFORE_CONTROL_ON is off)	None
master_w_aw_wlast_exact_len_collision	When data and control phase are coming simultaneously then wlast should only be asserted when awlen matches the data beat count.	None
master_aw_mpam_combination	If NS is 0, MPAN_NS cannot be 1, it is illegal combination for write address channel.	ARM IHI 0022f section 6.4.2
master_awidunq_awid_no_outstanding_wr_trans	When AWIDUNQ is asserted, there must be no outstanding write transactions from this master with the same AWID value	ARM IHI 0022f section 6.4.2
master_no_wr_req_with_same_awid	A master must not issue a write request with the same AWID as an outstanding write transaction that had AWIDUNQ asserted	ARM IHI 0022f section 6.4.2
master_awidunq_bidunq_match	If AWIDUNQ is deasserted or asserted for a request, the corresponding BIDUNQ signal must be deasserted or asserted for all response beats for that transaction	ARM IHI 0022f section 6.4.2
master_aw_awvalidchk_gen	awvalidchk should be generated whenever aresetn is asserted	ARM IHI 0022f pg E2.6-392
master_aw_awreadychk_gen	awreadychk should be generated whenever aresetn is asserted	ARM IHI 0022f pg E2.6-392
master_aw_awidchk_gen	awidchk should be generated whenever awvalid is asserted	ARM IHI 0022f pg E2.6-392
master_aw_awaddrchk_gen	awaddrchk should be generated whenever awvalid is asserted	ARM IHI 0022f pg E2.6-392
master_aw_awlenchk_gen	awlenchk should be generated whenever awvalid is asserted	ARM IHI 0022f pg E2.6-392
master_aw_awloopchk_gen	awloopchk should be generated whenever awvalid is asserted	ARM IHI 0022f pg E2.6-392
master_aw_awtracechk_gen	awtracechk should be generated whenever awvalid is asserted	ARM IHI 0022f pg E2.6-392
master_aw_awuserchk_gen	awuserchk should be generated whenever awvalid is asserted	ARM IHI 0022f pg E2.6-392
master_aw_awctlchk0_gen	awctlchk0 should be generated whenever awvalid is asserted	ARM IHI 0022f pg E2.6-392
master_aw_awctlchk1_gen	awctlchk1 should be generated whenever awvalid is asserted	ARM IHI 0022f pg E2.6-392
master_aw_awnsaidchk_gen	awnsaidchk should be generated whenever awvalid is asserted	ARM IHI 0022f pg E2.6-392

AXI ABVIP User Guide
Appendix D: AXI5 ABVIP Package Contents--AXI5 Checks

master_aw_awmpamchk_gen	awmpamchk should be generated whenever awvalid is asserted	ARM IHI 0022f pg E2.6-392
master_aw_awmmuchk_gen	awmmuchk should be generated whenever awvalid is asserted	ARM IHI 0022f pg E2.6-392
master_aw_awmmusidchk_gen	awmmusidchk should be generated whenever awvalid is asserted	ARM IHI 0022f pg E2.6-392
master_aw_awmussidchk_gen	awmussidchk should be generated whenever awvalid is asserted	ARM IHI 0022f pg E2.6-392
master_w_wvalidchk_gen	wvalidchk should be generated whenever aresetn is asserted	ARM IHI 0022f pg E2.6-392
master_w_wreadychk_gen	wreadychk should be generated whenever aresetn is asserted	ARM IHI 0022f pg E2.6-392
master_w_wdatachk_gen	wdatachk should be generated whenever wvalid is asserted	ARM IHI 0022f pg E2.6-392
master_w_wlastchk_gen	wlastchk should be generated whenever wvalid is asserted	ARM IHI 0022f pg E2.6-392
master_w_wloopchk_gen	wloopchk should be generated whenever wvalid is asserted	ARM IHI 0022f pg E2.6-392
master_w_wtracechk_gen	wtracechk should be generated whenever wvalid is asserted	ARM IHI 0022f pg E2.6-392
master_w_wuserchk_gen	wuserchk should be generated whenever wvalid is asserted	ARM IHI 0022f pg E2.6-392
master_w_wpoisonchk_gen	wpoisonchk should be generated whenever wvalid is asserted	ARM IHI 0022f pg E2.6-392
master_wdata_recom_inactivebytelines	wdata is driven to zero for inactive byte lanes	ARM IHI0022f pg A3-42
master_aw_trsize_never_cross_max_bytes	Write transaction must not cross maximum transaction bytes	ARM IHI 0022f Section E1.18. pg E1-400
master_aw_trsize_never_cross_max_boundary	Write transaction must not cross maximum transaction boundary	ARM IHI 0022f Section E1.18. pg E1-400

1.4 Master Atomic Access

master_ar_excl_araddr_aligned	The address of exclusive access must be aligned to the total number of bytes in the transaction.	ARM IHI 0022f Section A7.2.4 A7.99
master_ar_excl_arlen_correct	Exclusive accesses are not permitted to use a burst length greater than 16.	ARM IHI 0022f Section A7.2.4 A7.99
master_ar_excl_arlen_arsize_correct_bytes	The number of bytes in an exclusive burst should be 1,2,4,8,16,32,64 or 128.	ARM IHI 0022f Section A7.2.4 A7.99
master_ar_excl_arcache_low	Exclusive access burst cannot be cacheable.	ARM IHI 0022f Section A7.2.4 A7-99
master_aw_excl_control_signals_match	The control signals for the read and write portions of the exclusive access must be identical.	ARM IHI 0022f Section A7.2.4 A7.99
master_ar_aw_excl_exwrite_after_exread	The master must not commence the write portion of the exclusive access until the read portion is complete.	ARM IHI 0022f Section A7.2.2 A7.98

AXI ABVIP User Guide
Appendix D: AXI5 ABVIP Package Contents--AXI5 Checks

master_aw_excl_awlen_correct	Exclusive accesses are not permitted to use a burst length greater than 16.	ARM IHI 0022f Section A7.2.4 A7.99
master_aw_excl_awlen_awsiz_correct_bytes	The number of bytes to be transferred in an exclusive access burst must be a power of 2, that is 1,2,4,8,16,32,64 or 128 bytes.	ARM IHI 0022f Section A7.2.4 A7.99
master_aw_awmmussid_valid	For transactions which do not specify a sub-stream identifier, AXMMUSSID must be driven to 0	ARM IHI 0022f Section E2.12. pg E2-371
master_aw_awmmuatst_asserted	Master must drive awmmusecsid and awmmussidv low for transactions already undergone translation, as indicated by awmmuatst asserted	ARM IHI 0022f Section E2.12. pg E2-371
master_aw_awmmusecsid_deasserted	Master must drive awprot[1] high indicating non-sec transaction, when awmmusecsid is deasserted	ARM IHI 0022f Section E2.12. pg E2-371
master_aw_awatop_nonatomic	Master must drive awatop non-atomic if atomic trn is not supported	ARM IHI 0022f section E2.1.6 E2-339
master_aw_awatop_valid	Master must drive valid values of awatop signal as specified by table section 3.6	ARM IHI 0022f section E2.1.6 E2-339
master_aw_atomic_trn_data_size	Master must drive data value size 1,2,4,8 bytes for Atomic Store, Atomic Load, Atomic Swap transactions	ARM IHI 0022f section E2.1.3 E2-337
master_aw_atomic_compare_data_size	Master must drive data value size 2,4,16,32,64 bytes for Atomic Compare transactions	ARM IHI 0022f section E2.1.3 E2-337
master_aw_atomic_trn_awsiz_full	For any atomic transaction, with awlen > 0, awsiz is required to be full data bus width	ARM IHI 0022f section E2.1.3 E2-337
master_aw_atomic_trn_awid_in_use	Atomic transaction must not use id that is used by non-atomic transactions. Multiple atomic transactions at same time should not use same id	ARM IHI 0022f section E2.1.4 E2-339
master_aw_atomic_awburst_incr	For all AtomicStore, AtomicLoad and AtomicSwap transactions, the burst type must be INCR.	ARM IHI 0022f section E2.1.3 E2-338
master_aw_atomic_awaddr_aligned	Master must drive aligned address for AtomicLoad, AtomicStore and AtomicSwap transactions.	ARM IHI 0022f section E2.1.3 E2-337
master_aw_atomic_awlock	For Atomic transactions, awlock must be 0.	ARM IHI 0022f section E2.1.5 E2-339
master_aw_cas_addr_aligned_burst_wrap	Master must drive wrap burst if address is not aligned with respect to total size of transfer for AtomicCompare.	ARM IHI 0022f section E2.1.3 E2-338
master_aw_awlock_no_excl_access_throttle_cnstr	When EXCL_ACCESS_ON is OFF, disable exclusive transactions for write address channel.	None
master_ar_arlock_no_excl_access_throttle_cnstr	When EXCL_ACCESS_ON is OFF, disable exclusive transactions for read address channel.	None
master_aw_non_atomic_trn_awid_in_use	Atomic and non atomic transaction can use same axi id, provided one transaction has fully completed before the other is issued	ARM IHI 0022f section E2.1.4 E2-339

AXI ABVIP User Guide
Appendix D: AXI5 ABVIP Package Contents--AXI5 Checks

master_ar_arid_atomic_in_use	Atomic and the non atomic transaction can use the same axi id, provided one transaction has fully completed before the other is issued.	ARM IHI 0022f section E2.1.4 E2-339
master_aw_ar_awid_atomic_arid_same_cycle	Atomic transaction and read transaction should not be launched at same cycle with same id	ARM IHI 0022f section E2.1.4 E2-339
master_aw_cas_awaddr_aligned	The address of an AtomicCompare must be aligned to a single outbound data value, that is, half the total outbound data size	ARM IHI 0022f section E2.1.3 E2-337
master_aw_cas_addr_unaligned_burst_wrap	Master must drive wrap burst if the address is not aligned wrt total size of transfer for AtomicCompare	ARM IHI 0022f section E2.1.3 E2-338
master_w_wdata_invalid_lane_0	If Data Check signaling is supported it is required that all invalid data lanes are driven to zero.	ARM IHI 0022f section E2.5.3 E2-353
master_ar_excl_req_with_wr_ctl_done	The master should not send an exclusive read req when there is already excl write req occurred with same ID	ARM IHI 0022f Section A7.2.4 A7.99
1.5 Master Write Data Before Control (DBC)		
master_aw_w_dbc_awlen_exact_len	If data phase finishes before control, the subsequent control phase must match that data transfer length.	None
master_w_aw_wlast_exact_len_dbc	wlast should be asserted in the last beat of a write transaction.	ARM IHI 0022f Section A3.2.2 A3.43
1.6 Master Reset		
master_rst_arvalid_low_reset	Master should drive arvalid low during reset.	None
master_rst_arvalid_low_reset_released	Master should not drive arvalid at a posedge of ackl after aresetn is high.	None
master_rst_awvalid_low_reset	Master should drive awvalid low during reset.	None
master_rst_awvalid_low_reset_released	Master should not drive awvalid at a posedge of ackl after aresetn is high.	None
master_rst_wvalid_low_reset	Master should drive wvalid low during reset.	None
master_rst_wvalid_low_reset_released	Master should not drive wvalid at a posedge of ackl after aresetn is high.	None
1.7 Master X-Checks		
master_xcheck_arvalid	When reset is high, arvalid should not have X or Z value.	ARM IHI 0022f Section A3.1.2 A3-40
master_xcheck_arid	When arvalid is asserted, arid should not have X or Z value on any bit.	ARM IHI 0022f Section A3.2 o A3-41
master_xcheck_araddr	When arvalid is asserted, arid should not have X or Z value on any bit.	ARM IHI 0022f Section A3.2.2 A3-42,43
master_xcheck_arlen	When arvalid is asserted, arlen should not have X or Z value on any bit.	ARM IHI 0022f Section A3.2 o A3-41
master_xcheck_arsize	When arvalid is asserted, arsize should not have X or Z value on any bit.	ARM IHI 0022f Section A3.2 o A3-41

AXI ABVIP User Guide
Appendix D: AXI5 ABVIP Package Contents--AXI5 Checks

master_xcheck_arburst	When arvalid is asserted, arburst should not have X or Z value on any bit.	ARM IHI 0022f Section A3.2 o A3-41
master_xcheck_arlock	When arvalid is asserted, arlock should not have X or Z value on any bit.	ARM IHI 0022f Section A3.2 o A3-41
master_xcheck_arcache	When arvalid is asserted, arcache should not have X or Z value on any bit.	ARM IHI 0022f Section A3.2 o A3-41
master_xcheck_arqos	When arvalid is asserted, arqos should not have X or Z value on any bit.	ARM IHI 0022f Section A3.2 o A3-41
master_xcheck_arregion	When arvalid is asserted, arregion should not have X or Z value on any bit.	ARM IHI 0022f Section A3.2 o A3-41
master_xcheck_aruser	When arvalid is asserted, aruser should not have X or Z value on any bit.	ARM IHI 0022f Section A3.2.2 A3-41
master_xcheck_rready	When rvalid is asserted, rready should not have X or Z value.	ARM IHI 0022f Section A3.2 o A3-42,43
master_xcheck_awvalid	When reset is asserted, awvalid should not have X or Z value.	ARM IHI 0022f Section A3.1 o A3-40
master_xcheck_awid	When awvalid is asserted, awid should not have X or Z value on any bit.	ARM IHI 0022f Section A3.2 o A3-41
master_xcheck_awaddr	When awvalid is asserted, awaddr should not have X or Z value on any bit.	ARM IHI 0022f Section A3.2 o A3-41
master_xcheck_awlen	When awvalid is asserted, awlen should not have X or Z value on any bit.	ARM IHI 0022f Section A3.2 o A3-41
master_xcheck_awsiz	When awvalid is asserted, awsiz should not have X or Z value on any bit.	ARM IHI 0022f Section A3.2 o A3-41
master_xcheck_awburst	When awvalid is asserted, awburst should not have X or Z value on any bit.	ARM IHI 0022f Section A3.2 o A3-41
master_xcheck_awlock	When awvalid is asserted, awlock should not have X or Z value on any bit.	ARM IHI 0022f Section A3.2 o A3-41
master_xcheck_awprot	When awvalid is asserted, awprot should not have X or Z value on any bit.	ARM IHI 0022f Section A3.2 o A3-41
master_xcheck_awcache	When awvalid is asserted, awcache should not have X or Z value on any bit.	ARM IHI 0022f Section A3.2 o A3-41
master_xcheck_awqos	When awqos is asserted, awcache should not have X or Z value on any bit.	ARM IHI 0022f Section A3.2 o A3-41

AXI ABVIP User Guide
Appendix D: AXI5 ABVIP Package Contents--AXI5 Checks

master_xcheck_awregion	When awregion is asserted, awcache should not have X or Z value on any bit.	ARM IHI 0022f Section A3.2 o A3-41
master_xcheck_awuser	When awuser is asserted, awcache should not have X or Z value on any bit.	ARM IHI 0022f Section A3.2 o A3-41
master_xcheck_wvalid	When reset is asserted, wvalid should not have X or Z value.	ARM IHI 0022f Section A3.1 o A3-40
master_xcheck_wdata	When wvalid is asserted, wdata should not have X or Z value on any bit.	ARM IHI 0022f Section A3.2 o A3-41
master_xcheck_wstrb	When wvalid is asserted, wstrb should not have X or Z value on any bit.	ARM IHI 0022f Section A3.2 o A3-41
master_xcheck_wlast	When wvalid is asserted, wlast should not have X or Z value.	ARM IHI 0022f Section A3.2 o A3-41
master_xcheck_bready	When bvalid is asserted, bready should not have X or Z value.	ARM IHI 0022f Section A3.2 o A3-41
master_xcheck_armmusecsid	When arvalid is asserted armmusecsid should not have X or Z value.	ARM IHI 0022f Section E2.12 E2-370
master_xcheck_armmusid	When arvalid is asserted armmusid should not have X or Z value.	ARM IHI 0022f Section E2.12 E2-370
master_xcheck_armmussidv	When arvalid is asserted armmussidv should not have X or Z value.	ARM IHI 0022f Section E2.12 E2-370
master_xcheck_armmussid	When arvalid is asserted armmussid should not have X or Z value.	ARM IHI 0022f Section E2.12 E2-370
master_xcheck_armmuatst	When arvalid is asserted armmuatst should not have X or Z value.	ARM IHI 0022f Section E2.12 E2-370
master_xcheck_artrace	When arvalid is asserted artrace should not have X or Z value.	ARM IHI 0022f Section E2.6 o E2-355
master_xcheck_arloop	When arvalid is asserted arloop should not have X or Z value.	ARM IHI 0022f Section E2.7 o E2-357
master_xcheck_awmmusecsid	When awvalid is asserted awmmusecsid should not have X or Z value.	ARM IHI 0022f Section E2.12 E2-370
master_xcheck_awmmusid	When awvalid is asserted awmmusid should not have X or Z value.	ARM IHI 0022f Section E2.12 E2-370
master_xcheck_awmmussidv	When awvalid is asserted awmmussidv should not have X or Z value.	ARM IHI 0022f Section E2.12 E2-370
master_xcheck_awmmussid	When awvalid is asserted awmmussid should not have X or Z value.	ARM IHI 0022f Section E2.12 E2-370

AXI ABVIP User Guide
Appendix D: AXI5 ABVIP Package Contents--AXI5 Checks

master_xcheck_awmmuatst	When <code>awvalid</code> is asserted <code>awmmuatst</code> should not have X or Z value.	ARM IHI 0022f Section E2.12 E2-370
master_xcheck_awatop	When <code>awvalid</code> is asserted <code>awatop</code> should not have X or Z value.	ARM IHI 0022f Section E2.12 E2-370
master_xcheck_wpoison	When <code>wvalid</code> is asserted <code>wpoison</code> should not have X or Z value.	ARM IHI 0022f Section E2.5.2 E2-352
master_xcheck_wdatachk	When <code>wvalid</code> is asserted <code>wdatachk</code> should not have X or Z value.	ARM IHI 0022f Section E2.5.1 E2-352
master_xcheck_awtrace	When <code>awvalid</code> is asserted <code>awtrace</code> should not have X or Z value.	ARM IHI 0022f Section E2.6 o E2-355
master_xcheck_wtrace	When <code>wvalid</code> is asserted <code>wtrace</code> should not have X or Z value.	ARM IHI 0022f Section E2.6 o E2-355
master_xcheck_awloop	When <code>awvalid</code> is asserted <code>awloop</code> should not have X or Z value.	ARM IHI 0022f Section E2.7 o E2-357
master_xcheck_armpam	<code>armpam</code> should not have X or Z value when <code>arvalid</code> is asserted	ARM IHI 0022f Section E1.17. pg 384
master_xcheck_aridunq	<code>aridunq</code> should not have X or Z value when <code>arvalid</code> is asserted	ARM IHI 0022f Section E1.17. pg 384
master_xcheck_archunken	<code>archunken</code> should not have X or Z value when <code>arvalid</code> is asserted	ARM IHI 0022f Section E1.14 376
master_xcheck_awmpam	<code>awmpam</code> should not have X or Z value when <code>awvalid</code> is asserted	ARM IHI 0022f Section E1.14 376
master_xcheck_avidunq	<code>avidunq</code> should not have X or Z value when <code>awvalid</code> is asserted	ARM IHI 0022f Section E1.17. pg 384
master_xcheck_araddrchk	<code>araddrchk</code> should not have X or Z value when <code>arvalid</code> is asserted	ARM IHI 0022f pg E2.4-389
master_xcheck_aridchk	<code>aridchk</code> should not have X or Z value when <code>arvalid</code> is asserted	ARM IHI 0022f pg E2.4-389
master_xcheck_arlenchk	<code>arlenchk</code> should not have X or Z value when <code>arvalid</code> is asserted	ARM IHI 0022f pg E2.4-389
master_xcheck_arctlchk0	<code>araddrchk0</code> should not have X or Z value when <code>arvalid</code> is asserted	ARM IHI 0022f pg E2.4-389
master_xcheck_arctlchk1	<code>arctlchk1</code> should not have X or Z value when <code>arvalid</code> is asserted	ARM IHI 0022f pg E2.4-389
master_xcheck_artracechk	<code>artracechk</code> should not have X or Z value when <code>arvalid</code> is asserted	ARM IHI 0022f pg E2.4-389
master_xcheck_arloopchk	<code>arloopchk</code> should not have X or Z value when <code>arvalid</code> is asserted	ARM IHI 0022f pg E2.4-389
master_xcheck_arnsaidchk	<code>arnsaidchk</code> should not have X or Z value when <code>arvalid</code> is asserted	ARM IHI 0022f pg E2.4-389

AXI ABVIP User Guide
Appendix D: AXI5 ABVIP Package Contents--AXI5 Checks

master_xcheck_arpamchk	arpamchk should not have X or Z value when arvalid is asserted	ARM IHI 0022(pg E2.4-389
master_xcheck_armmuchk	armmuchk should not have X or Z value when arvalid is asserted	ARM IHI 0022(pg E2.4-389
master_xcheck_armmusidchk	armmusidchk should not have X or Z value when arvalid is asserted	ARM IHI 0022(pg E2.4-389
master_xcheck_armmussidchk	armmussidchk should not have X or Z value when arvalid is asserted	ARM IHI 0022(pg E2.4-389
master_xcheck_aruserchk	aruserchk should not have X or Z value when arvalid is asserted	ARM IHI 0022(pg E2.4-389
master_xcheck_awaddrchk	awaddrchk should not have X or Z value when awvalid is asserted	ARM IHI 0022(pg E2.4-389
master_xcheck_awidchk	awidchk should not have X or Z value when awvalid is asserted	ARM IHI 0022(pg E2.4-389
master_xcheck_awlenchk	awlenchk should not have X or Z value when awvalid is asserted	ARM IHI 0022(pg E2.4-389
master_xcheck_awctlchk0	awctlchk0 should not have X or Z value when awvalid is asserted	ARM IHI 0022(pg E2.4-389
master_xcheck_awctlchk1	awctlchk1 should not have X or Z value when awvalid is asserted	ARM IHI 0022(pg E2.4-389
master_xcheck_awtracechk	awtracechk should not have X or Z value when awvalid is asserted	ARM IHI 0022(pg E2.4-389
master_xcheck_awloopchk	awloopchk should not have X or Z value when awvalid is asserted	ARM IHI 0022(pg E2.4-389
master_xcheck_awuserchk	awuserchk should not have X or Z value when awvalid is asserted	ARM IHI 0022(pg E2.4-389
master_xcheck_awnsaidchk	awnsaidchk should not have X or Z value when awvalid is asserted	ARM IHI 0022(pg E2.4-389
master_xcheck_awmmuchk	awmmuchk should not have X or Z value when awvalid is asserted	ARM IHI 0022(pg E2.4-389
master_xcheck_awmmusidchk	awmmusidchk should not have X or Z value when awvalid is asserted	ARM IHI 0022(pg E2.4-389
master_xcheck_awmmussidchk	awmmussidchk should not have X or Z value when awvalid is asserted	ARM IHI 0022(pg E2.4-389
master_xcheck_awmpamchk	awmpamchk should not have X or Z value when awvalid is asserted	ARM IHI 0022(pg E2.4-389
master_xcheck_wdatachk	wdatachk should not have X or Z value when wvalid is asserted	ARM IHI 0022(pg E2.4-389
master_xcheck_wlastchk	wlastchk should not have X or Z value when wvalid is asserted	ARM IHI 0022(pg E2.4-389
master_xcheck_wstrbchk	wstrbchk should not have X or Z value when wvalid is asserted	ARM IHI 0022(pg E2.4-389
master_xcheck_wpoisonchk	wpoisonchk should not have X or Z value when wvalid is asserted	ARM IHI 0022(pg E2.4-389
master_xcheck_wtracechk	wtracechk should not have X or Z value when wvalid is asserted	ARM IHI 0022(pg E2.4-389
master_xcheck_wuserchk	wuserchk should not have X or Z value when wvalid is asserted	ARM IHI 0022(pg E2.4-389

AXI ABVIP User Guide
Appendix D: AXI5 ABVIP Package Contents--AXI5 Checks

master_xchecks_armmuflow	armmuflow should not have X or Z value when arvalid is asserted	ARM IHI 0022f pg E2.4-389
master_xchecks_awmmuflow	awmmuflow should not have X or Z value when awvalid is asserted	ARM IHI 0022f pg E2.4-389
master_xcheck_artagop	artagop should not have X or Z value when arvalid is asserted	ARM IHI 0022f pg E2.4-389
master_xcheck_awtagop	awtagop should not have X or Z value when awvalid is asserted	ARM IHI 0022f pg E2.4-389
master_xchecks_wtag	wtag should not have X or Z value when wvalid is asserted	ARM IHI 0022f pg E2.4-389
master_xchecks_wtagchk	wtagchk should not have X or Z value when wvalid is asserted	ARM IHI 0022f pg E2.4-389
master_xchecks_wtagupdate	wtagupdate should not have X or Z value when wvalid is asserted	ARM IHI 0022f pg E2.4-389
1.8 Slave Stable		
slave_r_rvalid_stable	The RVALID signal must remain stable until RREADY goes high.	ARM IHI 0022f Section A3.2.1 A3-41
slave_r_rlast_stable	The RLAST signal must remain stable when RVALID is high until RREADY goes high.	ARM IHI 0022f Section A3.2.1 A3-41
slave_r_rdata_stable	The RDATA signal must remain stable when RVALID is high until RREADY goes high.	ARM IHI 0022f Section A3.2.1 A3-41
slave_r_rid_stable	The RID signal must remain stable when RVALID is high until RREADY goes high.	ARM IHI 0022f Section A3.2.1 A3-41
slave_r_rresp_stable	The RRESP signal must remain stable when RVALID is high until RREADY goes high.	ARM IHI 0022f Section A3.2.1 A3-41
slave_r_ruser_stable	The RUSER signal must remain stable when RVALID is high until RREADY goes high.	ARM IHI 0022f Section A3.2.1 A3-41
slave_b_bvalid_stable	The BVALID signal must remain stable until BREADY goes high.	ARM IHI 0022f Section A3.2.2 A3-43
slave_b_bid_stable	The BID signal must remain stable when BVALID is high until BREADY goes high.	ARM IHI 0022f Section A3.2.1 A3-41
slave_b_bresp_stable	The BRESP signal must remain stable when BVALID is high until BREADY goes high.	ARM IHI 0022f Section A3.2.2 A3-43
slave_b_buser_stable	The BUSER signal must remain stable when BVALID is high until BREADY goes high.	ARM IHI 0022f Section A3.2.1 A3-41
slave_rpoison_stable	The rpoison signal must remain stable when RVALID is high until RREADY goes high.	ARM IHI 0022f Section E2.5.2 E2-352
slave_rdatachk_stable	The rdatachk signal must remain stable when RVALID is high until RREADY goes high.	ARM IHI 0022f Section E2.5.2 E2-352

AXI ABVIP User Guide
Appendix D: AXI5 ABVIP Package Contents--AXI5 Checks

slave_rtrace_stable	The <code>rtrace</code> signal must remain stable when RVALID is high until RREADY goes high.	ARM IHI 0022f Section E2.6 o E2-355
slave_rloop_stable	The <code>rloop</code> signal must remain stable when RVALID is high until RREADY goes high.	ARM IHI 0022f Section E2.7 o E2-357
slave_btrace_stable	The <code>btrace</code> signal must remain stable when BALID is high until BREADY goes high.	ARM IHI 0022f Section E2.6 o E2-355
slave_bloop_stable	The <code>bloop</code> signal must remain stable when BALID is high until BREADY goes high.	ARM IHI 0022f Section E2.7 o E2-357
slave_vawqosaccept_stable	The <code>vawqosaccept</code> signal must remain stable when AWVALID is high until AWREADY goes high.	ARM IHI 0022f Section E2.7 o E2-358
slave_rchunkv_stable	<code>rchunkv</code> should remain stable if <code>rvalid</code> && <code>!rready</code>	ARM IHI 0022f pg E1.16-381
slave_rchunknum_stable	<code>rchunknum</code> should remain stable if <code>rvalid</code> && <code>!rready</code>	ARM IHI 0022f pg E1.16-381
slave_rchunkstrb_stable	<code>rchunkstrb</code> should remain stable if <code>rvalid</code> && <code>!rready</code>	ARM IHI 0022f pg E1.16-381
slave_ridunq_stable	<code>ridunq</code> should remain stable if <code>rvalid</code> && <code>!rready</code>	ARM IHI 0022f pg E1.16-381
slave_bidunq_stable	<code>bidunq</code> should remain stable if <code>bvalid</code> && <code>!bready</code>	ARM IHI 0022f pg E1.16-381
slave_rdatachk_stable	<code>rdatachk</code> should remain stable if <code>rvalid</code> && <code>!rready</code>	ARM IHI 0022f pg E2.4-389
slave_rrespchk_stable	<code>rrespchk</code> should remain stable if <code>rvalid</code> && <code>!rready</code>	ARM IHI 0022f pg E2.4-389
slave_rrlastchk_stable	<code>rlastchk</code> should remain stable if <code>rvalid</code> && <code>!rready</code>	ARM IHI 0022f pg E2.4-389
slave_rrloopchk_stable	<code>rloopchk</code> should remain stable if <code>rvalid</code> && <code>!rready</code>	ARM IHI 0022f pg E2.4-389
slave_rrpoisonchk_stable	<code>rpoisonchk</code> should remain stable if <code>rvalid</code> && <code>!rready</code>	ARM IHI 0022f pg E2.4-389
slave_rridchk_stable	<code>ridchk</code> should remain stable if <code>rvalid</code> && <code>!rready</code>	ARM IHI 0022f pg E2.4-389
slave_rrtracechk_stable	<code>rtracechk</code> should remain stable if <code>rvalid</code> && <code>!rready</code>	ARM IHI 0022f pg E2.4-389
slave_rrchunkchk_stable	<code>rchunkchk</code> should remain stable if <code>rvalid</code> && <code>!rready</code>	ARM IHI 0022f pg E2.4-389
slave_rruserchk_stable	<code>ruserchk</code> should remain stable if <code>rvalid</code> && <code>!rready</code>	ARM IHI 0022f pg E2.4-389
slave_brespchk_stable	<code>brespchk</code> should remain stable if <code>bvalid</code> && <code>!bready</code>	ARM IHI 0022f pg E2.4-389
slave_bbloopchk_stable	<code>bloopchk</code> should remain stable if <code>bvalid</code> && <code>!bready</code>	ARM IHI 0022f pg E2.4-389
slave_bbuserchk_stable	<code>buserchk</code> should remain stable if <code>bvalid</code> && <code>!bready</code>	ARM IHI 0022f pg E2.4-389

AXI ABVIP User Guide
Appendix D: AXI5 ABVIP Package Contents--AXI5 Checks

slave_b_btracechk_stable	btracechk should remain stable if bvalid && !bready	ARM IHI 0022f pg E2.4-389
slave_b_bidchk_stable	bidchk should remain stable if bvalid && !bready	ARM IHI 0022f pg E2.4-389
slave_b_btagmatch_stable	btagmatch should remain stable if bvalid && !bready	ARM IHI 0022f pg E2.4-389
slave_b_bcomp_stable	bcomp should remain stable if bvalid && !bready	ARM IHI 0022f pg E2.4-389
slave_r_rtag_stable	rtag should remain stable if rvalid && !rready	ARM IHI 0022f pg E2.4-389
slave_r_rtagchk_stable	rtagchk should remain stable if rvalid && !rready	ARM IHI 0022f pg E2.4-389
1.9 Slave Read		
slave_r_ar_rid_match	The Slaves are required to respond with an appropriate RID only after a valid req has been received from the master	ARM IHI 0022f Section A5.1 o A5-80
slave_r_ar_rid_no_interleave	In a sequence of read transactions with different ARID values, the slave can return read data that is out-of-order with respect to the order in which read requests are received. In case read data interleaving is not supported, RID should not change before end of read data.	ARM IHI 0022f Section A6.4 o A6-91
slave_r_ar_rid_in_order	The slave must return RRESP in the same order that the read request has been issued.	ARM IHI 0022f A6.4
slave_r_ar_rlast_exact_len	The slave must assert the rlast signal when it drives the final read transfer in the burst.	ARM IHI 0022f Section A3.2.2 A3.43
slave_ar_arready_wait_cycles	The maximum number of cycles for which ARREADY remains low after ARVALID goes high must not exceed the MAX_WAIT_CYCLES_AR cycles specified.	None
slave_r_resp_trace	The slave should respond with rtrace asserted if atrace was asserted in the read request.	ARM IHI 0022f Section E2.6 o E2-356
slave_r_rloop	The slave should respond with same rloop as in the read request.	ARM IHI 0022f Section E2.7 o E2-357
slave_b_bresp_trace	The slave should respond with btrace asserted if awtrace was asserted in the write request	ARM IHI 0022f Section E2.6 o E2-356
slave_b_bloop	The slave should respond with same bloop as in the write request	ARM IHI 0022f Section E2.7 o E2-357
slave_r_rvalidchk_gen	rvalidchk should be generated whenever aresetn is asserted	ARM IHI 0022f pg E2.6-392
slave_r_rreadychk_gen	rreadychk should be generated whenever aresetn is asserted	ARM IHI 0022f pg E2.6-392
slave_r_ridchk_gen	ridchk should be generated whenever rvalid is asserted	ARM IHI 0022f pg E2.6-392
slave_r_rloopchk_gen	rloopchk should be generated whenever rvalid is asserted	ARM IHI 0022f pg E2.6-392
slave_r_rtracechk_gen	rtracechk should be generated whenever rvalid is asserted	ARM IHI 0022f pg E2.6-392

AXI ABVIP User Guide
Appendix D: AXI5 ABVIP Package Contents--AXI5 Checks

slave_r_rpoisonchk_gen	rpoisonchk should be generated whenever rvalid is asserted	ARM IHI 0022f pg E2.6-392
slave_r_rlastchk_gen	rlastchk should be generated whenever rvalid is asserted	ARM IHI 0022f pg E2.6-392
slave_r_rdatachk_gen	rdatachk should be generated whenever rvalid is asserted	ARM IHI 0022f pg E2.6-392
slave_r_rrespchk_gen	rrespchk should be generated whenever rvalid is asserted	ARM IHI 0022f pg E2.6-392
slave_r_rchunkchk_gen	rchunkchk should be generated whenever rvalid is asserted	ARM IHI 0022f pg E2.6-392
slave_r_ruserchk_gen	ruserchk should be generated whenever rvalid is asserted	ARM IHI 0022f pg E2.6-392
slave_rdata_recom_inactivebytelines	rdata is driven to zero for inactive byte lanes	ARM IHI0022f pg A3-42
slave_r_ar_resp_consistent_decoder	DECERR is signaled for every beat of read data, or no beats of read data within each cache line of data	ARM IHI 0022f Section E1.18. pg E1.401

2.0 Slave Write

slave_b_aw_bid_match	The slaves are required to respond with an appropriate RID only after a valid req has been received from the master.	ARM IHI 0022f Section A5.1 o A5-80
slave_b_aw_bid_order_within_id	All transactions with a given ID must be ordered.	ARM IHI 0022f section A5.3.1 A5-82
slave_b_aw_bid_in_order	The slave must return BRESP in the same order that the write request has been issued.	ARM IHI 0022f section A6.2 or A6-89
slave_b_bvalidchk_gen	bvalidchk should be generated whenever aresetn is asserted	ARM IHI 0022f pg E2.6-392
slave_b_breadychk_gen	breadychk should be generated whenever aresetn is asserted	ARM IHI 0022f pg E2.6-392
slave_b_bidchk_gen	bidchk should be generated whenever bvalid is asserted	ARM IHI 0022f pg E2.6-392
slave_b_bpoisonchk_gen	bpoisonchk should be generated whenever bvalid is asserted	ARM IHI 0022f pg E2.6-392
slave_b_brespchk_gen	brespchk should be generated whenever bvalid is asserted	ARM IHI 0022f pg E2.6-392
slave_b_buserchk_gen	buserchk should be generated whenever bvalid is asserted	ARM IHI 0022f pg E2.6-392
slave_b_btracechk_gen	btracechk should be generated whenever bvalid is asserted	ARM IHI 0022f pg E2.6-392
slave_b_bloopchk_gen	bloopchk should be generated whenever bvalid is asserted	ARM IHI 0022f pg E2.6-392
slave_w_aw_wready_dbc	If data before control is allowed for slave, it should issue wready with or after awready when DATA_ACCEPT_WITH_OR_AFTER_CONTROL is high	None

2.1 Slave Atomic Access

slave_r_excl_resp_no_exokay	The slave must not send an EXOKAY response for a normal read transaction.	ARM IHI 0022f Section A7.2.2 A7.98
-----------------------------	---	------------------------------------

AXI ABVIP User Guide
Appendix D: AXI5 ABVIP Package Contents--AXI5 Checks

slave_b_excl_bresp_exokay	The slave must send an EXOKAY response when there is a valid exclusive write request	ARM IHI 0022f Section A7.2.2 A7.92
slave_b_excl_bresp_no_exokay	The slave must not send an EXOKAY response when there is not any valid exclusive write request	ARM IHI 0022f Section A7.2.2 A7.92
slave_b_excl_bresp_no_exokay_supported	The slave must not send an EXOKAY response for a normal write transaction	ARM IHI 0022f Section A7.2.3 A7.99
slave_b_aw_excl_resp_overlapping_addr	Exclusive Write fails when same location is updated since the exclusive read has happened	ARM IHI 0022f Section A7.2.4 A7.99
slave_r_excl_rresp_okay	The slave must not send an EXOKAY response when slave does not support exclusive access	ARM IHI 0022f Section A7.2.4 A7.99
slave_r_ar_excl_resp_okay	The slave should not send an OKAY and EXOKAY exclusive read resp in different beat of same transaction	ARM IHI 0022f Section A7.2.4 A7.99
slave_r_ar_excl_resp_exokay	The slave should not send an OKAY and EXOKAY exclusive read resp in different beat of same transaction	ARM IHI 0022f Section A7.2.4 A7.99

2.2 Slave Reset

slave_rst_rvalid_low_reset	Slave should drive rvalid low during reset.	None
slave_rst_bvalid_low_reset	Slave should drive bvalid low during reset.	None
slave_rst_rvalid_low_reset_released	Slave should not drive rvalid at a posedge of aclk after aresetn is high.	None
slave_rst_bvalid_low_reset_released	Slave should not drive bvalid at a posedge of aclk after aresetn is high.	None
slave_rst_varqosaccept	The default value of VARQOSACCEPT signal is 0	ARM IHI0022f Section E2.8 o E2-359
slave_rst_vawqosaccept	The default value of VARQOSACCEPT signal is 0	ARM IHI0022f Section E2.8 o E2-359

2.3 Slave X-Checks

slave_xcheck_arready	When arvalid is asserted, arready should not have X or Z value.	ARM IHI 0022f Section A3.2.2 A3-42,43
slave_xcheck_rvalid	When reset is high, rvalid should not have X or Z value.	ARM IHI 0022f Section A3.1.2 A3-40
slave_xcheck_rdata	When rvalid is asserted, rdata should not have X or Z value on any bit.	ARM IHI 0022f Section A3.2 o A3-41,42
slave_xcheck_rid	When rvalid is asserted, rid should not have X or Z value on any bit.	ARM IHI 0022f Section A3.2 o A3-41
slave_xcheck_rresp	When rvalid is asserted, rresp should not have X or Z value on any bit.	ARM IHI 0022f Section A3.2 o A3-41.42

AXI ABVIP User Guide
Appendix D: AXI5 ABVIP Package Contents--AXI5 Checks

slave_xcheck_ruser	When rvalid is asserted, ruser should not have X or Z value on any bit.	ARM IHI 0022f Section A3.2 o A3-41
slave_xcheck_rlast	When rvalid is asserted, rlast should not have X or Z value on any bit.	ARM IHI 0022f Section A3.2 o A3-41
slave_xcheck_awready	When avalid is asserted, awready should not have X or Z value.	ARM IHI 0022f Section A3.2 o A3-41
slave_xcheck_wready	When avalid is asserted, awready should not have X or Z value.	ARM IHI 0022f Section A3.2 o A3-41
slave_xcheck_bvalid	When reset is high, bvalid should not have X or Z value.	ARM IHI 0022f Section A3.2 o A3-41
slave_xcheck_bid	When bvalid is asserted, bid should not have X or Z value on any bit.	ARM IHI 0022f Section A3.2 o A3-41
slave_xcheck_bresp	When bvalid is asserted, bresp should not have X or Z value on any bit.	ARM IHI 0022f Section A3.2 o A3-41
slave_xcheck_buser	When bvalid is asserted, buser should not have X or Z value on any bit.	ARM IHI 0022f Section A3.2 o A3-41
slave_xcheck_rpoison	When rvalid is asserted rpoison should not have X or Z value.	ARM IHI 0022f Section E2.5.2 E2-352
slave_xcheck_rdatachk	When rvalid is asserted rdatachk should not have X or Z value.	ARM IHI 0022f Section E2.5.1 E2-352
slave_xcheck_vargosaccept	When arvalid is asserted vargosaccept should not have X or Z value.	ARM IHI 0022f Section E2.8 o E2-358
slave_xcheck_rtrace	When rvalid is asserted rtrace should not have X or Z value.	ARM IHI 0022f Section E2.6 o E2-355
slave_xcheck_rloop	When rvalid is asserted rloop should not have X or Z value.	ARM IHI 0022f Section E2.7 o E2-357
slave_xcheck_vawgosaccept	When avalid is asserted vawgosaccept should not have X or Z value	ARM IHI 0022f Section E2.8 o E2-358
slave_xcheck_btrace	When bvalid is asserted btrace should not have X or Z value	ARM IHI 0022f Section E2.6 o E2-355
slave_xcheck_bloop	When bvalid is asserted bloop should not have X or Z value	ARM IHI 0022f Section E2.7 o E2-357
slave_xcheck_ridunq	ridunq should not have X or Z value when rvalid is asserted	ARM IHI 0022f Section E1.17. pg 384
slave_xcheck_rchunkv	rchunkv should not have X or Z value when rvalid is asserted	ARM IHI 0022f Section E1.14 376

AXI ABVIP User Guide
Appendix D: AXI5 ABVIP Package Contents--AXI5 Checks

slave_xcheck_rchunknum	rchunknum should not have X or Z value when rvalid is asserted	ARM IHI 0022(Section E1.14 376
slave_xcheck_rchunkstrb	rchunkstrb should not have X or Z value when rvalid is asserted	ARM IHI 0022(Section E1.14 376
slave_xcheck_bidung	bidung should not have X or Z value when bvalid is asserted	ARM IHI 0022(Section E1.17. pg 384
slave_xcheck_ridchk	ridchk should not have X or Z value when rvalid is asserted	ARM IHI 0022(pg E2.4-389
slave_xcheck_rrespchk	rrespchk should not have X or Z value when rvalid is asserted	ARM IHI 0022(pg E2.4-389
slave_xcheck_rlastchk	rlastchk should not have X or Z value when rvalid is asserted	ARM IHI 0022(pg E2.4-389
slave_xcheck_rloopchk	rloopchk should not have X or Z value when rvalid is asserted	ARM IHI 0022(pg E2.4-389
slave_xcheck_rtracechk	rtracechk should not have X or Z value when rvalid is asserted	ARM IHI 0022(pg E2.4-389
slave_xcheck_ruserchk	ruserchk should not have X or Z value when rvalid is asserted	ARM IHI 0022(pg E2.4-389
slave_xcheck_rdatachk	rdatachk should not have X or Z value when rvalid is asserted	ARM IHI 0022(pg E2.4-389
slave_xcheck_rchunkchk	rchunkchk should not have X or Z value when rvalid is asserted	ARM IHI 0022(pg E2.4-389
slave_xcheck_rpoisonchk	rpoisonchk should not have X or Z value when rvalid is asserted	ARM IHI 0022(pg E2.4-389
slave_xcheck_bidchk	bidchk should not have X or Z value when bvalid is asserted	ARM IHI 0022(pg E2.4-389
slave_xcheck_brespchk	brespchk should not have X or Z value when bvalid is asserted	ARM IHI 0022(pg E2.4-389
slave_xcheck_bloopchk	bloopchk should not have X or Z value when bvalid is asserted	ARM IHI 0022(pg E2.4-389
slave_xcheck_buserchk	buserchk should not have X or Z value when bvalid is asserted	ARM IHI 0022(pg E2.4-389
slave_xcheck_btracechk	btracechk should not have X or Z value when bvalid is asserted	ARM IHI 0022(pg E2.4-389
slave_xcheck_bpoisonchk	bpoisonchk should not have X or Z value when bvalid is asserted	ARM IHI 0022(pg E2.4-389
slave_xcheck_btagmatch	btagmatch should not have X or Z value when bvalid is asserted	ARM IHI 0022(Section E1.15 389
slave_xcheck_bcomp	bcomp should not have X or Z value when bvalid is asserted	ARM IHI 0022(Section E1.15 389
slave_xcheck_rtag	rtag should not have X or Z value when rvalid is asserted	ARM IHI 0022(Section E1.15 389
slave_xcheck_rtagchk	rtagchk should not have X or Z value when rvalid is asserted	ARM IHI 0022(Section E1.15 389

AXI ABVIP User Guide
Appendix D: AXI5 ABVIP Package Contents--AXI5 Checks

2.4 Low Power Checks

assert_lpi_csysack_after_csysreq	csysack must follow CSYSREQ while entering or exiting the low power state.	ARM IHI 0022f Section A9.2 o A9-103
assert_lpi_cactive_after_csysreq	cactive changes must happen according to csysreq.	ARM IHI 0022f Section A9.2 o A9-103
assert_lpi_csysreq_remains_high_until_ack	csysreq once high must remain stable until it is acknowledged by csysack.	ARM IHI 0022f Section A9.2 o A9-103
assert_lpi_csysreq_remains_low_until_ack	csysreq once low must remain stable until it is acknowledged by csysack.	ARM IHI 0022f Section E2.9.1 E2-360
assert_xcheck_csysreq	When reset is high, csysreq should not have X or Z value.	ARM IHI 0022f Section A3.2.2 A3-38,39
assert_xcheck_csysack	When reset is high, csysack should not have X or Z value.	ARM IHI 0022f Section A3.2.2 A3-38,39
assert_xcheck_cactive	When reset is high, cactive should not have X or Z value.	ARM IHI 0022f Section A3.2.2 A3-38,39
master_lpi_awakeup_cycle_before_awvalid	It is recommended that awakeup is asserted a cycle before assertion of valid	ARM IHI 0022f Section E2.9.1 E2-360
master_lpi_awakeup_cycle_before_arvalid	It is recommended that awakeup is asserted a cycle before assertion of valid	ARM IHI 0022f Section E2.9.1 E2-360
master_lpi_awakeup_cycle_before_wvalid	It is recommended that awakeup is asserted a cycle before assertion of valid	ARM IHI 0022f Section E2.9.1 E2-360
master_lpi_awakeup_awvalid_stable	awakeup must remain asserted until associated valid/ready handshake	ARM IHI 0022f Section E2.9.1 E2-360
master_lpi_awakeup_arvalid_stable	awakeup must remain asserted until associated valid/ready handshake	ARM IHI 0022f Section E2.9.1 E2-360
master_lpi_awakeup_wvalid_stable	awakeup must remain asserted until associated valid/ready handshake	ARM IHI 0022f Section E2.9.1 E2-360
master_lpi_awakeup_wr_active	After the AWVALID, AWREADY handshake, the interconnect must remain active until the transaction has completed	ARM IHI 022F: Section E2.9.1 E2-360
master_lpi_awakeup_rd_active	After the ARVALID, ARREADY handshake, the interconnect must remain active until the transaction has completed	ARM IHI 022F: Section E2.9.1 E2-360
master_lpi_awakeup_arvalid_deassert	awakeup signal should be deasserted when no further transactions are required	ARM IHI 0022f Section E2.9.1 E2-360

2.5 Internal Table Checks

assert_ar_rd_tbl_no_overflow	This assertion checks that the table never overflows.	None
assert_aw_wr_tbl_no_overflow	This assertion checks that the table never overflows.	None

AXI ABVIP User Guide
Appendix D: AXI5 ABVIP Package Contents--AXI5 Checks

assert_ar_excl_tbl_no_overflow	This assertion checks that the exclusive table never overflows.	None
master_ar_rd_tbl_no_overflow	Master should not issue <code>arvalid</code> when read table is full	None
slave_ar_rd_tbl_no_overflow	Slave should not issue <code>arready</code> when read table is full	None
master_aw_wr_tbl_no_overflow	Master should not issue <code>awvalid</code> when write table is full	None
slave_aw_wr_tbl_no_overflow	Slave should not issue <code>awready</code> when write table is full	None
master_w_wr_tbl_no_overflow	Master should not issue <code>wvalid</code> when write table is full and no write data is pending	None
slave_w_wr_tbl_no_overflow	Slave should not issue <code>wready</code> when write table is full and no write data is pending	None
master_ar_arlock_excl_tbl_no_overflow	Master should not issue <code>arlock</code> when exclusive access table is full	None
slave_ar_arready_excl_tbl_no_overflow	Slave should not accept exclusive access transaction when exclusive access table is full	None
2.6 Deadlock Checks		
master_aw_awvalid_eventually	All pending write control requests must start eventually	None
master_aw_awvalid_wait_cycles	All pending write control requests must start within <code>MAX_WAIT_CYCLES_W_AW</code> cycles	None
slave_aw_awready_eventually	Slave must assert <code>awready</code> eventually if there are pending write requests	None
master_w_wvalid_eventually	All pending write data must start eventually	None
master_aw_wvalid_wait_cycles	All pending write data requests must start within <code>MAX_WAIT_CYCLES_AW_W</code> cycles	None
slave_w_wready_eventually	Slave must assert <code>wready</code> eventually if there are pending write data	None
slave_aw_wready_wait_cycles	All write data must be accepted within <code>MAX_WAIT_CYCLES_AW_W</code> cycles.	None
slave_b_bvalid_eventually	All pending write response must start eventually	None
slave_a_bvalid_wait_cycles	All pending responses must be given by slave within <code>MAX_WAIT_CYCLES_AW_B</code> cycles	None
master_b_bready_eventually	Master must assert <code>bready</code> if there are pending write response	None
master_b_bready_wait_cycles	Master must assert <code>bready</code> within <code>MAX_WAIT_CYCLES_AW_B</code> cycles if there are pending write response	None
slave_aw_awvalid_awready_eventually	If <code>awvalid</code> is asserted when <code>awready</code> is low, slave must assert <code>awready</code> eventually	None
slave_w_wvalid_wready_eventually	If <code>wvalid</code> is asserted when <code>wready</code> is low, slave must assert <code>wready</code> eventually	None
master_b_bvalid_bready_eventually	If <code>bvalid</code> is asserted when <code>bready</code> is low, master must assert <code>bready</code> eventually	None
slave_aw_awready_wait_cycles	The maximum number of cycles for which <code>awready</code> remains low after <code>awvalid</code> going high must not exceed the <code>MAX_WAIT_CYCLES_AW</code> specified	None
slave_w_wready_wait_cycles	The maximum number of cycles for which <code>wready</code> remains low after <code>wvalid</code> going high must not exceed the <code>MAX_WAIT_CYCLES_W</code> specified	None

AXI ABVIP User Guide
Appendix D: AXI5 ABVIP Package Contents--AXI5 Checks

master_b_bready_wait_cycles	The maximum number of cycles for which bready remains low after bvalid going high must not exceed the MAX_WAIT_CYCLES_B specified	None
slave_r_rvalid_eventually	All pending read responses must start eventually	None
slave_ar_rvalid_wait_cycles	All pending read responses must start within MAX_WAIT_CYCLES_AR_R cycles.	None
master_r_rready_eventually	Master must assert ready if there are pending read response	None
slave_ar_arvalid_arready_eventually	If arvalid is asserted when arready is low, slave must assert arready eventually	None
master_r_rvalid_rready_eventually	If rvalid is asserted when rready is low, master must assert rready eventually	None
slave_atomic_r_rvalid_eventually	All pending read responses must start eventually for atomic transaction	None
master_atomic_r_rready_eventually	Master must assert rready if there are pending read response for atomic transaction	None
slave_ar_arready_wait_cycles	The maximum number of cycles for which ARREADY remains low after ARVALID going high must not exceed the MAX_WAIT_CYCLES_AR cycles specified	None
master_r_rready_wait_cycles	The maximum number of cycles for which RREADY remains low after RVALID going high must not exceed the MAX_WAIT_CYCLES_R cycles specified	None
master_ar_rready_wait_cycles	All read data must be accepted within MAX_WAIT_CYCLES_AR_R cycles.	None

2.7 Read Data Chunking Checks

master_ar_chunken_size_arlen	ARCHUNKEN must only be asserted for transactions with ARSIZE is equal to the data bus width or ARLEN is one beat.	ARM IHI0022C section E1.14.
master_ar_chunken_size	ARCHUNKEN must only be asserted for transactions with ARSIZE is equal or greater than 128 bits.	ARM IHI0022C section E1.14.
master_ar_chunken_araddr	ARCHUNKEN must only be asserted for transactions with ARADDR is aligned to 16 bytes.	ARM IHI0022C section E1.14.
master_ar_chunken_arburst	ARCHUNKEN must only be asserted for transactions with ARBURST is INCR or WRAP	ARM IHI0022C section E1.14.
master_no_rd_req_with_same_arid	The master must not issue a request on the read channel with the same ARID as an outstanding request that had ARCHUNKEN asserted	ARM IHI0022C section E1.14.
master_chunkid_no_outstanding_rd_trans	ARCHUNKEN can only be asserted if there are no outstanding read transactions using the same ARID value	ARM IHI0022C section E1.14.
master_ar_chunken_aridunq	If present on the interface, ARIDUNQ must be asserted	ARM IHI0022C section E1.14.
slave_r_rchunkv_atomic	RCHUNKV cannot be asserted for ARCHUNKEN, Atomic transaction are sent on write channel.	ARM IHI0022C section E1.14.
slave_r_archunken_rchunkv_match	If ARCHUNKEN is asserted, the corresponding RCHUNKV can be asserted response beats of the transaction.	ARM IHI0022C section E1.14.
slave_r_rchunkv_same_all_beat	RCHUNKV must be the same for every response beat of a transaction.	ARM IHI0022C section E1.14.
slave_r_ar_chunken_rchunknum_limit	When RVALID and RCHUNKV are asserted, RCHUNKNUM must be between zero and ARLEN.	ARM IHI0022C section E1.14.
slave_r_ar_chunken_rchunkstrb	When RVALID and RCHUNKV are asserted, RCHUNKSTRB must not be zero.	ARM IHI0022C section E1.14.

AXI ABVIP User Guide
Appendix D: AXI5 ABVIP Package Contents--AXI5 Checks

slave_r_ar_chunken_deassert	When RVALID is asserted and RCHUNKV is de-asserted, RCHUNKNUM and RCHUNKSTRB must be zero.	ARM IHI0022G section E1.14.
slave_r_ar_chunkstrb_match	The strobes of data chunk transferred must be unique for every chunk number	ARM IHI0022G section E1.14.
slave_r_ar_rlast_chunklen	The slave must assert the rlast signal when it drives the final read transfer in the burst	ARM IHI0022G section E1.14.
2.7 Regular Transaction Checks		
master_regular_arlen_arburst	For regular transactions- ARLEN is 1, 2, 4, 8 or 16. ARBURST is INCR or WRAP, not fixed.	ARM IHI0022H section A3.4.3
master_regular_arsize	For regular transactions- ARSIZE is the same as data bus width, if ARLEN is greater than 1.	ARM IHI0022H section A3.4.3
master_regular_araddr_incr_aligned	For regular transactions- ARADDR is aligned to the transaction container for INCR transactions.	ARM IHI0022H section A3.4.3
master_regular_araddr_wrap_aligned	For regular transactions- ARADDR is aligned to ARSIZE for WRAP transactions.	ARM IHI0022H section A3.4.3
master_regular_awlen_awburst	For regular transactions- AWLEN is 1, 2, 4, 8 or 16. AWBURST is INCR or WRAP, not fixed.	ARM IHI0022H section A3.4.3
master_regular_awsiz	For regular transactions- AWSIZE is the same as data bus width, if ARLEN is greater than 1.	ARM IHI0022H section A3.4.3
master_regular_awaddr_incr_aligned	For regular transactions- AWADDR is aligned to the transaction container for INCR transactions.	ARM IHI0022H section A3.4.3
master_regular_awaddr_wrap_aligned	For regular transactions- AWADDR is aligned to AWSIZE for WRAP transactions.	ARM IHI0022H section A3.4.3
2.8 Memory Tagging Checks		
master_ar_artagop_arburst	AxBURST must be INCR or WRAP, not FIXED	ARM IHI 0022H Section E1.15. pg E1-389
master_aw_awtagop_arburst	AxBURST must be INCR or WRAP, not FIXED	ARM IHI 0022H Section E1.15. pg E1-389
master_ar_artagop_arcache	The transaction must be to Normal Write-Back memory	ARM IHI 0022H Section E1.15. pg E1-389
master_aw_awtagop_arcache	The transaction must be to Normal Write-Back memory	ARM IHI 0022H Section E1.15. pg E1-389
master_ar_artagop_aridunq	ARIDUNQ must be asserted for a read with tag Transfer or Fetch	ARM IHI 0022H Section E1.15. pg E1-389
master_aw_awtagop_awidunq	AWIDUNQ must be asserted for a write with tag Transfer or Fetch	ARM IHI 0022H Section E1.15. pg E1-389
master_ar_artagop_no_matching_id	A read with tag Transfer or Fetch can only be issued if there are no outstanding read transactions using the same ARID value	ARM IHI 0022H Section E1.15. pg E1-389
master_aw_awtagop_no_matching_id	A write with tag Transfer or Fetch can only be issued if there are no outstanding write transactions using the same AWID value	ARM IHI 0022H Section E1.15. pg E1-389

AXI ABVIP User Guide
Appendix D: AXI5 ABVIP Package Contents--AXI5 Checks

master_ar_read_no_matching_artagop_id	Master must not issue a request on the read channel with the same ARID as an outstanding read with tag Transfer or Fetch	ARM IHI 0022f Section E1.15. pg E1-389
master_aw_write_no_matching_awtagop_id	Master must not issue a request on the write channel with the same AWID as an outstanding Write with tag Transfer or Fetch	ARM IHI 0022f Section E1.15. pg E1-389
slave_r_rtag_strb	For read transactions that use strobes, only tags which correspond to valid strobes are required to be valid;	ARM IHI 0022f Section E1.15. pg E1-390
slave_r_rtag_invalid_op	For read transactions that has invalid tag operation, rtags RTAG is invalid and must be zero	ARM IHI 0022 Section E1.15. pg E1-390
master_ar_artagop_legal_val	Read request tag operation, encoded as:0 Invalid, 1 Transfer, 2 Reserved, 3 Fetch	ARM IHI 0022f Section E1.15. pg E1-388
master_w_wtag_invalid_op	For write transactions invalid tag operation, WTAG is invalid and must be zero and WTAGUPDATE must be de-asserted	ARM IHI 0022f Section E1.15. pg E1-392
master_w_wtag_invalid_op_curr	For write transactions invalid tag operation, WTAG is invalid and must be zero and WTAGUPDATE must be de-asserted	ARM IHI 0022f Section E1.15. pg E1-392
master_w_wtag_invalid_op_dbc	For write transactions invalid tag operation, WTAG is invalid and must be zero and WTAGUPDATE must be de-asserted	ARM IHI 0022f Section E1.15. pg E1-392
master_w_wtagupdate_transfer_match	For write transactions transfer tag operation, WTAG bits must be valid for every byte in the transaction container, WTAGUPDATE must be de-asserted	ARM IHI 0022f Section E1.15. pg E1-392
master_w_wtagupdate_transfer_match_curr	For write transactions transfer tag operation, WTAG bits must be valid for every byte in the transaction container, WTAGUPDATE must be de-asserted	ARM IHI 0022f Section E1.15. pg E1-392
master_w_wtagupdate_transfer_match_dbc	For write transactions transfer tag operation, WTAG bits must be valid for every byte in the transaction container, WTAGUPDATE must be de-asserted	ARM IHI 0022f Section E1.15. pg E1-392
master_wtag_wtagupdate_same_for_less_data_width	multiple beats address the same tag, WTAG and WTAGUPDATE values must be consistent for each 4-bit tag that is accessed by the transaction	ARM IHI 0022f Section E1.15. pg E1-390
master_awtag_tranfer_update_no_awtop	Atomic transactions cannot be used with Transfer or Update operations	ARM IHI 0022f Section E1.15. pg E1-394
master_awtag_awtop_compare_wtag_same_cmp_swp_bytes	For AtomicCompare transactions of 32 bytes, the same tag value must be used for tag bits associated with the compare and swap bytes	ARM IHI 0022f Section E1.15. pg E1-394
master_awtag_awtop_compare_wtag_same_cmp_swp_bytes_dbc	AtomicCompare transactions of 32 bytes, the same tag value must be used for tag bits associated with the compare and swap bytes	ARM IHI 0022f Section E1.15. pg E1-394
master_awtag_awtop_compare_wtag_same_cmp_swp_bytes_curr	AtomicCompare transactions of 32 bytes, the same tag value must be used for tag bits associated with the compare and swap bytes	ARM IHI 0022f Section E1.15. pg E1-394
master_awtag_match_awtop_compare_size	AtomicCompare transactions with Match can be 16 bytes or 32 bytes	ARM IHI 0022f Section E1.15. pg E1-394
slave_r_rtag_atomic_resp	Read data that is returned within an Atomic Transaction does not have valid RTAG values, so RTAG is recommended to be zero	ARM IHI 0022f Section E1.15. pg E1-394

AXI ABVIP User Guide
Appendix D: AXI5 ABVIP Package Contents--AXI5 Checks

master_w_wtagupdate_update_op	Tags that are only partially addressed in the transaction must have WTAGUPDATE deasserted. WriteUniqueFull and WriteFullCMO with Update must have all associated WTAGUPDATE bits asserted	ARM IHI 0022f Section E1.15. pg E1-392
slave_b_aw_bvalid_for_match_tagop	Slaves are required to send bcomp or btagmatch with each bvalid	ARM IHI 0022f Section E1.15. pg E1-393
slave_b_aw_bvalid_btagmatch_part2_rep	BID must have the same value as AWID	ARM IHI 0022f Section E1.15. pg E1-393
slave_b_aw_bcomp_btagmatch_valid_value_tagop	Valid combinations of BCOMP and BTAGMATCH	ARM IHI 0022f Section E1.15. pg E1-394
slave_b_aw_bcomp_btagmatch_valid_value_no_tagop	Valid combinations of BCOMP and BTAGMATCH	ARM IHI 0022f Section E1.15. pg E1-394
slave_b_aw_btagmatch_bresp_no_exokay	BRESP can be OKAY, SLVERR, or DECERR	ARM IHI 0022f Section E1.15. pg E1-393
slave_b_aw_bcomp_bresp_no_transfault	BRESP can be OKAY, SLVERR, EXOAKY or DECERR;	ARM IHI 0022f Section E1.15. pg E1-393
slave_b_aw_btagmatch_eventually	All pending btagmatch write responses must start eventually	ARM IHI 0022f Section E1.5
master_b_bready_btagmatch_eventually	All btagmatch must be eventually accepted	ARM IHI 0022f Section E1.5
master_aw_awtagop_legal_val	Write request tag operation, encoded as: 0 Invalid, 1 Transfer, 2 Update, 3 Match	ARM IHI 0022f Section E1.15. pg E1-388
2.9 UNTRANSLATED TRANSACTION VERSION 2 CHECKS		
master_ar_armmuatst_zero_v2	this signal is not present for Untranslated Transactions v2. If not present the default value is 0	ARM IHI 0022f Section E1.9.1 E1-370
master_ar_atst_flow_mmusecid_low	When the flow is ATST, AxMMUSECSID must be LOW	ARM IHI 0022f Section E2.12. pg E2-371
master_ar_atst_flow_mmuusidv_low	When the flow is ATST, AxMMUSSIDV must be LOW	ARM IHI 0022f Section E2.12. pg E2-371
slave_r_ar_resp_consistent_transfault	If TRANSFAULT is used for one response beat, it must be used for all response beats of a transaction;	ARM IHI 0022f Section E2.12. pg E2-371
slave_r_resp_transfault_check	If ARMMUFLOW is not PRI, RRESP must not be TRANSFAULT	ARM IHI 0022f Section E1.9.6 E1-374
slave_r_resp_legal_val	RRESP can be normal response or transfault	ARM IHI 0022f Section E1.9.1 E1-370
master_aw_awmmuatst_zero_v2	this signal is not present for Untranslated Transactions v2. If not present the default value is 0;	ARM IHI 0022f Section E1.9.1 E1-370

master_aw_atst_flow_mmusecid_low	When the flow is ATST,AxMMUSECSID must be LOW	ARM IHI 0022f Section E2.12. pg E2-371
master_aw_atst_flow_mmussidv_low	When the flow is ATST,AxMMUSSIDV must be LOW	ARM IHI 0022f Section E2.12. pg E2-371
slave_b_resp_transfault_check	If AWMMUFLOW is not PRI, BRESP must not be TRANSFAULT	ARM IHI 0022f Section E1.9.6 E1-374
slave_b_resp_legal_val	BRESP can be normal response or transfault	ARM IHI 0022f Section E1.9.1 E1-371
master_aw_awmmuflow_zero_v1	this signal is not present for Untranslated Transactions v1. If not present the default value is 0	ARM IHI 0022f Section E1.9.1 E1-371
master_ar_armmuflow_zero_v1	this signal is not present for Untranslated Transactions v1. If not present the default value is 0	ARM IHI 0022f Section E1.9.1 E1-371

Coverage Checks

The monitor provides coverage checks to collect coverage information for possible scenarios in an AXI4-based design. The coverage information is required for complete verification of the DUV.

Table B-7 describes the SVA coverage checks for the AXI4 monitor.

Table B-7 AXI Monitor Coverage Points

Coverage Assertions	Description
1.1 TRANSITION ID COVERS	
cover_arid_0	arid should take id 0
cover_arid_1	arid should take id 1
cover_arid_other	arid should take all other possible values
cover_awid_0	awid should take id 0
cover_awid_1	awid should take id 1
cover_awid_other	awid should take all other possible values
1.2 TRANSACTION LENGTH COVERS	
cover_arlen_len1	arlen, awlen can be upto 256
cover_arlen_len2	arlen, awlen can be upto 256
cover_arlen_len3	arlen, awlen can be upto 256
cover_arlen_len4	arlen, awlen can be upto 256
cover_arlen_short	arlen, awlen can be upto 256
cover_arlen_long	arlen, awlen can be upto 256
cover_arlen_max	arlen, awlen can be upto 256
cover_awlen_len1	arlen, awlen can be upto 256
cover_awlen_len2	arlen, awlen can be upto 256

AXI ABVIP User Guide
Appendix D: AXI5 ABVIP Package Contents--Coverage Checks

cover_awlen_len3	arlen, awlen can be upto 256
cover_awlen_len4	arlen, awlen can be upto 256
cover_awlen_short	arlen, awlen can be upto 256
cover_awlen_long	arlen, awlen can be upto 256
cover_awlen_max	arlen, awlen can be upto 256
1.3 TRANSACTION BURST COVERS	
cover_arburst_fixed	arburst can be of fixed type
cover_arburst_incr	arburst can be of increment type
cover_arburst_wrap	arburst can be of wrap type
cover_awburst_fixed	awburst can be of fixed type
cover_awburst_incr	awburst can be of increment type
cover_awburst_wrap	awburst can be of wrap type
cover_arburst_wrap_len_2	If read is of wrap type, the arlen can be 1
cover_arburst_wrap_len_4	If read is of wrap type, the arlen can be 3
cover_arburst_wrap_len_8	If read is of wrap type, the arlen can be 7
cover_arburst_wrap_len_16	If read is of wrap type, the arlen can be 15
cover_awburst_wrap_len_2	If write is of wrap type, the awlen can be 1
cover_awburst_wrap_len_4	If write is of wrap type, the awlen can be 3
cover_awburst_wrap_len_8	If write is of wrap type, the awlen can be 7
cover_awburst_wrap_len_16	If write is of wrap type, the awlen can be 15
cover_arburst_fixed_araddr_unaligned	Unaligned read address during fixed burst
cover_arburst_incr_araddr_unaligned	Unaligned read address during incrementing burst
cover_awburst_fixed_awaddr_unaligned	Unaligned write address during fixed burst
cover_awburst_incr_awaddr_unaligned	Unaligned write address during incrementing burst
1.4 SLAVE RESPONSE SIGNALLING COVERS	
cover_rresp_okay	rresp can have okay response
cover_rresp_slverr	rresp can have slave error response
cover_rresp_decerr	rresp can have decode error response
cover_bresp_okay	bresp can have okay response

cover_bresp_slvrr	bresp can have slave error response
cover_bresp_decerr	bresp can have decode error response
1.5 SEARCHPOINT COVERS	
cover_searchpoint_rd_control	Cover for read control
cover_searchpoint_rd_data	Cover for read data
cover_searchpoint_rd_data_last	Cover for read last data
cover_searchpoint_wr_control	Cover for write control
cover_searchpoint_wr_data	Cover for write data
cover_searchpoint_wr_data_last	Cover for write last data
cover_searchpoint_wr_response	Cover for write response
1.6 READ/WRITE ORDER COVER	
cover_order_control_before_data	Control can come before data
cover_order_control_with_data	Control can come with data
cover_order_control_during_data	Control can come during data
cover_order_control_after_data	Control can come after data
cover_order_control_dbc_ongoing_no_wvalid	Data before control and no wvalid
cover_order_control_dbc_ongoing_no_wlast	Data before control and no wlast
cover_order_control_dbc_ongoing_wlast	Data before control and an ongoing wlast
cover_order_data_dbc_limit	Last transaction comes for DBC operation.
cover_order_data_dbc_no_limit_pending_ctl	Control information before last transaction of DBC operation.
cover_order_rready_before_rvalid	rready can come before rvalid
cover_order_rready_after_rvalid	rready can come after rvalid
cover_order_rready_with_rvalid	rready can come with rvalid
cover_order_wready_before_wvalid	wready can come before wvalid
cover_order_wready_after_wvalid	wready can come after wvalid
cover_order_wready_with_wvalid	wready can come with wvalid
cover_order_bready_before_bvalid	bready can come before bvalid
cover_order_bready_after_bvalid	bready can come after bvalid
cover_order_bready_with_bvalid	bready can come with bvalid
cover_order_wlast_before_wready	wlast can come before wready
cover_order_wlast_after_wready	wlast can come after wready
cover_order_wlast_with_wready	wlast can come with wready
cover_order_awvalid_arvalid_together	awvalid can come with arvalid
1.7 Max Wait Cycles Cover	

AXI ABVIP User Guide
Appendix D: AXI5 ABVIP Package Contents--Coverage Checks

cover_max_wait_for_wr_request	Maximum number of cycles for which awready is low and awvalid is high
cover_max_wait_for_wr_data	Maximum number of cycles for which wready is low and wvalid is high
cover_max_wait_for_wr_response	Maximum number of cycles for which bready is low and bvalid is high
cover_max_wait_for_rd_request	Maximum number of cycles for which arready is low and arvalid is high
cover_max_wait_for_rd_response	Maximum number of cycles for which rready is low and rvalid is high

1.8 Read/Write Sequence Covers

cover_seq_rd_len_1	Read sequence with arlen 0
cover_seq_rd_len_2	Read sequence with arlen 1
cover_seq_rd_len_3	Read sequence with arlen 2
cover_seq_rd_len_4	Read sequence with arlen 3
cover_seq_wr_len_1	Write sequence with awlen 0
cover_seq_wr_len_2	Write sequence with awlen 1
cover_seq_wr_len_3	Write sequence with awlen 2
cover_seq_wr_len_4	Write sequence with awlen 3
cover_seq_rd_rresp_out_of_order	Read response out of order possible
cover_seq_wr_bresp_out_of_order	Write response out of order possible

1.9 Exclusive Access Covers

cover_rresp_exokay	Exokay response for rresp possible
cover_bresp_exokay	Exokay response for bresp possible
cover_ex_arsize_0_bytecount_1	During exclusive read, byte count is 1
cover_ex_arsize_0_bytecount_2	During exclusive read, byte count is 2
cover_ex_arsize_0_bytecount_4	During exclusive read, byte count is 4
cover_ex_arsize_0_bytecount_8	During exclusive read, byte count is 8
cover_ex_arsize_0_bytecount_16	During exclusive read, byte count is 16
cover_ex_arsize_1_bytecount_2	During exclusive read, byte count is 2

AXI ABVIP User Guide
Appendix D: AXI5 ABVIP Package Contents--Coverage Checks

cover_ex_arsize_1_bytecount_4	During exclusive read, byte count is 4
cover_ex_arsize_1_bytecount_8	During exclusive read, byte count is 8
cover_ex_arsize_1_bytecount_16	During exclusive read, byte count is 16
cover_ex_arsize_2_bytecount_4	During exclusive read, byte count is 4
cover_ex_arsize_2_bytecount_8	During exclusive read, byte count is 8
cover_ex_arsize_2_bytecount_16	During exclusive read, byte count is 16
cover_ex_arsize_2_bytecount_32	During exclusive read, byte count is 32
cover_ex_arsize_2_bytecount_64	During exclusive read, byte count is 64
cover_ex_arsize_3_bytecount_8	During exclusive read, byte count is 8
cover_ex_arsize_3_bytecount_16	During exclusive read, byte count is 16
cover_ex_arsize_3_bytecount_32	During exclusive read, byte count is 32
cover_ex_arsize_3_bytecount_64	During exclusive read, byte count is 64
cover_ex_arsize_3_bytecount_128	During exclusive read, byte count is 128
cover_ex_awsiz_0_bytecount_1	During exclusive write, byte count is 1
cover_ex_awsiz_0_bytecount_2	During exclusive write, byte count is 2
cover_ex_awsiz_0_bytecount_4	During exclusive write, byte count is 4
cover_ex_awsiz_0_bytecount_8	During exclusive write, byte count is 8
cover_ex_awsiz_0_bytecount_16	During exclusive write, byte count is 16
cover_ex_awsiz_1_bytecount_2	During exclusive write, byte count is 2
cover_ex_awsiz_1_bytecount_4	During exclusive write, byte count is 4
cover_ex_awsiz_1_bytecount_8	During exclusive write, byte count is 8
cover_ex_awsiz_1_bytecount_16	During exclusive write, byte count is 16
cover_ex_awsiz_2_bytecount_4	During exclusive write, byte count is 4
cover_ex_awsiz_2_bytecount_8	During exclusive write, byte count is 8

AXI ABVIP User Guide
Appendix D: AXI5 ABVIP Package Contents--Coverage Checks

cover_ex_awsiz_2_bytecount_16	During exclusive write, byte count is 16
cover_ex_awsiz_2_bytecount_32	During exclusive write, byte count is 32
cover_ex_awsiz_2_bytecount_64	During exclusive write, byte count is 64
cover_ex_awsiz_3_bytecount_8	During exclusive write, byte count is 8
cover_ex_awsiz_3_bytecount_16	During exclusive write, byte count is 16
cover_ex_awsiz_3_bytecount_32	During exclusive write, byte count is 32
cover_ex_awsiz_3_bytecount_64	During exclusive write, byte count is 64
cover_ex_awsiz_3_bytecount_128	During exclusive write, byte count is 128
cover_ex_max_number_of_bytes_read	The maximum number of bytes that can be transferred in a read exclusive burst is 128.
cover_ex_max_number_of_bytes_write	The maximum number of bytes that can be transferred in a write exclusive burst is 128.
cover_ex_exread_start	Exclusive read possible
cover_ex_exwrite_start	Exclusive write possible
cover_ex_exread_exwrite_same_cycle	Exclusive read and write at the same cycle
cover_ex_exread_followed_by_exread	Exclusive read followed by another exclusive read
cover_ex_exread_not_exread_exread	Exclusive read sequence
cover_ex_exwrite_followed_by_exwrite	exwrite followed by another exclusive write
cover_ex_wr_is_noncacheable	Exclusive write burst is noncacheable
cover_ex_rd_is_noncacheable	Exclusive read burst is noncacheable
cover_ex_exwrite_after_exread	Exclusive write after exclusive read
cover_ex_arlock_normal	Normal access possible during exclusive read
cover_ex_awlock_normal	Normal access possible during exclusive write
cover_ex_tbl_full	Exclusive table full
cover_ex_tbl_full_to_empty	Exclusive table full to empty
2.0 Internal Table Covers	
cover_rd_tbl_full	Read table full
cover_rd_tbl_full_2_empty	Read table full to empty cover
cover_wr_tbl_full	Write table data full cover

cover_wr_tbl_full_2_empty	Write table full to empty cover
cover_wr_tbl_full_control	Write table control full
cover_wr_tbl_full_data	Write table full
cover_rd_tbl_cnt	Internal read table count cover
2.1 Atomic Transaction Covers	
cover_atomic_rd_wr_rsp_same_cycle	Covers for property when both read and write response of a atomic transaction comes in same cycle.
cover_atomic_rd_rsp_wr_rsp	Covers for property when read response comes before write response of a atomic transaction.
cover_atomic_wr_wrp_rd_rsp	Covers for property when write response comes before read response of a atomic transaction.
cover_atomic_rdRsp_wrRsp	Covers for property for atomic and read transaction.
2.2 Read Data Chunking Covers	
cover_data_chunking_aligned	Covers Read Data Chunking with DATA_WIDTH 256
cover_data_chunking_unaligned	Covers Read Data Chunking with DATA_WIDTH 256 with unaligned address
2.3 Unique ID Covers	
cover_unq11_rd_diff_id	Covers for unique ID indication, two transaction with high aridunq signal
cover_unq10_rd_diff_id	Covers for unique ID indication, two transaction with one high and one low aridunq signal
cover_unq01_rd_diff_id	Covers for unique ID indication, two transaction with one high and one low aridunq signal
cover_unq00_rd_diff_id	Covers for unique ID indication, two transaction with both low aridunq signal
cover_unq_rd_tx_same_cycle	Covers for unique ID indication, two read transaction at same cycle
cover_unq_rd_transaction	Cover for whole write transaction with aridunq
cover_unq_rd_tx_diff_cycle	Covers for unique ID indication, two read transaction at different cycle
cover_unq_aridunq_ridunq_same_low	Read request sends low aridunq, read response has low ridunq signal in return
cover_unq_aridunq_ridunq_same_high	Read request sends high aridunq, read response has high ridunq signal in return

AXI ABVIP User Guide
Appendix D: AXI5 ABVIP Package Contents--Coverage Checks

cover_unql1_wr_diff_id	Covers for unique ID indication, two transaction with high awidunq signal
cover_unql0_wr_diff_id	Covers for unique ID indication, two transaction with one high and one low awidunq signal
cover_unq01_wr_diff_id	Covers for unique ID indication, two transaction with one high and one low awidunq signal
cover_unq00_wr_diff_id	Covers for unique ID indication, two transaction with both low awidunq signal
cover_unq_wr_tx_same_cycle	Covers for unique ID indication, two write transaction at same cycle
cover_unq_wr_transaction	Cover for whole write transaction with awidunq
cover_unq_wr_tx_diff_cycle	Covers for unique ID indication, two write transaction at different cycle
cover_unq_awidunq_bidunq_same_low	write request sends low awidunq, write response has low bidunq signal in return
cover_unq_awidunq_bidunq_same_high	write request sends high awidunq, write response has high bidunq signal in return
2.4 MTE Covers	
cover_invalid_tag_op_wrtransaction	Covers for invalid tag transaction
cover_update_tag_op_wrtransaction	Covers for update tag transaction
cover_wtag_wvalid_comb	Covers for all possible values of wtag
cover_wtag_wvalid_comb	Covers for all possible values of wtagupdate
cover_transfer_tag_op_wrtransaction	Covers for transfer tag transaction
cover_match_tag_op_wrtransaction	Covers for whole tagmatch transaction
cover_bvalid_bcomp_btagmatch_pass	Covers for pass match and combined response
cover_bvalid_btagmatch_pass	Covers for btagmatch pass response part
cover_bvalid_bcomp	Covers for match response part 1
cover_bvalid_bcomp_btagmatch_fail	Covers for fail match and combined response
cover_bvalid_btagmatch_fail	Covers for btagmatch fail response part
cover_bvalid_btagmatch	Covers for btagmatch response part
cover_invalid_tag_op_rdtransaction	Covers for INVALID tag operation

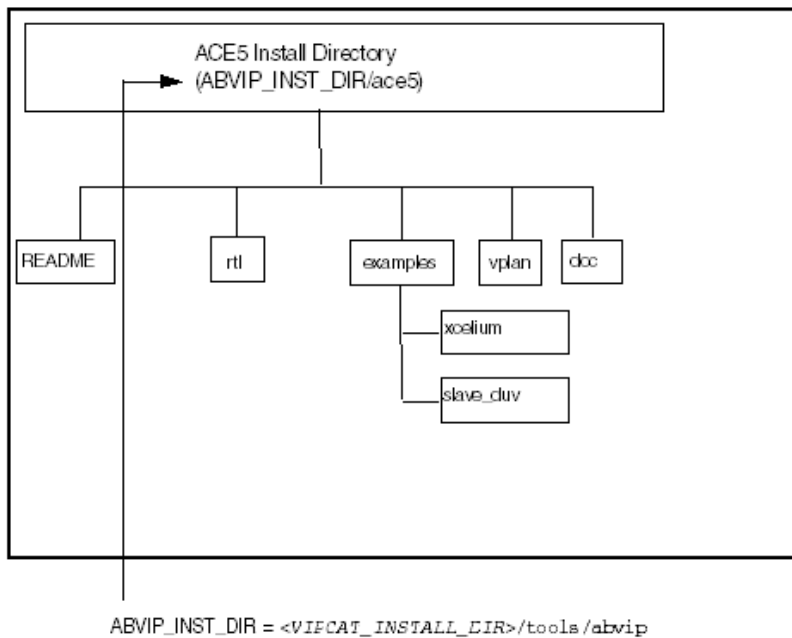
cover_transfer_tag_op_rdtransaction	Covers for TRANSFER tag operation
cover_transfer_tag_op_rdtransaction	Covers for FETCH tag operation
cover_rtag_rvalid_comb	Covers for every value of rtag

Appendix E: ACE5 ABVIP Package Contents

- [Package Contents](#)
- [ACE5 ABVIP Pin-Level Interface](#)
- [ACE5 Monitor Parameters](#)
- [ACE5 Checks](#)
 - [Compliance Checks](#)
- [Coverage Checks](#)

This appendix describes the contents of the ACE5 ABVIP package. [Figure B-1](#) shows the package directory structure.

Figure B-1 Directory Structure of the ACE5 ABVIP Package



Package Contents

[Table B-1](#) shows the contents of the ACE5 installation directory.

Table B-1 Contents of the ACE5 Installation Directory



README	Mentions the contents of the package.
rtl	Contains the code for ACE5 ABVIP.
examples	Contains the examples of ACE5 ABVIP to be used in formal and simulation based verification.
doc	Contains the AXI ABVIP user guide.

[Table B-2](#) shows the files stored in the rtl folder.

Table B-2 Contents of the rtl Folder

Version.v	Prints the version number of the ACE ABVIP.
-----------	---

AXI ABVIP User Guide
Appendix E: ACE5 ABVIP Package Contents--Package Contents

cdn_abvip_ace5_master.svp	ACE5 master driving ACE master interface signals and checking ACE5 slave interface signals.
cdn_abvip_ace5_slave.svp	ACE5 slave driving ACE5 slave interface signals and checking ACE5 master interface signals.
cdn_abvip_ace5_monitor.svp	ACE5 slave checking ACE5 slave and ACE5 master interface signals.
cdn_abvip_amba5_g_master.svp	Included in the cdn_abvip_ace5_master.svp. Drives the ACE5-G version of master interface signal and checks slave interface signals
cdn_abvip_amba5_g_slave.svp	Included in the cdn_abvip_ace5_slave.svp. Drives the ACE5-G version of slave interface signal and checks master interface signals
cdn_abvip_amba5_g_monitor.svp	Included in the cdn_abvip_ace5_monitor.svp. Checks the ACE5-G version of ACE5 master and ACE5 slave interface signals
cdn_abvip_amba5_h_master.svp	Included in the cdn_abvip_ace5_master.svp. Drives the ACE5-H version of master interface signal and checks slave interface signals
cdn_abvip_amba5_h_slave.svp	Included in the cdn_abvip_ace5_slave.svp. Drives the ACE5-H version of slave interface signal and checks master interface signals
cdn_abvip_amba5_h_monitor.svp	Included in the cdn_abvip_ace5_monitor.svp. Checks the ACE5-H version of ACE5 master and ACE5 slave interface signals
cdn_abvip_ace5_lite_master.svp	<p>ACE5-Lite master driving ACE5-Lite master interface signals and checking ACE5-Lite slave interface signals.</p> <div>  Use this module with parameter <code>CONFIG_ACCELERATOR_PORT = 1</code> for ACE5-LiteACP ABVIP master. </div>
cdn_abvip_ace5_lite_slave.svp	<p>ACE5-Lite slave driving ACE5-Lite slave interface signals and checking ACE-Lite master interface signals.</p> <div>  Use this module with parameter <code>CONFIG_ACCELERATOR_PORT = 1</code> for ACE5-LiteACP ABVIP slave. </div>

AXI ABVIP User Guide
Appendix E: ACE5 ABVIP Package Contents--Package Contents

cdn_abvip_ace5_lite_monitor.svp	<p>ACE5-Lite slave checking ACE5-Lite slave and ACE-Lite master interface signals.</p> <div style="border: 1px solid black; padding: 5px; margin-top: 10px;"> <p>⚠ Use this module with parameter CONFIG_ACCELERATOR_PORT = 1 for ACE5-LiteACP ABVIP monitor.</p> </div>
cdn_abvip_ace5_lite_dvm_master.svp	ACE5-Lite (with DVM interface) master by driving ACE5-Lite master interface signals and checking ACE5-Lite slave interface signals.
cdn_abvip_ace5_lite_dvm_slave.svp	ACE5-Lite (with DVM interface) slave driving ACE5-LITE slave interface signals and checking ACE5-Lite master interface signals.
cdn_abvip_ace5_lite_dvm_monitor.svp	ACE5-Lite (with DVM interface) slave checking ACE5-Lite slave and ACE5-Lite master interface signals.
cdn_abvip_amba5_g_ace_lite_master.svp	Included in cdn_abvip_ace5_lite_master.svp and cdn_abvip_ace5_lite_dvm_master.svp file. Drives ACE5-G-Lite master interface signals and checking ACE5-G-Lite slave interface signals.
cdn_abvip_amba5_g_ace_lite_slave.svp	Included in cdn_abvip_ace5_lite_slave.svp and cdn_abvip_ace5_lite_dvm_slave.svp file. Drives ACE5-G-Lite slave interface signals and checking ACE5-G-Lite master interface signals.
cdn_abvip_amba5_g_ace_lite_monitor.svp	Included in cdn_abvip_ace5_lite_monitor.svp and cdn_abvip_ace5_lite_dvm_monitor.svp file. Checks ACE5-G-Lite slave interface signals,ACE5-G-Lite master interface signals.
cdn_abvip_amba5_h_ace_lite_master.svp	Included in cdn_abvip_ace5_lite_master.svp and cdn_abvip_ace5_lite_dvm_master.svp file. Drives ACE5-H-Lite master interface signals and checking ACE5-H-Lite slave interface signals.
cdn_abvip_amba5_h_ace_lite_slave.svp	Included in cdn_abvip_ace5_lite_slave.svp and cdn_abvip_ace5_lite_dvm_slave.svp file. Drives ACE5-H-Lite slave interface signals and checking ACE5-H-Lite master interface signals.
cdn_abvip_amba5_h_ace_lite_monitor.svp	Included in cdn_abvip_ace5_lite_monitor.svp and cdn_abvip_ace5_lite_dvm_monitor.svp file. Checks ACE5-H-Lite slave interface signals,ACE5-H-Lite master interface signals.

amba4_defines.svh

AMBA definition file

The example folder contains examples for running the ACE5 monitor in different configurations. Each subdirectory contains a README file that describes the steps for running the example and the content of that folder.

Table B-3 Contents of the Example Folder in ACE5 installation directory

README	Lists the contents of example directory.
slave_duv	Contains Verilog example for verifying ACE5 slave using ACE5 ABVIP in formal using Jasper
xcelium	Contains Verilog example for verifying ACE5 slave using AXI5 ABVIP in simulation using Xcelium

ACE5 ABVIP Pin-Level Interface

The pin-level interface is a collection of HDL signals contained in an HDL module and connected to the DUV. A description of the pin-level interface signals for the ACE5 ABVIP is listed in [Table B-4](#).

Table B-4 The ACE5 Monitor Pin Interface

Pin	Source	Description
aclk	Clock source	System clock
aresetn	Reset controller	System reset--active low
awid[ID_WIDTH-1:0]	Master	Write address ID--AWID
awidunq	Master	Write Unique ID Indicator - AWIDUNQ
awaddr[ADDR_WIDTH-1:0]	Master	Write address--AWADDR
awlen[LEN_WIDTH-1:0]	Master	Burst length--AWLEN
awsize[SIZE_WIDTH-1:0]	Master	Burst size--AWSIZE
awburst[BURST_WIDTH-1:0]	Master	Burst type--AWBURST
awlock	Master	Lock type--AWLOCK
awcache[CACHE_WIDTH-1:0]	Master	Cache type--AWCACHE
awprot[PROT_WIDTH-1:0]	Master	Protection type--AWPROT
awqos[QOS_WIDTH-1:0]	Master	Quality of service-AWQOS
awregion[REGION_WIDTH-1:0]	Master	Address region-AWREGION
awuser[AUSER_WIDTH-1:0]	Master	User signal--AWUSER
awdomain[DOMAIN_WIDTH-1:0]	Master	Domain signal- AWDOMAIN
awsnoop[AWSNOOP_WIDTH-1:0]	Master	Snoop signal-AWSNOOP
awbar[BAR_WIDTH-1:0]	Master	Bar signal-AWBAR
awmpam[10:0]	Master	Write Channel MPAM information – AWMPAM
awvalid	Master	Write address valid--AWVALID
awready	Slave	Write address ready--AWREADY

awcmo[1:0]	Master	Write address channel CMO indicator
wdata[DATA_WIDTH-1:0]	Master	Write data--WDATA
wstrb[DATA_WIDTH/8-1:0]	Master	Write strobes--WSTRB
wuser[WUSER_WIDTH-1:0]	Master	User signal- wuser
wlast	Master	Write last--WLAST
wack	Master	Write acknowledgement- WACK
wvalid	Master	Write valid--WVALID
wready	Slave	Write ready--WREADY
bid[ID_WIDTH-1:0]	Slave	Response ID--BID
bidunq	Slave	Write Response Unique ID Indicator - BIDUNQ
bresp[BRESP_WIDTH-1:0]	Slave	Write response--BRESP
buser[BUSER_WIDTH-1:0]	Slave	User signal--buser
bvalid	Slave	Write response valid--BVALID
bready	Master	Response ready--BREADY
arid[ID_WIDTH-1:0]	Master	Read address ID--ARID
aridunq	Master	Read Unique ID Indicator--ARIDUNQ
araddr[ADDR_WIDTH-1:0]	Master	Read address--ARADDR
arlen[LEN_WIDTH-1:0]	Master	Burst length--ARLEN
arsize[SIZE_WIDTH-1:0]	Master	Burst size--ARSIZE
arburst[BURST_WIDTH-1:0]	Master	Burst type--ARBURST
arlock	Master	Lock type--ARLOCK
arcache[CACHE_WIDTH-1:0]	Master	Cache type--ARCACHE
arprot[PROT_WIDTH-1:0]	Master	Protection type--ARPROT
archunken	Master	Read Data Chunking Enable--ARCHUNKEN
argos[QOS_WIDTH-1:0]	Master	Quality of service-ARQOS
aruser[ARUSER_WIDTH-1:0]	Master	User signal -ARUSER
arregion[REGION_WIDTH-1:0]	Master	Address region-ARREGION
ardomain[DOMAIN_WIDTH-1:0]	Master	Domain signal-ARDOMAIN
armpam[10:0]	Master	Read Channel MPAM information – ARMPAM
arsnoop[SNOOP_WIDTH-1:0]	Master	Snoop signal-ARSNOOP
arbar[BAR_WIDTH-1:0]	Master	Master Bar signal- ARBAR
arvalid	Master	Read address valid--ARVALID
arready	Slave	Read address ready-ARREADY

rid[ID_WIDTH-1:0]	Slave	Read ID tag--RID
ridunq	Slave	Read Response Unique ID Indicator--RIDUNQ
rdata[DATA_WIDTH-1:0]	Slave	Read data--RDATA
rresp[RRESP_WIDTH-1:0]	Slave	Read response--RRESP
rlast	Slave	Read last--RLAST
ruser[USER_WIDTH-1:0]	Slave	User signal-RUSER
rack	Master	Read acknowledgement- RACK
rvalid	Slave	Read valid--RVALID
rready	Slave	Read ready--RREADY
rchunkv	Slave	Valid signal of rchunknum and rchunkstrb – RCHUNKV
rchunknum [CHUNKNUM_WIDTH-1:0]	Slave	Read Data Chunk Number--RCHUNKNUM
rchunkstrb [CHUNKSTB_WIDTH-1:0]	Slave	Read Data Chunk Strobe --RCHUNKSTRB
acaddr[ADDR_WIDTH-1:0]	Interconnect	Snoop address- ACADDR
acprot[PROT_WIDTH-1:0]	Interconnect	Snoop protection signal- ACPROT
acsnoop[SNOOP_WIDTH-1:0]	Interconnect	Snoop signal- ACSNOOP
acvalid	Interconnect	Snoop valid- ACVALID
aready	Interconnect	Snoop ready- AREADY
crvalid	Master	Snoop response valid
cready	Master	Snoop response ready
crresp	Master	Snoop response
cdvalid	Master	Snoop data valid
cdready	Master	Snoop data ready
cddata	Master	Snoop data
cdlast	Master	Snoop data last
rpoison	Slave	Read Data Poison
wpoison	Master	Write Data Poison
cdpoison	Master	Snoop Data Poison
rdatachk	Slave	Read Data Check
wdatachk	Master	Write Data Check
cddatachk	Master	Snoop Data Check
varqosaccept	Slave	Read Address QoS Accept
vawqosaccept	Slave	Write Address QoS Accept
artrace	Master	Read address trace
rtrace	Slave	Read data trace

awtrace	Master	Write address trace
wtrace	Slave	Write data trace
btrace	Slave	Write response trace
actrace	Interconnect	Snoop address trace
ctrace	Master	Snoop response trace
cdtrace	Master	Snoop data trace
arloop	Master	Read address loopback
rloop	Slave	Read data loopback
awloop	Master	Write address loopback
bloop	Slave	Write response loopback
awakeup	Master	low power wakeup signal
acwakeup	Interconnect	Snoop Wake Up
syscoreq	Master	Coherency connect request
syscoack	Interconnect	Coherency connect acknowledge
arnsaid	Master	Read address non-secure access identifier
awnsaid	Master	Write address non-secure access identifier
crnsaid	Master	Snoop Response Non-secure Access Identifier
arvmidext	Master	Read Address VMID Extension
acvmidext	Interconnect	Snoop Address VMID Extension
awvalidchk	Master	Check signal for AWVALID
awreadychk	Slave	Check signal for AWREADY
awidchk [IDWRCHK_WIDTH-1:0]	Master	Check signal for AWID
awaddrchk [ADDRCHK_WIDTH-1:0]	Master	Check signal for AWADDR
awlenchk	Master	Check signal for AWLEN
awctlchk0	Master	Check signal for AWSIZE, AWBURST, AWLOCK, AWPROT
awctlchk1	Master	Check signal for AWREGION, AWCACHE, AWQOS
awctlchk2	Master	Check signal for AWDOMAIN, AWSNOOP, AWUNIQUE, AWBAR
awctlchk3	Master	Check signal for AWATOP
awuserchk [AWUSERCHK_WIDTH-1:0]	Master	Check signal for AWUSER
awstashnidchk	Master	Check signal for AWSTASHNID, AWSTASHNIDEN
awstashlpidchk	Master	Check signal for AWSTASHLPID, AWSTASHLPDEN

awtracechk	Master	Check signal for AWTRACE
awloopchk	Master	Check signal for AWLOOP
awmmuchk	Master	Check signal for AWMMUATST, AWMMUSECSID, AWMMUSSIDV
awmmusidchk [MMUSIDCHK_WIDTH-1:0]	Master	Check signal for AWMMUSID
awmmussidchk [MMUSSIDCHK_WIDTH-1:0]	Master	Check signal for AWMMUSSID
awnsaidchk	Master	Check signal for AWNSAID
awmpamchk	Master	Check signal for AWMPAM
awidunqchk	Master	Check signal for AWIDUNQ
awcmo	Master	Write address cmo signal
wvalidchk	Master	Check signal for WVALID
wreadychk	Slave	Check signal for WREADY
wdatachk [DATACHK_WIDTH-1:0]	Master	Check signal for WDATA
wstrbchk [WSTRBCHK_WIDTH-1:0]	Master	Check signal forWSTRB
wlastchk	Master	Check signal for WLAST
wuserchk [WUSERCHK_WIDTH-1:0]	Master	Check signal for WUSER
wpoisonchk [POISONCHK_WIDTH-1:0]	Master	Check signal for WPOISON
wtracechk	Master	Check signal for WTRACE
bvalidchk	Slave	Check signal for BVALID
breadychk	Master	Check signal for BREADY
bidchk [IDWRCHK_WIDTH-1:0]	Slave	Check signal for BID
brespchk	Slave	Check signal for BRESP
btracechk	Slave	Check signal for BTRACE
bloopchk	Slave	Check signal for BLOOP
buserchk [BUSERCHK_WIDTH-1:0]	Slave	Check signal for BUSER
arreadychk	Slave	Check signal for ARREADY
arvalidchk	Master	Check signal for ARVALID
aridchk [IDRDCHK_WIDTH-1:0]	Master	Check signal for ARID
araddrchk [ADDRCHK_WIDTH-1:0]	Master	Check signal for ARADDR
arlenchk	Master	Check signal for ARLEN
arctlchk0	Master	Check signal for ARSIZE, ARBURST, ARLOCK, ARPROT
arctlchk1	Master	Check signal for ARREGION, ARCACHE, ARQOS
arctlchk2	Master	Check signal for ARDOMAIN, ARSNOOP, ARBAR

arctlchk3	Master	Check signal for ARVMIDEXT, ARCHUNKEN
aruserchk [ARUSERCHK_WIDTH-1:0]	Master	Check signal for ARUSER
artracechk	Master	Check signal for ARTRACE
arloopchk	Master	Check signal for ARLOOP
armmuchk	Master	Check signal for ARMMUATST, ARMMUSECSID, ARMMUSSIDV
armmusidchk [MMUSIDCHK_WIDTH-1:0]	Master	Check signal for ARMMUSID
armmuSSIDchk [MMUSSIDCHK_WIDTH-1:0]	Master	Check signal for ARMMUSSID
arnsaidchk	Master	Check signal for ARNSAID
aridunqchk	Master	Check signal for ARIDUNQ
armpamchk	Master	Check signal for ARMPAM
rvalidchk	Slave	Check signal for RVALID
rreadychk	Master	Check signal for RREADY
ridchk [IDRDCHK_WIDTH-1:0]	Slave	Check signal for RID
rdatachk [DATACHK_WIDTH -1:0]	Slave	Check signal for RDATA
rrespchk	Slave	Check signal for RRESP
rloopchk	Slave	Check signal for RLOOP
ruserchk [RUSERCHK_WIDTH-1:0]	Slave	Check signal for RUSER
rtracechk	Slave	Check signal for RTRACE
rchunkchk	Slave	Check signal for RCHUNKEN, RCHUNKSTRB, RCHUNKNUM
rlastchk	Slave	Check signal for RLAST
rpoisonchk [POISONCHK_WIDTH-1:0]	Slave	Check signal for RPOISON
acvalidchk	Interconnect	Check signal for ACVALID
acreadychk	Master	Check signal for ACREADY
acctlchk	Interconnect	Check signal for ACSNOOP, ACPROT
acvmidextchk	Interconnect	Check signal for ACVMIDEXT
actracechk	Interconnect	Check signal for ACTRACE
crvalidchk	Master	Check signal for CRVALID
crreadychk	Interconnect	Check signal for CRREADY
crrespchk	Master	Check signal for CRRESP
crtracechk	Master	Check signal for CRTRACE
crnsaidchk	Master	Check signal for CRNSAID
cdreadychk	Interconnect	Check signal for CDREADY

cdvalidchk	Master	Check signal for CDVALID
cddatachk	Master	Check signal for CDDATA
cdlastchk	Master	Check signal for CDLAST
cdpoisonchk	Master	Check signal for CDPOISON
cdtracechk	Master	Check signal for CDTRACE
rackchk	Master	Check signal for RACK
wackchk	Master	Check signal for WACK
vawqosacceptchk	Slave	Check signal for VAWQOSACCEPT
varqosacceptchk	Slave	Check signal for VARQOSACCEPT
awakeupchk	Master	Check signal for AWAKEUP
acakeupchk	Slave	Check signal for ACAKEUP
syscoreqchk	Master	Check signal for SYSCOREQ
syscoackchk	Interconnect	Check signal for SYSCOACK
armmuflow	Master	Indicates the SMMU flow for managing translation faults ARMMUFLOW
awmmuflow	Master	Indicates the SMMU flow for managing translation faults AWMMUFLOW
awtagop	Master	Write request tag operation
artagop	Master	Read request tag operation
wtag	Master	The tag that is associated with write data
wtagupdate	Master	Indicates which tags must be written to memory in an Update operation
rtag	Slave	The tag that is associated with read data
btagmatch	Slave	Indicates the result of a tag comparison on a write transaction
wtagchk	Master	Parity Check signal write tag
rtagchk	Slave	Parity Check signal read tag



ACE5 Monitor Parameters

The ACE5 ABVIP is highly configurable. You can change the parameters listed in [Table B-5](#) as required.

Table B-5 ACE5 Monitor Parameters

Parameter	Description
-----------	-------------

ADDR_WIDTH	<p>Address bus width.</p> <p>Default value is 32. The minimum value of ADDR_WIDTH during byte strobe checking (BYTE_STROBE_ON is 1) must be 8.</p>
DATA_WIDTH	<p>Data bus width.</p> <p>Default value is 64.</p>
ID_WIDTH	<p>The number of bits needed to capture the number of IDs supported.</p> <p>Default value is 2.</p>
LEN_WIDTH	<p>The number of bits needed to capture the length of data per request.</p> <p>Default value is 4.</p>
ALL_STROBES_HIGH_ON	<p>Whether all strobes are set to high.</p> <p>The default value is 0, which means this parameter is not set.</p>
ALLOW_SPARSE_STROBE	<p>When this parameter is set, master ABVIP can drive <code>wstrb</code> bits to 0 for one or more valid byte lanes.</p> <p>The default value of this parameter is 1. This parameter is effective only if parameter <code>BYTE_STROBE_ON</code> is set to 1.</p>
BYTE_STROBE_ON	<p>Whether write byte strobe (WSTRB) checking is required.</p> <p>The default value is 0, which means byte strobe checking is not required.</p>
COVERAGE_ON	<p>Whether coverage generation is set.</p> <p>The default value is 0, which is no coverage.</p>
DATA_BEFORE_CONTROL_ON	<p>Whether data before control is supported.</p> <p>The default value is 1, which means data before control is supported.</p>
DATA_ACCEPT_WITH_OR_AFTER_CONTROL	<p>Whether slave can accept data with or after control.</p> <p>The default value is 0, which means data can be accepted before control.</p>
ERROR_CHECK_ON	<p>Set to 1 disable the ABVIP's max address boundary properties. The address will be allowed to cross the max boundary range.</p> <p>The default value is 0.</p>
EXCL_ACCESS_ON	<p>Whether exclusive access is supported.</p> <p>The default value is 0, which means no exclusive access.</p>
BARRIER_ON	<p>Whether barrier transaction is supported.</p> <p>The default is 0 for ACE5 variants, which means barrier transaction is NOT supported.</p>
DVM_ON	<p>Whether DVM transaction is supported.</p> <p>The default is 0, which means no DVM transaction is allowed. This parameter is not present in ACE5-Lite variants.</p>

READ_INTERLEAVE_ON	<p>Whether read data interleaving is supported.</p> <p>The default value is 1, which means read interleave mode is supported.</p>
WRITE_INTERLEAVE_ON	<p>Whether write data interleaving is supported.</p> <p>The default value is 1, which means write interleave mode is supported.</p>
READ_RESP_IN_ORDER_ON	<p>Whether read response in order is supported.</p> <p>The default value is 0, which means read response can come in any order.</p>
WRITE_RESP_IN_ORDER_ON	<p>Whether write response in order is supported.</p> <p>The default value is 0, which means write response can come in any order.</p>
ADDR_HAZARD_CHECKS_ON	<p>Whether address hazard check is supported.</p> <p>The default is 0, which means no hazard checks.</p>
RST_CHECKS_ON	<p>Whether reset checks are supported.</p> <p>The default value is 0, which means no reset checks.</p>
XCHECKS_ON	<p>Support of X-checks. The default value is 0, meaning no X-checks.</p> <div>  Use the Jasper elaborate switch “-enable_sva_isunknown” for X-Prop analysis when this parameter is enabled. </div>
CONFIG_PARAM_CHECKS	<p>Set to 1 will enable the ABVIP parameter checks.</p> <p>Default value is 1.</p>
CONFIG_LIVENESS	<p>Liveness properties will be created for the scenarios whose MAX_WAIT_CYCLES_* parameter is 0. Setting this parameter to 0 disables all liveness properties.</p> <p>Default value is 1.</p> <p>When enabled, all liveness assumptions and assertions are enabled.</p> <p>Also, if CONFIG_LIVENESS and DEADLOCK_CHKS_ON, both parameters are enabled then DEADLOCK_CHKS_ON will take priority. It will disable all liveness properties and will enable only deadlock checks.</p> <div>  Define macro JG_ABVIP_STRONG_SEMANTICS to enable strong semantics for liveness properties when using sv09 and above. </div>



DEADLOCK_CHKS_ON	<p>Liveness properties will be created for the scenarios whose <code>MAX_WAIT_CYCLES_*</code> parameter is 0. Setting this parameter to 1 enables all deadlock checks.</p> <p>Default value is 0.</p> <p>When enabled, then only liveness asserts are enabled and constraints are disabled.</p> <p>Also, if <code>CONFIG_LIVENESS</code> and <code>DEADLOCK_CHKS_ON</code>, both parameters are enabled then <code>DEADLOCK_CHKS_ON</code> will take priority. It will disable all liveness properties and will enable only deadlock checks.</p> <div>  Define macro <code>JG_ABVIP_STRONG_SEMANTICS</code> to enable strong semantics for deadlock checks when using sv09 and above. </div>
CONFIG_STABLE_CHECKS	<p>Set to 1 will enable the ABVIP stable property checks for signals.</p> <p>Default value is 1.</p>
CONFIG_RDATA_MASKED	<p>Set to 0 will enable the X-propagation and stability properties to apply to full Read data.</p> <p>Set to 1 will enable the X-propagation and stability properties to apply to valid byte lane of Read data.</p> <p>Default value is 1.</p>
CONFIG_RD_FULL_STRB	<p>Set to 0 will enable full read strobe not required for the read transaction.</p> <p>Set to 1 will enable full read strobe required for the read transaction.</p> <p>Default value is 0.</p>
CONFIG_CSR_INTERFACE	<p>This parameter is used to connect ABVIP with Jasper CSR App. The default value of this parameter is 0. You can set the parameter value to 1 to enable ABVIP with CSR checkers.</p>
CONFIG_CSR_FEED_EARLY_W	<p>Tune internal CSR logic behavior:</p> <ul style="list-style-type: none"> Set to 0 will feed write data to CSR checker on write response transaction. Set to 1 will feed write data to CSR checker on write data transaction. <p>Default value is 0.</p> <div>  Arrays of size <code>MAX_PENDING_WR*WLAXLEN*DATA_WIDTH</code> will be created in ABVIP when this parameter is 0. </div> <p>-disable_auto_bbox or -no_bbox_m or -no_bbox_i or -bbox_a switch needs to be added in elaborate command to avoid black-boxing of these arrays in ABVIP.</p> <p>Elaboration time will increase in this mode.</p>






CONFIG_CSR_FEED_ON_BVALID	<p>Tune internal CSR logic behavior when CONFIG_CSR_FEED_EARLY_W = 0,</p> <ul style="list-style-type: none"> Set to 0 will feed write data to the CSR checker on write response channel <code>bvalid</code> and <code>bready</code> assertion. Set to 1 will feed write data to the CSR checker on write response channel <code>bvalid</code> assertion. <p>Default value is 0.</p>
CONFIG_CSR_FEED_ON_RVALID	<p>Tune internal CSR logic behavior,</p> <p>Set to 0 will feed read data to CSR checker on read response channel <code>rvalid</code> and <code>rready</code> assertion.</p> <p>Set to 1 will feed read data to CSR checker on read response channel <code>rvalid</code> assertion.</p> <p>Default value is 0.</p>
LOW_POWER_ON	<p>Whether low power interface checking is required.</p> <p>Default value is 0, which means no low power interface checking. If low power mode is not supported, low power interface signals such as <code>CSYSREQ</code>, <code>CACTIVE</code>, and <code>CSYSACK</code> must be tied to 1'b1.</p>
MAX_WAIT_CYCLES_ON	<p>If set to 1, all <code>MAX_WAIT_CYCLES_*</code> parameters can be used. If set to 0, the <code>MAX_WAIT_CYCLES_*</code> parameters are assigned a value of infinity by using liveness properties. This value means, for instance, if an <code>AWVALID</code> goes high, <code>AWREADY</code> must eventually go high.</p> <div style="border: 1px solid #f0e68c; padding: 10px; margin-top: 10px;"> <p>⚠ Define macro <code>JG_ABVIP_STRONG_SEMANTICS</code> to enable strong semantics for liveness properties when using sv09 and above.</p> </div>
MAX_WAIT_CYCLES_AW	The maximum number of cycles for which <code>AWREADY</code> is low after <code>AWVALID</code> goes high.
MAX_WAIT_CYCLES_W	The maximum number of cycles for which <code>WREADY</code> is low after <code>WVALID</code> goes high.
MAX_WAIT_CYCLES_B	The maximum number of cycles for which <code>BREADY</code> is low after <code>BVALID</code> goes high.
MAX_WAIT_CYCLES_AR	The maximum number of cycles for which <code>ARREADY</code> is low after <code>ARVALID</code> goes high.
MAX_WAIT_CYCLES_R	The maximum number of cycles for which <code>RREADY</code> is low after <code>RVALID</code> goes high.
MAX_WAIT_CYCLES_AW_W	The maximum number of cycles for which write signals(<code>WVALID</code> , <code>WREADY</code>) are low after <code>AWVALID</code> goes high
MAX_WAIT_CYCLES_W_AW	The maximum number of cycles for which write address channel signals(<code>AWVALID</code> , <code>AWREADY</code>) are low after <code>WVALID</code> goes high
MAX_WAIT_CYCLES_W_B	The maximum number of cycles for which write response is low after <code>wvalid</code> goes high.
MAX_WAIT_CYCLES_AW_B	The maximum number of cycles for which write response is low after <code>AWVALID</code> goes high



MAX_WAIT_CYCLES_AR_R	The maximum number of cycles for which read response is low after <code>ARVALID</code> goes high
MAX_PENDING_SNOOP	The maximum number of pending snoop requests for which the response or data is incomplete. This is not present in ACE-LITE. This parameter should never be set to 0.
MAX_PENDING_BAR	The maximum number of pending barrier requests for which the response or data is incomplete. This parameter should never be set to 0.
MAX_PENDING_DVM_SYNC	The maximum number of pending DVM Sync requests allowed. This is not present in ACE-LITE. This parameter should never be set to 0.
MAXLEN	The maximum number of beats that can be transferred in a burst for read/write requests. Default value is 256. This will constrain <code>AWLEN</code> value to be less than <code>MAXLEN</code> . When <code>CONFIG_CSR_INTERFACE</code> is set, the default value of this parameter is 16.
MAXLEN_RD	The maximum number of beats that can be transferred in a burst for read requests. Default value is 256. This constrains <code>ARLEN</code> value to be less than <code>MAXLEN_RD</code> . When <code>CONFIG_CSR_INTERFACE</code> is set, the default value of this parameter is 16.
MAXLEN_WR	The maximum number of beats that can be transferred in a burst for write requests. Default value is 256. This constrains <code>AWLEN</code> value to be less than <code>MAXLEN_WR</code> . When <code>CONFIG_CSR_INTERFACE</code> is set, the default value of this parameter is 16.
MAX_WAIT_CYCLES_AC	The maximum number of cycles for which <code>ACREADY</code> is low after <code>ACVALID</code> goes high.
MAX_WAIT_CYCLES_CR	The maximum number of cycles for which <code>CRREADY</code> is low after <code>CRVALID</code> goes high.
MAX_WAIT_CYCLES_CD	The maximum number of cycles for which <code>CDREADY</code> is low after <code>CDVALID</code> goes high.
MAX_WAIT_CYCLES_RACK	The maximum number of cycles for which <code>RACK</code> is low after the read response.
MAX_WAIT_CYCLES_WACK	The maximum number of cycles for which <code>WACK</code> is low after the write response.
HINT_MULTIPART	DVM multipart Hint message. The default value is 1, which means DVM hint multipart is supported.
SHAREABLE_START_ADDR	Shareable start address. Default value is 0.
SHAREABLE_END_ADDR	Shareable end address. The default value is $(10 * \text{CACHE_LINE_SIZE} - 1)$
NONSHAREABLE_START_ADDR	Non-Shareable start address. The default value is $(10 * \text{CACHE_LINE_SIZE})$
NONSHAREABLE_END_ADDR	Non-Shareable end address. The default value is $(10 * \text{CACHE_LINE_SIZE}) + 5120$






SYSTEM_START_ADDR	System start address. The default value is $(10 * \text{CACHE_LINE_SIZE}) + 5121$
CLK_CNTL_PRESENT	Clock controller present. The default value is 0, which means the clock controller is not present.
PROT_IN_OVERLAP	Prot value in overlapping address calculation. The default value is 1, which means the prot value is used in address overlapping calculation.
CDNS_ABVIP_STOP_OVERFLOW	Disable ABVIP assumptions to prevent table overflow. When set to 1, master ABVIP will not generate new *valid signals when MAX_PENDING transactions are outstanding. Also, when set to 1, slave ABVIP will not generate *ready signals when MAX_PENDING transactions are outstanding. The default value is 1, which means ABVIP overflow assumptions are present.
CDNS_ABVIP_STOP_OVERFLOW_RD	Disable ABVIP assumptions to prevent Read table overflow. Default value is same as that of CDNS_ABVIP_STOP_OVERFLOW i.e. 1, which means ABVIP overflow assumptions are present.
CDNS_ABVIP_STOP_OVERFLOW_WR	Disable ABVIP assumptions to prevent Write table overflow. Default value is same as that of CDNS_ABVIP_STOP_OVERFLOW i.e. 1, which means ABVIP overflow assumptions are present.
CDNS_ABVIP_STOP_OVERFLOW_SNP	Disable ABVIP assumptions to prevent Snoop table overflow. Default value is same as that of CDNS_ABVIP_STOP_OVERFLOW i.e. 1, which means ABVIP overflow assumptions are present.
CDNS_READY_OVERFLOW_CHECKS	When set to 1, master or monitor ABVIP will check that slave does not generate *ready signals when MAX_PENDING transactions are outstanding. Default value is 1.
CDNS_VALID_OVERFLOW_CHECKS	When set to 1, slave or monitor ABVIP will check that master does not generate *valid signals when MAX_PENDING transactions are outstanding. Default value is 1.
CDNS_ABVIP_SOFT_CONS_ON	Disable assumptions required for assertion driven simulation (ADS) mode. The default value is 1, which means assumptions for ADS are active.
SNOOP_FILTER_PRESENT	Snoop filter present. The default value is 0, which means the snoop filter is not present.
CCI400_MULTI_CACHE_LINE	Parameter to apply CCI400 related assumptions.
WREVICT_ON	WriteEvict support. The default value is 0, which means WriteEvict is not supported.
DVM_V8	Set to 1 to enable DVM properties complaint to DVMv8 message format.
CDNS_READY_OVERFLOW_CHECKS	This parameter is used to enable/disable properties that prevent the assertion of ready signal when table/tracker is full. Default value is 1. It is recommended to set it to 0 when doing verification with cci400.


RECM_CHECKS_ON	Setting to 1 enables recommended checks only if the corresponding feature is set to 1. Default value is 0.
RECM_ASSUME_ON	The default value of this parameter is 1'b1. If the parameter is set to 0 then recommended checks are enabled and recommended constraints are disabled. When set to 1, it enables both recommended checks and constraints.
RECM_CHKS_ALL_ON	Setting to 1 enables all recommended checks irrespective of the corresponding feature parameter. Default value is 0.
RECM_CHECKS_EXCL_ON	Enable recommended exclusive checks. Default value is 1.
CONFIG_SIMULATION DYNAMIC_DATA_WDTH_EN DATA_WIDTH_inB CDDATA_WIDTH_inB	These parameters should be left to their default values.
RECM_CHECKS_BARRIER_ON	Enable recommended barrier check. Default value is 0.
RECM_CHECKS_DVM_ON	Enable recommended DVM checks. Default value is 0.
ADDR_RANGE_CHECKS	This parameter is used to enable/disable address range properties to constraint/check address values in ranges specified by parameters <code>SHAREABLE_START_ADDR</code> , <code>SHAREABLE_END_ADDR</code> , <code>NONSHAREABLE_START_ADDR</code> , and <code>NONSHAREABLE_END_ADDR</code> . The default value of this parameter is 0.
DVM_V8_1	Enable properties related to DVM8.1. Default value is 1. <code>DVM_ON</code> parameter should also be set to 1 when <code>DVM_V8_1</code> is set to 1.
CONFIG_ATOMIC_TRN	This parameter is used to enable/disable properties related to Atomic Transactions. Default value is 1. This parameter is only applicable to ACE5-Lite and ACE5-LiteDVM.
CONFIG_CACHE_STASH	This parameter is used to enable/disable properties related to Cache Stashing. This parameter is only applicable for ACE5-Lite variants. Default value is 1.
CONFIG_DEALLOCATING_TRN	This parameter is used to enable/disable properties related to the deallocating transaction. This parameter is only applicable for ACE5-Lite and ACE5-Lite DVM. Default value is 1.
CONFIG_CLEANSHAREDPERSIST	This parameter is used to enable/disable properties related to the CleanSharedPersist transaction. Default value is 1. This parameter does not apply to ACE5-LiteACP. In the AMBA5-G version, This parameter is used to enable CleanSharedPersist transaction for Write Channel (depends on <code>CMO_ON_WRITE</code> or <code>CMO_ON_READ</code> is supported)
CONFIG_SHAREABLE_DOMAIN_DEP	This parameter is used to suppress deprecated inner shareability domain. Default value is 1.
CONFIG_NSaid	This parameter is used to enable/disable non-secure access identifiers properties. This parameter is not applicable for ACE5-LiteACP. Default value is 1.

CONFIG_LP_COHERENCY_CHKS	This parameter is used to enable/disable Coherency Connection Signaling checks. This parameter is applicable only for ACE5 and ACE5-LiteDVM. Default value is 1.
POISON_WIDTH	Poison bit width. The default value is DATA_WIDTH/64
CONFIG_POISON_SIGNAL_CHKS	This parameter is used to enable/disable poison signal checks. Default value is 1.
CONFIG_DATA_CHK_SIGNAL_CHKS	This parameter is used to enable/disable data to check signal properties. Default value is 1.
CONFIG_QOS_SIGNAL_CHKS	This parameter is used to enable/disable QoS signal properties. Default value is 1. This parameter is not applicable for ACE5-LiteACP
CONFIG_TRACE_SIGNAL_CHKS	This parameter is used to enable/disable trace signal properties. Default value is 1.
CONFIG_USER_LOOPBACK_CHKS	This parameter is used to enable/disable loopback signal properties. Default value is 1. This parameter is not applicable for ACE5-LiteACP.
LOOP_WIDTH	Width of <code>awloop</code> , <code>arloop</code> , <code>bloop</code> , and <code>rloop</code> signals. Default value is 8.
CONFIG_LP_WAKEUP_SIG_CHKS	<p>This parameter is used to enable/disable wakeup signal properties. Default value is 1.</p> <p> LOW_POWER_ON parameter should be set to 1, to enable the wakeup signal properties.</p>
CONFIG_UNTRANSLATED_TRN	<p>This parameter is used to enable/disable Untranslated Transactions.</p> <p>The required signals are supported for version 2 of untranslated transactions if the value of this parameter is 2</p> <p>This parameter does not apply to ACE5-LiteACP and G-Version in Ace-lite_DVM. Default value is 1.</p> <p> CDNS_AMBA5_H parameter should be set to 1, to enable version 2 properties</p>
MMUSID_WIDTH	Width of <code>AxMMUSID</code> signals
MMUSSID_WIDTH	Width of <code>AxMMUSSID</code> signals
UPDATE_EXCL_TBL_ON_RD_RESP	<p>When this parameter is set to 1, the exclusive access trackers are updated based on the response of the exclusive read. This allows you to be less firm on the previous exclusive write response if an overwriting exclusive read is received.</p> <p>The default value of this parameter is 0.</p>
MAX_WAIT_CYCLES_QOSACCEPT	Set to any integral value, the deterministic maximum number of clock cycles taken to accept a transaction with <code>qos</code> level equal to or higher than <code>va*qosaccept</code> . The value 0 means that the property is disabled. The default value of this parameter is 0.

ARM_DSU_ON	<p>When set to 1: It enables DSU features. When set to 0: It disables DSU feature.</p> <p>The default value of this parameter is 0.</p>
CDNS_AMBA5_G	<p>This parameter enables the AMBA5-G features.</p> <p>The default value of this parameter is 0.</p>
CDNS_AMBA5_H	<p>This parameter enables the AMBA5-H features.</p> <p>The default value of this parameter is 0.</p>
CONFIG_UNQ_ID_INDICATOR	<p>When set to 1, it enables the Unique ID Support feature.</p> <p>The default value of this parameter is 0.</p> <div>  CDNS_AMBA5_G parameter should be set to 1, to enable the unique ID signal properties. </div>
CONFIG_MPAM_SUPPORT	<p>Parameter <i>CONFIG_MPAM_SUPPORT</i> enables the interface to support Memory Partitioning and Mapping technology.</p> <p>The default value of this parameter is 0.</p> <div>  CDNS_AMBA5_G parameter should be set to 1, to enable the mpam signal properties. </div>
CONFIG_READ_DATA_CHUNKING	<p>It enables the slave interface to send the read data in the chunk's forms.</p> <p>The default value of this parameter is 0.</p> <div>  CDNS_AMBA5_G parameter should be set to 1, to enable the read data chunking signal properties. </div>
CHUNKNUM_WIDTH	width of the rchunknum signal
CHUNKSTB_WIDTH	width of the rchunkstrb signal
CONFIG_PARITY_CHECK_TYPE	<p>It enables the Parity feature in the interface</p> <p>Set to 0: Disable parity Check scheme, Set to 1: to enable data parity bit, Set to 2: enable all Parity bit</p> <div>  CDNS_AMBA5_G parameter should be set to 1, to enable the read data chunking signal properties. </div>
CONFIG_ODD_PARITY_BYTE_DATA	<p>Set to 0: Disable byte data parity check</p> <div>  CDNS_AMBA5_G parameter should be set to 1, to enable the read data chunking signal properties. </div>

CONFIG_ODD_PARITY_BYTE_ALL	<p>Set to 0: Disable byte all parity check</p> <p> CDNS_AMBA5_G parameter should be set to 1, to enable the read data chunking signal properties.</p>
ADDRCHK_WIDTH	Width of the Address check signal
ARUSERCHK_WIDTH	Width of Read Address User check signal
AWUSERCHK_WIDTH	Width of Write Address User check signal
IDRDCHK_WIDTH	Width of Read ID check signal
DATACHK_WIDTH	Width of Data check signal
MMUSIDCHK_WIDTH	Width of mmusid check signal
MMUSSIDCHK_WIDTH	Width of mmussid check signal
IDWRCHK_WIDTH	Width of Write ID check signal
WSTRBCHK_WIDTH	Width of Write Strb check signal
WUSERCHK_WIDTH	Width of Write user check signal
BUSERCHK_WIDTH	Width of B user check signal
POISONCHK_WIDTH	Width of Poison check signal
RUSERCHK_WIDTH	Width of Read user check signal
EXCL_ACCESS_WRITE_MATCH_EX_READ	Set to 0 to enable the master to generate the illegal transactions in the exclusive access. By default, it will generate the exclusive write transaction matching with exclusive read
EXCL_ACCESS_SLAVE_ON	By default, it is 1. Set to 0 to disable exclusive access of slave
EXCL_ACCESS_WR_RESP_OKAY_ALLOWED	Set to 1 to allow that slave response can send EXOKAY or OKAY response in case of valid exclusive access. By default, It is set to 0 means slave can send only exokay response
CMO_ON_READ_SUPPORT	<p>It is used to indicate whether an interface supports CMOs on the read channels. By default, the value of this parameter is 1 before G version</p> <p> CDNS_AMBA5_G parameter should be set to 1, to disable this feature.</p>
CMO_ON_WRITE_SUPPORT	<p>It is used to indicate whether an interface supports CMOs on the write channels. By default, the value of this parameter is 1 in G version</p> <p> CDNS_AMBA5_G parameter should be set to 1, to enable the cmo on write feature</p>
CDNS_AMBA5_H	<p>This parameter enables the AMBA5-H features.</p> <p>The default value of this parameter is 0.</p>

CONFIG_REGULAR_TRANSACTION	<p>When set to 1, it enables the Regular Transactions feature.</p> <p>The default value of this parameter is 0.</p> <p>CDNS_AMBA5_H parameter should be set to 1, to enable the properties of the regular transaction.</p>
CONFIG_CONSISTENT_DECERR	<p>when set to 1, it enables DECERR to be signaled for every beat of read data, or no beats of read data within each cache line of data. Default value is 0</p>
CONFIG_SHAREABLE_TRANSACTION	<p>parameter when set to 1 defines whether a master issues Inner Shareable or Outer Shareable transactions and whether a slave supports them. Default value is 1</p>
MAX_TRANSACTION_BYTES	<p>defines the maximum size of a transaction in bytes. Default value is 4096</p>
USER_REQ_WIDTH	<p>new width of REQ User defined signals. Default value is 32</p> <div>  CDNS_AMBA5_H parameter should be set to 1, to enable the properties of the regular transaction. </div>
USER_DATA_WIDTH	<p>new width of DATA User defined signals. Default value is 32</p> <div>  CDNS_AMBA5_H parameter should be set to 1, to enable the properties of the regular transactions. </div>
USER_RESP_WIDTH	<p>new width of RESP User defined signals. Default value is 32</p> <div>  CDNS_AMBA5_H parameter should be set to 1, to enable the properties of the regular transactions. </div>
LOOP_R_WIDTH	<p>define the width of Read address loop signals. Default value are 8</p> <div>  CDNS_AMBA5_H parameter should be set to 1, to enable the properties of the regular transactions. </div>
LOOP_W_WIDTH	<p>define the width of Write address loop signals. Default value are 8</p> <div>  CDNS_AMBA5_H parameter should be set to 1, to enable the properties of the regular transaction. </div>

CONFIG_MTE_SUPPORT	Define the Support of Memory tagging. Set to 1: Value of MTE is Standard, Set to 2: value of MTE is Basic <div> CDNS_AMBA5_H parameter should be set to 1, to enable the properties of the regular transactions.</div>
TAGUPDATE_WIDTH	Define the width of wtagupdate signal,value depends on Data_width
TAG_WIDTH	Define the width of tag signal in read and write channel, value depends on Data_width
TAGCHK_WIDTH	Define the width of tagchk parity signal

ACE5 Checks

Compliance Checks

[Table C-6](#) describes the SVA compliance checks for the master and slave in the ACE5 ABVIP. The properties have been grouped into sub-categories based on different aspects of the ACE5 protocol. The property naming conventions are also shown in the table.

To view AXI5 monitor compliance checks, refer to [AXI5 Monitor Compliance Checks](#). To view additional ACE5 checks, refer to [Table C-6](#).

Table C-6 ACE5 ABVIP Compliance Checks

Property Name	Description	AXI Spec	Applicable for ACE-LITE
1.1 X-Checks			
master_xcheck_ardomain	No AXI 4 signal should have X or Z value when the corresponding valid signal is asserted.	ARM IHI 0022D, Section A3.2.2 on pg A3-38	Yes
master_xcheck_arsnoop	No AXI 4 signal should have X or Z value when the corresponding valid signal is asserted.	ARM IHI 0022D, Section A3.2.2 on pg A3-38	Yes
master_xcheck_arbar	No AXI 4 signal should have X or Z value when the corresponding valid signal is asserted.	ARM IHI 0022D, Section A3.2.2 on pg A3-38	Yes

AXI ABVIP User Guide
Appendix E: ACE5 ABVIP Package Contents--ACE5 Checks

master_xcheck_rack	No AXI 4 signal should have X or Z value when the corresponding valid signal is asserted.	ARM IHI 0022D, Section A3.2.2 on pg A3-38	No
master_xcheck_awdomain	No AXI 4 signal should have X or Z value when the corresponding valid signal is asserted.	ARM IHI 0022D, Section A3.2.2 on pg A3-38	Yes
master_xcheck_awsnoop	No AXI 4 signal should have X or Z value when the corresponding valid signal is asserted.	ARM IHI 0022D, Section A3.2.2 on pg A3-38	Yes
master_xcheck_awbar	No AXI 4 signal should have X or Z value when the corresponding valid signal is asserted.	ARM IHI 0022D, Section A3.2.2 on pg A3-38	Yes
master_xcheck_awcmo	No ACE5 signal should have X or Z value when the corresponding valid signal is asserted.	ARM IHI 0022G	Yes
master_xcheck_wack	No AXI 4 signal should have X or Z value when the corresponding valid signal is asserted.	ARM IHI 0022D, Section A3.2.2 on pg A3-38	No
master_xcheck_acready	No AXI 4 signal should have X or Z value when the corresponding valid signal is asserted.	ARM IHI 0022D, Section A3.2.2 on pg A3-38	Yes
master_xcheck_crvalid	No AXI 4 signal should have X or Z value when the corresponding valid signal is asserted.	ARM IHI 0022D, Section A3.2.2 on pg A3-38	Yes
master_xcheck_crresp	No AXI 4 signal should have X or Z value when the corresponding valid signal is asserted.	ARM IHI 0022D, Section A3.2.2 on pg A3-38	Yes
master_xcheck_cdvalid	No AXI 4 signal should have X or Z value when the corresponding valid signal is asserted.	ARM IHI 0022D, Section A3.2.2 on pg A3-38	No
master_xcheck_cddata	No AXI 4 signal should have X or Z value when the corresponding valid signal is asserted.	ARM IHI 0022D, Section A3.2.2 on pg A3-38	No

AXI ABVIP User Guide
Appendix E: ACE5 ABVIP Package Contents--ACE5 Checks

master_xcheck_cdlast	No AXI 4 signal should have X or Z value when the corresponding valid signal is asserted.	ARM IHI 0022D, Section A3.2.2 on pg A3-38	No
slave_xcheck_acvalid	No AXI 4 signal should have X or Z value when the corresponding valid signal is asserted.	ARM IHI 0022D, Section A3.2.2 on pg A3-38	Yes
slave_xcheck_acaddr	No AXI 4 signal should have X or Z value when the corresponding valid signal is asserted.	ARM IHI 0022D, Section A3.2.2 on pg A3-38	Yes
slave_xcheck_acsnoop	No AXI 4 signal should have X or Z value when the corresponding valid signal is asserted.	ARM IHI 0022D, Section A3.2.2 on pg A3-38	Yes
slave_xcheck_acprot	No AXI 4 signal should have X or Z value when the corresponding valid signal is asserted.	ARM IHI 0022D, Section A3.2.2 on pg A3-38	Yes
slave_xcheck_crready	No AXI 4 signal should have X or Z value when the corresponding valid signal is asserted.	ARM IHI 0022D, Section A3.2.2 on pg A3-38	Yes
slave_xcheck_cdready	No AXI 4 signal should have X or Z value when the corresponding valid signal is asserted.	ARM IHI 0022D, Section A3.2.2 on pg A3-38	No
master_xcheck_arvmidext	arvmidext should not have X or Z value when arvalid is asserted	ARM IHI 0022F: Section E2.11 on page E2-368	No
slave_xcheck_acvmidext	acvmidext should not have X or Z value when acvalid is asserted	ARM IHI 0022F: Section E2.11 on page E2-368	No
master_xcheck_arnsaid	arnsaid should not have X or Z value when arvalid is asserted	ARM IHI 0022F: Section E2.13 on pg E2-374	Yes

AXI ABVIP User Guide
Appendix E: ACE5 ABVIP Package Contents--ACE5 Checks

master_xcheck_arnsaid	awnsaid should not have X or Z value when arvalid is asserted	ARM IHI 0022F: Section E2.13 on pg E2-374	Yes
master_xcheck_crnsaid	crnsaid should not have X or Z value when crvalid is asserted	ARM IHI 0022F: Section E2.13 on pg E2-374	Yes
master_xcheck_cdpoison	cdpoison should not have X or Z value when cdvalid is asserted	ARM IHI 0022F: Section E2.5.2 on pg E2-352	No
master_xcheck_cddatachk	cddatachk should not have X or Z value when cdvalid is asserted	ARM IHI 0022F: Section E2.5.2 on pg E2-352	No
master_xcheck_cdtrace	cdtrace should not have X or Z value when cdvalid is asserted	ARM IHI0022F section E2.6 on page E2-355	No
master_xcheck_crtrace	crtrace should not have X or Z value when crvalid is asserted	ARM IHI0022F section E2.6 on page E2-355	No
slave_xcheck_actrace	actrace should not have X or Z value when acvalid is asserted	ARM IHI0022F section E2.6 on page E2-355	No
master_xcheck_arpam	arpam should not have X or Z value when arvalid is asserted	ARM IHI 0022G: Section E1.17.2 on pg 384	Yes
master_xcheck_aridunq	aridunq should not have X or Z value when arvalid is asserted	ARM IHI 0022G: Section E1.17.2 on pg 384	Yes
slave_xcheck_ridunq	ridunq should not have X or Z value when rvalid is asserted	ARM IHI 0022G: Section E1.17.2 on pg 384	Yes

AXI ABVIP User Guide
Appendix E: ACE5 ABVIP Package Contents--ACE5 Checks

master_xcheck_archunken	archunken should not have X or Z value when arvalid is asserted	ARM IHI 0022G: Section E1.14 on pg 376	Yes
slave_xcheck_rchunkv	rchunkv should not have X or Z value when rvalid is asserted	ARM IHI 0022G: Section E1.14 on pg 376	Yes
slave_xcheck_rchunknum	rchunknum should not have X or Z value when rvalid is asserted	ARM IHI 0022G: Section E1.14 on pg 376	Yes
slave_xcheck_rchunkstrb	rchunkstrb should not have X or Z value when rvalid is asserted	ARM IHI 0022G: Section E1.14 on pg 376	Yes
master_xcheck_awmpam	awmpam should not have X or Z value when awvalid is asserted	ARM IHI 0022G: Section E1.14 on pg 376	Yes
master_xcheck_awidunq	awidunq should not have X or Z value when awvalid is asserted	ARM IHI 0022G: Section E1.17.2 on pg 384	Yes
slave_xcheck_bidunq	bidunq should not have X or Z value when bvalid is asserted	ARM IHI 0022G: Section E1.17.2 on pg 384	Yes
master_xcheck_araddrchk	araddrchk should not have X or Z value when arvalid is asserted	ARM IHI 0022G on pg E2.4-389	Yes
master_xcheck_aridchk	aridchk should not have X or Z value when arvalid is asserted	ARM IHI 0022G on pg E2.4-389	Yes
master_xcheck_arlenchk	arlenchk should not have X or Z value when arvalid is asserted	ARM IHI 0022G on pg E2.4-389	Yes
master_xcheck_arctlchk0	araddrchk0 should not have X or Z value when arvalid is asserted	ARM IHI 0022G on pg E2.4-389	Yes
master_xcheck_arctlchk1	arctlchk1 should not have X or Z value when arvalid is asserted	ARM IHI 0022G on pg E2.4-389	Yes

AXI ABVIP User Guide
Appendix E: ACE5 ABVIP Package Contents--ACE5 Checks

master_xcheck_arctlchk2	arctlchk2 should not have X or Z value when arvalid is asserted	ARM IHI 0022G on pg E2.4-389	Yes
master_xcheck_arctlchk3	arctlchk3 should not have X or Z value when arvalid is asserted	ARM IHI 0022G on pg E2.4-389	Yes
master_xcheck_artracechk	artracechk should not have X or Z value when arvalid is asserted	ARM IHI 0022G on pg E2.4-389	Yes
master_xcheck_arloopchk	arloopchk should not have X or Z value when arvalid is asserted	ARM IHI 0022G on pg E2.4-389	Yes
master_xcheck_arnsaidchk	arnsaidchk should not have X or Z value when arvalid is asserted	ARM IHI 0022G on pg E2.4-389	Yes
master_xcheck_armpamchk	armpamchk should not have X or Z value when arvalid is asserted	ARM IHI 0022G on pg E2.4-389	Yes
master_xcheck_armmuchk	armmuchk should not have X or Z value when arvalid is asserted	ARM IHI 0022G on pg E2.4-389	No
master_xcheck_armmusidchk	armmusidchk should not have X or Z value when arvalid is asserted	ARM IHI 0022G on pg E2.4-389	No
master_xcheck_armmuSSIDchk	armmuSSIDchk should not have X or Z value when arvalid is asserted	ARM IHI 0022G on pg E2.4-389	No
master_xcheck_aruserchk	aruserchk should not have X or Z value when arvalid is asserted	ARM IHI 0022G on pg E2.4-389	Yes
master_xcheck_awaddrchk	awaddrchk should not have X or Z value when awvalid is asserted	ARM IHI 0022G on pg E2.4-389	Yes
master_xcheck_awidchk	awidchk should not have X or Z value when awvalid is asserted	ARM IHI 0022G on pg E2.4-389	Yes
master_xcheck_awlenchk	awlenchk should not have X or Z value when awvalid is asserted	ARM IHI 0022G on pg E2.4-389	Yes

AXI ABVIP User Guide
Appendix E: ACE5 ABVIP Package Contents--ACE5 Checks

master_xcheck_awctlchk0	awctlchk0 should not have X or Z value when awvalid is asserted	ARM IHI 0022G on pg E2.4-389	Yes
master_xcheck_awctlchk1	awctlchk1 should not have X or Z value when awvalid is asserted	ARM IHI 0022G on pg E2.4-389	Yes
master_xcheck_awctlchk2	awctlchk2 should not have X or Z value when awvalid is asserted	ARM IHI 0022G on pg E2.4-389	Yes
master_xcheck_awctlchk3	awctlchk3 should not have X or Z value when awvalid is asserted	ARM IHI 0022G on pg E2.4-389	Yes
master_xcheck_awtracechk	awtracechk should not have X or Z value when awvalid is asserted	ARM IHI 0022G on pg E2.4-389	Yes
master_xcheck_awloopchk	awloopchk should not have X or Z value when awvalid is asserted	ARM IHI 0022G on pg E2.4-389	Yes
master_xcheck_awuserchk	awuserchk should not have X or Z value when awvalid is asserted	ARM IHI 0022G on pg E2.4-389	Yes
master_xcheck_awstashnidchk	awstashnidchk should not have X or Z value when awvalid is asserted	ARM IHI 0022G on pg E2.4-389	Yes
master_xcheck_awstashlpidchk	awstashlpidchk should not have X or Z value when awvalid is asserted	ARM IHI 0022G on pg E2.4-389	Yes
master_xcheck_awnsaidchk	awnsaidchk should not have X or Z value when awvalid is asserted	ARM IHI 0022G on pg E2.4-389	Yes
master_xcheck_awmmuchk	awmmuchk should not have X or Z value when awvalid is asserted	ARM IHI 0022G on pg E2.4-389	No
master_xcheck_awmmusidchk	awmmusidchk should not have X or Z value when awvalid is asserted	ARM IHI 0022G on pg E2.4-389	No
master_xcheck_awmussidchk	awmussidchk should not have X or Z value when awvalid is asserted	ARM IHI 0022G on pg E2.4-389	No

AXI ABVIP User Guide
Appendix E: ACE5 ABVIP Package Contents--ACE5 Checks

master_xcheck_wtracechk	wtracechk should not have X or Z value when wvalid is asserted	ARM IHI 0022G on pg E2.4-389	Yes
master_xcheck_wpoisonchk	wpoisonchk should not have X or Z value when wvalid is asserted	ARM IHI 0022G on pg E2.4-389	Yes
master_xcheck_wstrbchk	wstrbchk should not have X or Z value when wvalid is asserted	ARM IHI 0022G on pg E2.4-389	Yes
master_xcheck_awmpamchk	awmpamchk should not have X or Z value when awvalid is asserted	ARM IHI 0022G on pg E2.4-389	Yes
master_xcheck_wdatachk	wdatachk should not have X or Z value when wvalid is asserted	ARM IHI 0022G on pg E2.4-389	Yes
master_xcheck_wlastchk	wlastchk should not have X or Z value when wvalid is asserted	ARM IHI 0022G on pg E2.4-389	Yes
slave_xcheck_ridchk	ridchk should not have X or Z value when rvalid is asserted	ARM IHI 0022G on pg E2.4-389	Yes
slave_xcheck_rrespchk	rrespchk should not have X or Z value when rvalid is asserted	ARM IHI 0022G on pg E2.4-389	Yes
slave_xcheck_rlastchk	rlastchk should not have X or Z value when rvalid is asserted	ARM IHI 0022G on pg E2.4-389	Yes
slave_xcheck_rloopchk	rloopchk should not have X or Z value when rvalid is asserted	ARM IHI 0022G on pg E2.4-389	Yes
slave_xcheck_rtracechk	rtracechk should not have X or Z value when rvalid is asserted	ARM IHI 0022G on pg E2.4-389	Yes
slave_xcheck_ruserchk	ruserchk should not have X or Z value when rvalid is asserted	ARM IHI 0022G on pg E2.4-389	Yes
slave_xcheck_rdatachk	rdatachk should not have X or Z value when rvalid is asserted	ARM IHI 0022G on pg E2.4-389	Yes

AXI ABVIP User Guide
Appendix E: ACE5 ABVIP Package Contents--ACE5 Checks

slave_xcheck_rchunkchk	rchunkchk should not have X or Z value when rvalid is asserted	ARM IHI 0022G on pg E2.4-389	Yes
slave_xcheck_rpoisonchk	rpoisonchk should not have X or Z value when rvalid is asserted	ARM IHI 0022G on pg E2.4-389	Yes
slave_xcheck_bidchk	bidchk should not have X or Z value when bvalid is asserted	ARM IHI 0022G on pg E2.4-389	Yes
slave_xcheck_brespchk	brespchk should not have X or Z value when bvalid is asserted	ARM IHI 0022G on pg E2.4-389	Yes
slave_xcheck_bloopchk	bloopchk should not have X or Z value when bvalid is asserted	ARM IHI 0022G on pg E2.4-389	Yes
slave_xcheck_buserchk	buserchk should not have X or Z value when bvalid is asserted	ARM IHI 0022G on pg E2.4-389	Yes
slave_xcheck_btracechk	btracechk should not have X or Z value when bvalid is asserted	ARM IHI 0022G on pg E2.4-389	Yes
slave_xcheck_bpoisonchk	bpoisonchk should not have X or Z value when bvalid is asserted	ARM IHI 0022G on pg E2.4-389	Yes
master_xcheck_wuserchk	wuserchk should not have X or Z value when wvalid is asserted	ARM IHI 0022G on pg E2.4-389	Yes
slave_xcheck_acaddrchk	acaddrchk should not have X or Z value when acvalid is asserted	ARM IHI 0022G on pg E2.4-389	Yes
slave_xcheck_acctlchk	acctlchk should not have X or Z value when acvalid is asserted	ARM IHI 0022G on pg E2.4-389	Yes
slave_xcheck_actracechk	actracechk should not have X or Z value when acvalid is asserted	ARM IHI 0022G on pg E2.4-389	Yes
slave_xcheck_acvmidextchk	acvmidextchk should not have X or Z value when acvalid is asserted	ARM IHI 0022G on pg E2.4-389	Yes

AXI ABVIP User Guide
Appendix E: ACE5 ABVIP Package Contents--ACE5 Checks

master_xcheck_crrespchk	crrespchk should not have X or Z value when crvalid is asserted	ARM IHI 0022G on pg E2.4-389	Yes
master_xcheck_crnsaidchk	crnsaidchk should not have X or Z value when crvalid is asserted	ARM IHI 0022G on pg E2.4-389	Yes
master_xcheck_crtracechk	crtracechk should not have X or Z value when crvalid is asserted	ARM IHI 0022G on pg E2.4-389	Yes
master_xcheck_cddatachk	cddatachk should not have X or Z value when cdvalid is asserted	ARM IHI 0022G on pg E2.4-389	No
master_xcheck_cdlastchk	cdlastchk should not have X or Z value when cdvalid is asserted	ARM IHI 0022G on pg E2.4-389	No
master_xcheck_cdpoisonchk	cdpoisonchk should not have X or Z value when cdvalid is asserted	ARM IHI 0022G on pg E2.4-389	No
master_xcheck_cdtracechk	cdtracechk should not have X or Z value when cdvalid is asserted	ARM IHI 0022G on pg E2.4-389	No
master_xcheck_awcmo	awcmo should not have X or Z value when awvalid is asserted	ARM IHI 0022G on pg E2.4-389	Yes
slave_xcheck_bcomp	bcomp should not have X or Z value when bvalid is asserted	ARM IHI 0022G on pg E2.4-389	Yes
slave_xcheck_bpersist	bpersist should not have X or Z value when bvalid is asserted	ARM IHI 0022G on pg E2.4-389	Yes
master_xchecks_armmuflow	armmuflow should not have X or Z value when arvalid is asserted	ARM IHI 0022G on pg E2.4-389	No
master_xchecks_awmmuflow	awmmuflow should not have X or Z value when awvalid is asserted	ARM IHI 0022G on pg E2.4-389	No
master_xcheck_artagop	artagop should not have X or Z value when arvalid is asserted	ARM IHI 0022G on pg E2.4-389	Yes

AXI ABVIP User Guide
Appendix E: ACE5 ABVIP Package Contents--ACE5 Checks

master_xcheck_awtagop	awtagop should not have X or Z value when awvalid is asserted	ARM IHI 0022G on pg E2.4-389	Yes
master_xchecks_wtag	wtag should not have X or Z value when wvalid is asserted	ARM IHI 0022G on pg E2.4-389	Yes
master_xchecks_wtagchk	wtagchk should not have X or Z value when wvalid is asserted	ARM IHI 0022G on pg E2.4-389	Yes
master_xchecks_wtagupdate	wtagupdate should not have X or Z value when wvalid is asserted	ARM IHI 0022G on pg E2.4-389	Yes
slave_xcheck_btagmatch	btagmatch should not have X or Z value when bvalid is asserted	ARM IHI 0022G: Section E1.15 on pg 389	Yes
slave_xcheck_bcomp	bcomp should not have X or Z value when bvalid is asserted	ARM IHI 0022G: Section E1.15 on pg 389	Yes
slave_xcheck_rtag	rtag should not have X or Z value when rvalid is asserted	ARM IHI 0022G: Section E1.15 on pg 389	Yes
slave_xcheck_rtagchk	rtagchk should not have X or Z value when rvalid is asserted	ARM IHI 0022G: Section E1.15 on pg 389	Yes
1.2 Stability Checks			
master_ar_ardomain_stable	ardomain must remain stable while arready is low.	ARM IHI 0022D, Section A3.2.1 on pg A3-37	Yes
master_ar_arsnoop_stable	arsnoop must remain stable while arready is low.	ARM IHI 0022D, Section A3.2.1 on pg A3-37	Yes
master_ar_arbar_stable	arbar must remain stable while arready is low.	ARM IHI 0022D, Section A3.2.1 on pg A3-37	Yes

AXI ABVIP User Guide
Appendix E: ACE5 ABVIP Package Contents--ACE5 Checks

master_aw_awdomain_stable	awdomain must remain stable while awready is low.	ARM IHI 0022D, Section A3.2.1 on pg A3-37	Yes
master_aw_awsnoop_stable	awsnoop must remain stable while awready is low.	ARM IHI 0022D, Section A3.2.1 on pg A3-37	Yes
master_aw_awbar_stable	awbar must remain stable while awready is low.	ARM IHI 0022D, Section A3.2.1 on pg A3-37	Yes
master_aw_awcmo_stable	awcmo must remain stable while awready is low.	ARM IHI 0022G	Yes
slave_ac_acvalid_stable	acvalid must remain stable while acready is low.	ARM IHI 0022D, Section C3.6.2 on pg C3-174	Yes
slave_ac_acaddr_stable	acaddr must remain stable while acvalid is high and acready low.	ARM IHI 0022D, Section C3.6.2 on pg C3-174	Yes
slave_ac_acsnoop_stable	acsnoop must remain stable while acvalid is high and acready low.	ARM IHI 0022D, Section C3.6.2 on pg C3-174	Yes
slave_ac_acprot_stable	acprot must remain stable while acvalid is high and acready low.	ARM IHI 0022D, Section C3.6.2 on pg C3-174	Yes
master_cr_crvalid_stable	crvalid must remain stable while cready is low.	ARM IHI 0022D, Section C3.7 on pg C3-176	Yes
master_cr_crresp_stable	crresp must remain stable while crvalid is high and cready low.	ARM IHI 0022D, Section C3.7 on pg C3-176	Yes

AXI ABVIP User Guide
Appendix E: ACE5 ABVIP Package Contents--ACE5 Checks

master_cd_cdvalid_stable	cdvalid must remain stable while cready is low.	ARM IHI 0022D, Section C3.8 on pg C3-180	No
master_cd_cddata_stable	cddata must remain stable while cdvalid is high and cready low.	ARM IHI 0022D, Section C3.8 on pg C3-180	No
master_cd_cdlast_stable	cdlast must remain stable while cdvalid is high and cready low.	ARM IHI 0022D, Section C3.8 on pg C3-180	No
master_ar_arvmidext_stable	arvmidext should remain stable if arvalid and !arready	ARM IHI 0022F: Section E2.11 on page E2-368	No
slave_ac_acvmidext_stable	acvmidext should remain stable if acvalid and !acready	ARM IHI 0022F: Section E2.11 on page E2-368	No
master_ar_arnsaid_stable	arnsaid should remain stable if arvalid and !arready	ARM IHI 0022F: Section E2.13 on pg E2-374	Yes
master_aw_awnsaid_stable	awnsaid should remain stable if awvalid and !awready	ARM IHI 0022F: Section E2.13 on pg E2-374	Yes
master_aw_awcmo_stable	awcmo should remain stable if awvalid and !awready	ARM IHI 0022G: Section E2.6 on pg 391	Yes
master_cr_crnsaid_stable	crnsaid should remain stable if crvalid and !cready	ARM IHI 0022F: Section E2.13 on pg E2-374	No

AXI ABVIP User Guide
Appendix E: ACE5 ABVIP Package Contents--ACE5 Checks

master_cdpoison_stable	cdpoison should remain stable if cdvalid and !cdready	ARM IHI 0022F: Section E2.5.2 on pg E2-352	No
master_cddatachk_stable	cddatachk should remain stable if cdvalid and !cdready	ARM IHI 0022F: Section E2.5.2 on pg E2-352	No
master_cdtrace_stable	cdtrace should remain stable if cdvalid and !cdready	ARM IHI 0022F: Section E2.6 on pg E2-355	No
master_crtrace_stable	crtrace should remain stable if crvalid and !crready	ARM IHI 0022F: Section E2.6 on pg E2-355	No
slave_actrace_stable	actrace should remain stable if acvalid and !acready	ARM IHI 0022F: Section E2.6 on pg E2-355	No
master_ar_arpam_stable	arpam should remain stable if arvalid && !arready	ARM IHI 0022G on pg E1.16-381	Yes
master_archunken_stable	archunken should remain stable if arvalid && !arready	ARM IHI 0022G on pg E1.16-381	Yes
slave_rchunkv_stable	rchunkv should remain stable if rvalid && !rready	ARM IHI 0022G on pg E1.16-381	Yes
slave_rchunknum_stable	rchunknum should remain stable if rvalid && !rready	ARM IHI 0022G on pg E1.16-381	Yes
slave_rchunkstrb_stable	rchunkstrb should remain stable if rvalid && !rready	ARM IHI 0022G on pg E1.16-381	Yes
master_aridunq_stable	aridunq should remain stable if arvalid && !arready	ARM IHI 0022G on pg E1.16-381	Yes

AXI ABVIP User Guide
Appendix E: ACE5 ABVIP Package Contents--ACE5 Checks

slave_ridunq_stable	ridunq should remain stable if rvalid && !rready	ARM IHI 0022G on pg E1.16-381	Yes
master_aw_awmpam_stable	awmpam should remain stable if awvalid && !awready	ARM IHI 0022G on pg E1.16-381	Yes
master_awidunq_stable	awidunq should remain stable if awvalid && !awready	ARM IHI 0022G on pg E1.16-381	Yes
slave_bidunq_stable	bidunq should remain stable if bvalid && !bready	ARM IHI 0022G on pg E1.16-381	Yes
master_ar_araddrchk_stable	araddrchk should remain stable if arvalid && !arready	ARM IHI 0022G on pg E1.16-381	Yes
master_ar_aridchk_stable	aridchk should remain stable if arvalid && !arready	ARM IHI 0022G on pg E1.16-381	Yes
master_ar_artracechk_stable	artracechk should remain stable if arvalid && !arready	ARM IHI 0022G on pg E1.16-381	Yes
master_ar_arloopchk_stable	arloopchk should remain stable if arvalid && !arready	ARM IHI 0022G on pg E1.16-381	Yes
master_ar_armpamchk_stable	armpamchk should remain stable if arvalid && !arready	ARM IHI 0022G on pg E1.16-381	Yes
master_ar_aruserchk_stable	aruserchk should remain stable if arvalid && !arready	ARM IHI 0022G on pg E1.16-381	Yes
master_ar_arlenchk_stable	arlenchk should remain stable if arvalid && !arready	ARM IHI 0022G on pg E1.16-381	Yes
master_ar_arctlchk1_stable	arctlchk0 should remain stable if arvalid && !arready	ARM IHI 0022G on pg E1.16-381	Yes
master_ar_arctlchk0_stable	arctlchk1 should remain stable if arvalid && !arready	ARM IHI 0022G on pg E1.16-381	Yes

AXI ABVIP User Guide
Appendix E: ACE5 ABVIP Package Contents--ACE5 Checks

master_ar_arctlchk2_stable	arctlchk2 should remain stable if arvalid && !arready	ARM IHI 0022G on pg E1.16- 381	Yes
master_ar_arctlchk3_stable	arctlchk3 should remain stable if arvalid && !arready	ARM IHI 0022G on pg E1.16- 381	Yes
master_ar_arnsaidchk_stable	arnsaidchk should remain stable if arvalid && !arready	ARM IHI 0022G on pg E1.16- 381	Yes
master_ar_armmuchk_stable	armmuchk should remain stable if arvalid && !arready	ARM IHI 0022G on pg E1.16- 381	No
master_ar_armmusidchk_stable	armmusidchk should remain stable if arvalid && !arready	ARM IHI 0022G on pg E1.16- 381	No
master_ar_armmussidchk_stable	armmussidchk should remain stable if arvalid && !arready	ARM IHI 0022G on pg E1.16- 381	No
master_aw_awaddrchk_stable	awaddrchk should remain stable if awvalid && !awready	ARM IHI 0022G on pg E1.16- 381	Yes
master_aw_awidchk_stable	awidchk should remain stable if awvalid && !awready	ARM IHI 0022G on pg E1.16- 381	Yes
master_aw_awlenchk_stable	awlenchk should remain stable if awvalid && !awready	ARM IHI 0022G on pg E1.16- 381	Yes
master_aw_awtracechk_stable	awtracechk should remain stable if awvalid && !awready	ARM IHI 0022G on pg E1.16- 381	Yes
master_aw_awloopchk_stable	awloopchk should remain stable if awvalid && !awready	ARM IHI 0022G on pg E1.16- 381	Yes
master_aw_awmpamchk_stable	awmpamchk should remain stable if awvalid && !awready	ARM IHI 0022G on pg E1.16- 381	Yes
master_aw_awuserchk_stable	awuserchk should remain stable if awvalid && !awready	ARM IHI 0022G on pg E1.16- 381	Yes

AXI ABVIP User Guide
Appendix E: ACE5 ABVIP Package Contents--ACE5 Checks

master_aw_awnsaidchk_stable	awnsaidchk should remain stable if awvalid && !awready	ARM IHI 0022G on pg E1.16-381	Yes
master_aw_awmmuchk_stable	awmmuchk should remain stable if awvalid && !awready	ARM IHI 0022G on pg E1.16-381	No
master_aw_awmmusidchk_stable	awmmusidchk should remain stable if awvalid && !awready	ARM IHI 0022G on pg E1.16-381	No
master_aw_awmmussidchk_stable	awmmussidchk should remain stable if awvalid && !awready	ARM IHI 0022G on pg E1.16-381	No
master_aw_awctlchk0_stable	awctlchk0 should remain stable if awvalid && !awready	ARM IHI 0022G on pg E1.16-381	Yes
master_aw_awctlchk1_stable	awctlchk1 should remain stable if awvalid && !awready	ARM IHI 0022G on pg E1.16-381	Yes
master_aw_awctlchk2_stable	awctlchk2 should remain stable if awvalid && !awready	ARM IHI 0022G on pg E1.16-381	Yes
master_aw_awctlchk3_stable	awctlchk3 should remain stable if awvalid && !awready	ARM IHI 0022G on pg E1.16-381	Yes
master_w_wdatachk_stable	wdatachk should remain stable if wvalid && !wready	ARM IHI 0022G on pg E1.16-381	Yes
master_w_wlastchk_stable	wlastchk should remain stable if wvalid && !wready	ARM IHI 0022G on pg E1.16-381	Yes
master_w_wstrbchk_stable	wstrbchk should remain stable if wvalid && !wready	ARM IHI 0022G on pg E1.16-381	Yes
master_w_wuserchk_stable	wuserchk should remain stable if wvalid && !wready	ARM IHI 0022G on pg E1.16-381	Yes
master_w_wpoisonchk_stable	wpoisonchk should remain stable if wvalid && !wready	ARM IHI 0022G on pg E1.16-381	Yes

AXI ABVIP User Guide
Appendix E: ACE5 ABVIP Package Contents--ACE5 Checks

master_w_wtracechk_stable	wtracechk should remain stable if wvalid && !wready	ARM IHI 0022G on pg E1.16-381	Yes
slave_acctlchk_stable	acctlchk should remain stable if acvalid && !acready	ARM IHI 0022G on pg E1.16-381	Yes
slave_actracechk_stable	actracechk should remain stable if acvalid && !acready	ARM IHI 0022G on pg E1.16-381	Yes
slave_ac_acaddrchk_stable	acaddrchk should remain stable if acvalid && !acready	ARM IHI 0022G on pg E1.16-381	Yes
slave_acvmidextchk_stable	acvmidextchk should remain stable if acvalid && !acready	ARM IHI 0022G on pg E1.16-381	Yes
master_cd_cddatachk_stable	cddatachk should remain stable if cdvalid && !cdready	ARM IHI 0022G on pg E1.16-381	No
master_cd_cdlastchk_stable	cdlastchk should remain stable if cdvalid && !cdready	ARM IHI 0022G on pg E1.16-381	No
master_cd_cdpoisonchk_stable	cdpoisonchk should remain stable if cdvalid && !cdready	ARM IHI 0022G on pg E1.16-381	No
master_cd_cdtracechk_stable	cdtracechk should remain stable if cdvalid && !cdready	ARM IHI 0022G on pg E1.16-381	No
master_cr_crrespchk_stable	crrespchk should remain stable if crvalid && !crready	ARM IHI 0022G on pg E1.16-381	Yes
master_cr_crtracepchk_stable	crtracechk should remain stable if crvalid && !crready	ARM IHI 0022G on pg E1.16-381	Yes
master_cr_crnsaidchk_stable	crnsaidchk should remain stable if crvalid && !crready	ARM IHI 0022G on pg E1.16-381	Yes
slave_b_bcomp_stable	bcomp should remain stable if bvalid && !bready	ARM IHI 0022G on pg E1.16-381	Yes

AXI ABVIP User Guide
Appendix E: ACE5 ABVIP Package Contents--ACE5 Checks

slave_b_bpersist_stable	bpersist should remain stable if bvalid && !bready	ARM IHI 0022G on pg E1.16-381	Yes
master_aw_awmmuflow_stable	awmmuflow should remain stable if awvalid and !awready	ARM IHI 0022H: Section E2.6 on pg 391	Yes
master_ar_armmuflow_stable	armmuflow should remain stable if arvalid and !arready	ARM IHI 0022H: Section E2.6 on pg 391	Yes
master_aw_awtagop_stable	awtagop should remain stable if awvalid and !awready	ARM IHI 0022H: Section E2.6 on pg 391	Yes
master_ar_artagop_stable	artagop should remain stable if arvalid and !arready	ARM IHI 0022H: Section E2.6 on pg 391	Yes
master_w_wtag_stable	wtag should remain stable if wvalid and !wready	ARM IHI 0022H: Section E2.6 on pg 391	Yes
master_w_wtagchk_stable	wtagchk should remain stable if wvalid and !wready	ARM IHI 0022H: Section E2.6 on pg 391	Yes
master_w_wtagupdate_stable	wtagupdate should remain stable if wvalid and !wready	ARM IHI 0022H: Section E2.6 on pg 391	Yes
slave_b_btagmatch_stable	btagmatch should remain stable if bvalid && !bready	ARM IHI 0022G on pg E2.4-389	Yes
slave_b_bcomp_stable	bcomp should remain stable if bvalid && !bready	ARM IHI 0022G on pg E2.4-389	Yes
slave_r_rtag_stable	rtag should remain stable if rvalid && !rready	ARM IHI 0022G on pg E2.4-389	Yes
slave_r_rtagchk_stable	rtagchk should remain stable if rvalid && !rready	ARM IHI 0022G on pg E2.4-389	Yes

AXI ABVIP User Guide
Appendix E: ACE5 ABVIP Package Contents--ACE5 Checks

1.3 Reset Checks

<code>master_rst_crvalid_low_reset</code>	Master should drive crvalid low during reset.	ARM IHI 0022D, Section A3.1.2 on pg A3-36	Yes
<code>master_rst_crvalid_low_reset_released</code>	Master should not drive crvalid at a posedge of aclk after aresetn is high.	ARM IHI 0022D, Section A3.1.2 on pg A3-36	Yes
<code>master_rst_cdvalid_low_reset</code>	Master should drive cdvalid low during reset.	ARM IHI 0022D, Section A3.1.2 on pg A3-36	No
<code>master_rst_cdvalid_low_reset_released</code>	Master should not drive cdvalid at a posedge of aclk after aresetn is high.	ARM IHI 0022D, Section A3.1.2 on pg A3-36	No
<code>slave_rst_acvalid_low_reset</code>	Slave should drive acvalid low during reset.	ARM IHI 0022D, Section A3.1.2 on pg A3-36	Yes
<code>slave_rst_acvalid_low_reset_released</code>	Slave should not drive acvalid at a posedge of aclk after aresetn is high	ARM IHI 0022D, Section A3.1.2 on pg A3-36	Yes
<code>slave_rst_varqosaccept</code>	The default value of VARQOSACCEPT signal is 0	ARM IHI0022F: Section E2.8 on pg E2-359	Yes
<code>slave_rst_vawqosaccept</code>	The default value of VARQOSACCEPT signal is 0	ARM IHI0022F: Section E2.8 on pg E2-359	Yes

1.4 Read Checks

<code>master_ar_arsize_for_coherent</code>	arsize must be equal to data bus width for Cache Line Size, Barrier and DVM Transactions, arsize must be equal to or less than data bus width for ReadOnce Transactions	ARM IHI 0022D, Section C3.1.5 on pg C3-163, C3-166, section C12.6 on C12-287	Yes
--	---	--	-----

AXI ABVIP User Guide
Appendix E: ACE5 ABVIP Package Contents--ACE5 Checks

master_ar_arlen_less_than_TRANS_COUNT	ARLEN should be less than the MAXLEN value configured	None	Yes
master_ar_arsize_arlen	Transaction size must be equal to cache line size for Cache Line Size Transactions	ARM IHI 0022D, Section C3.1.5 on pg C3-163	Yes
master_ar_arlen_for_coherent	arlen must be less than 16 for Cache Line Size Transactions arlen must be all zeros for Barrier/DVM Transactions	ARM IHI 0022D, Section C3.1.5 on pg C3-163, C3-166, section C12.6 on C12-287	Yes
master_ar_arburst_for_coherent	arburst must be INCR for all Barrier/DVM Transactions arburst must be INCR or WRAP for all other coherent Transactions	ARM IHI 0022D, Section C3.1.5 on pg C3-163, C3-166, section C12.6 on C12-287	Yes
master_ar_araddr_for_barrier	araddr must be all zeros for Barrier	ARM IHI 0022D, Section C3.1.5 on pg C3-166	Yes
master_ar_araddr_for_cachelinesize	When ARBURST is INCR, the address must be aligned to the cache line size	ARM IHI 0022D, Section C3.1.5 on pg C3-163	Yes
master_ar_arcache_for_coherent	arcache must be normal non-cacheable (AxCACHE=0010) for Barrier/DVM Transactions. arcache must be modifiable (AxCACHE[1]=1) for all other coherent Transactions	ARM IHI 0022D, Section C3.1.5 on pg C3-163 - C3-166, section C12.6 on C12-287	Yes

AXI ABVIP User Guide
Appendix E: ACE5 ABVIP Package Contents--ACE5 Checks

master_ar_arlock_for_coherent	arlock must be normal access (AxLOCK=0) for all coherent Read Transactions except ReadClean, ReadShared, CleanUnique.	ARM IHI 0022D, Section C3.1.5 on pg C3-163 - C3-166, section C12.6 on C12-287	Yes
master_ar_ardomain_by_arcache	Device transactions, as indicated by AxCACHE[1] = 0, must only use AxDOMAIN = 11, System. Cacheable transactions, as indicated by AxCACHE[3:2] != 00, must not use AxDOMAIN = 11, System.	ARM IHI 0022D, Section C3.1.1 on C3-158	Yes
master_ar_ardomain_for_coherent	ardomain must be INNER, OUTER for coherent and dvm transaction groups. ardomain must be INNER, OUTER, NON for cache maintenance transaction group	ARM IHI 0022D, Section C3.1.3 on pg C3-161, section C3.1.5 on pg C3-163, C3-164, C12.6 on pg C12-287	Yes
master_ar_arbar_for_coherent	arbar must be normal access for all Transactions except Barrier	ARM IHI 0022D, Section C3.1.5 on pg C3-163 - C3-166	Yes
master_ar_arsnoop_legal_values	Legal Read address channel combinations of arsnop, arbar, ardomain. All unused encodings are reserved.	ARM IHI 0022D, Section C3.1.3 on pg C3-161	Yes
master_ar_arsnoop_for_barrier	All barrier transactions, must have ARSNOOP[3:0] = 0000 arsnop must be all zeroes for Barrier Transactions	ARM IHI 0022D, Section C3.1.3 on pg C3-161, section C3.1.5 on pg C3-166	Yes
master_ar_arbar_for_dvm	All DVM transactions, arbar should have the value of 2'b00	ARM IHI 0022D, Section C12.6 on pg C12-287	No

AXI ABVIP User Guide
Appendix E: ACE5 ABVIP Package Contents--ACE5 Checks

master_ar_aw_no_overlapping_addr	Current read request address should not overlap with any of the outstanding write requests and current write request(if any).	None	Yes
master_ar_r_arcache_no_read_while_maintenance_scl	The master must not issue any further shareable transactions to the same cache line until the cache maintenance transaction is complete.	ARM IHI 0022D, Section C7.2.1 on pg C7-243	Yes
slave_r_ar_rresp	ReadNoSnoop, ReadUnique, CleanUnique, MakeUnique, CleanInvalid, and MakeInvalid transaction must have a response with IsShared RRESP[3] de-asserted (IsUnique). ReadNoSnoop, ReadOnce, ReadClean, CleanUnique, MakeUnique, CleanShared, CleanInvalid, and MakeInvalid transactions must have a response with PassDirty RRESP[2] de-asserted (PassClean). A barrier transaction and a DVM transaction response must use RRESP[3:2] = "00". A ReadNotSharedDirty transaction response must not use RRESP[3:2] = "11".	ARM IHI 0022D, Section C3.2.1 on pg C3-167, C3-168	No
master_r_rack_eventually	All pending read transactions will be acknowledged eventually.	None	No
master_r_rack_wait_cycles	All pending read transactions will be acknowledged within MAX_WAIT_CYCLES_RACK cycles	None	No
master_ar_deallocation_trn_cacheline_boundary	Deallocating transaction must not cross cache line boundary	ARM IHI 0022F section E2.3 on pg E2-349	Yes
master_ar_ardomain_inner_shareable_dep	Master must not drive ardomain value 01, as it is deprecated in AMBA5 Spec	ARM IHI 0022F Section E1.1 on pg E1-332	Yes
master_ar_arnsaid_valid	The master must arnsaid 0, for secure transactions.	ARM IHI 0022F: Section E2.13 on pg E2-375	Yes
master_ar_width_check	AWID, ARID should be of same width for read and write channels for exclusive transactions	None	Yes
master_ar_mpam_combination	If NS is 0, MPAN_NS cannot be 1, it is illegal combination for read address channel	ARM IHI 0022G: section 6.4.2	Yes
master_aridunq_no_outstanding_rd_trans	When ARIDUNQ is asserted, there must be no outstanding read transactions from this master with the same ARID value	ARM IHI 0022G on pg E1.16-381	Yes

AXI ABVIP User Guide
Appendix E: ACE5 ABVIP Package Contents--ACE5 Checks

master_no_rd_req_with_same_arid	A master must not issue a read request with the same ARID as an outstanding read transaction that had ARIDUNQ asserted	ARM IHI 0022G on pg E1.16-381	Yes
master_aridunq_ridunq_match	If ARIDUNQ is deasserted or asserted for a request, the corresponding RIDUNQ signals must be deasserted or asserted for all response beats for that transaction	ARM IHI 0022G on pg E1.16-381	Yes
master_ar_arvalidchk_gen	arvalidchk should be generated whenever aresetn is asserted	ARM IHI 0022G on pg E2.6-392	Yes
master_ar_arreadychk_gen	arreadychk should be generated whenever aresetn is asserted	ARM IHI 0022G on pg E2.6-392	Yes
master_ar_aridchk_gen	aridchk should be generated whenever arvalid is asserted	ARM IHI 0022G on pg E2.6-392	Yes
master_ar_araddrchk_gen	araddrchk should be generated whenever arvalid is asserted	ARM IHI 0022G on pg E2.6-392	Yes
master_ar_arlenchk_gen	arlenchk should be generated whenever arvalid is asserted	ARM IHI 0022G on pg E2.6-392	Yes
master_ar_aruserchk_gen	aruserchk should be generated whenever arvalid is asserted	ARM IHI 0022G on pg E2.6-392	Yes
master_ar_arloopchk_gen	arloopchk should be generated whenever arvalid is asserted	ARM IHI 0022G on pg E2.6-392	Yes
master_ar_artracechk_gen	artracechk should be generated whenever arvalid is asserted	ARM IHI 0022G on pg E2.6-392	Yes
master_ar_arctlchk0_gen	arctlchk0 should be generated whenever arvalid is asserted	ARM IHI 0022G on pg E2.6-392	Yes
master_ar_arctlchk1_gen	arctlchk1 should be generated whenever arvalid is asserted	ARM IHI 0022G on pg E2.6-392	Yes
master_ar_arnsaidchk_gen	arnsaidchk should be generated whenever arvalid is asserted	ARM IHI 0022G on pg E2.6-392	Yes

AXI ABVIP User Guide
Appendix E: ACE5 ABVIP Package Contents--ACE5 Checks

master_ar_armmuchk_gen	armmuchk should be generated whenever arvalid is asserted	ARM IHI 0022G on pg E2.6-392	No
master_ar_armmusidchk_gen	armmusidchk should be generated whenever arvalid is asserted	ARM IHI 0022G on pg E2.6-392	No
master_ar_armmussidchk_gen	armmussidchk should be generated whenever arvalid is asserted	ARM IHI 0022G on pg E2.6-392	No
master_ar_arpamchk_gen	arpamchk should be generated whenever arvalid is asserted	ARM IHI 0022G on pg E2.6-392	Yes
master_ar_shareable_tr_ardomain	The must must not send Inner Shareable and Outer Shareable transactions	ARM IHI 0022H: Section E1.18.2 on pg E1-399	Yes
master_ar_trsize_never_cross_max_bytes	Read transaction must not cross maximum transaction bytes	ARM IHI 0022H: Section E1.18.3 on pg E1-400	Yes
master_ar_trsize_never_cross_max_boundary	Read transaction must not cross maximum transaction boundary	ARM IHI 0022H: Section E1.18.3 on pg E1-400	Yes
slave_r_ridchk_gen	ridchk should be generated whenever rvalid is asserted	ARM IHI 0022G on pg E2.6-392	Yes
slave_r_rloopchk_gen	rloopchk should be generated whenever rvalid is asserted	ARM IHI 0022G on pg E2.6-392	Yes
slave_r_rtracechk_gen	rtracechk should be generated whenever rvalid is asserted	ARM IHI 0022G on pg E2.6-392	Yes
slave_r_rpoisonchk_gen	rpoisonchk should be generated whenever rvalid is asserted	ARM IHI 0022G on pg E2.6-392	Yes
slave_r_rlastchk_gen	rlastchk should be generated whenever rvalid is asserted	ARM IHI 0022G on pg E2.6-392	Yes

AXI ABVIP User Guide
Appendix E: ACE5 ABVIP Package Contents--ACE5 Checks

slave_r_rdatachk_gen	rdatachk should be generated whenever rvalid is asserted	ARM IHI 0022G on pg E2.6-392	Yes
slave_r_rrespchk_gen	rrespchk should be generated whenever rvalid is asserted	ARM IHI 0022G on pg E2.6-392	Yes
slave_r_rchunkchk_gen	rchunkchk should be generated whenever rvalid is asserted	ARM IHI 0022G on pg E2.6-392	Yes
slave_r_ruserchk_gen	ruserchk should be generated whenever rvalid is asserted	ARM IHI 0022G on pg E2.6-392	Yes
slave_rdata_recom_inactivebytelines	rdata is driven to zero for inactive byte lanes	ARM IHI0022H on pg A3-42	Yes
slave_r_ar_resp_consistent_decoderr	DECERR is signaled for every beat of read data, or no beats of read data within each cache line of data	ARM IHI 0022H: Section E1.18.4 on pg E1.401	Yes

1.5 Write Control Checks

master_aw_no_evict	An Evict transaction is only used by a master that supports a snoop filter.	ARM IHI 0022D, Section C4.9.1 on pg C4-213	No
master_aw_awunique_for_writeclean	AWUNIQUE must be de-asserted for WriteClean transaction	ARM IHI 0022D, Section C3.1.4 on pg C3-167	No
master_aw_awunique_for_writeevict	AWUNIQUE must be asserted for WriteEvict transaction	ARM IHI 0022D, Section C3.1.4 on pg C3-167	No
master_aw_no_writeevict	No WriteEvict transaction, if WREVICT_ON parameter is off.	None	No
master_ar_r_aw_b_awsnoop_no_maintenance_while_pending_scl	The master must complete any outstanding shareable transactions to the same cache line before issuing the cache maintenance transaction.	ARM IHI 0022D, Section C7.2.1 on pg C7-243	No

AXI ABVIP User Guide
Appendix E: ACE5 ABVIP Package Contents--ACE5 Checks

master_aw_awsiz_for_coherent	awsiz must be equal to data bus width for Cache Line Size and Barrier Transactions awsiz must be equal to or less than data bus width for WriteBack/Clean Transactions	ARM IHI 0022D, Section C3.1.5 on pg C3-163 - C3-166	Yes
master_aw_awsiz_awlen	Transaction size must be equal to cache line size for Cache Line Size Transactions transaction size must be equal to or less than cache line size for WriteBack/Clean Transactions	ARM IHI 0022D, Section C3.1.5 on pg C3-163 - C3-165	Yes
master_aw_awlen_less_than_TRANS_COUNT	AWLEN should be less than the MAXLEN value configure	None	Yes
master_aw_awlen_for_coherent	awlen must be 1, 2, 4, 8, 16 for Cache Line Size Transactions, awlen must be equal to or less than 16 for WriteBack/Clean Transactions, awlen must be all zeros for Barrier/DVM Transactions	ARM IHI 0022D, Section C3.1.5 on pg C3-163, C3-166	Yes
master_aw_awburst_for_coherent	awburst must be INCR for all Barrier/DVM Transactions, awburst must be INCR or WRAP for all other coherent Transactions	ARM IHI 0022D, Section C3.1.5 on pg C3-163, C3-165	Yes
master_aw_awaddr_for_barrier	awaddr must be all zeros for Barrier	ARM IHI 0022D, Section C3.1.5 on pg C3-166	Yes
master_aw_awaddr_for_cachelinesize	When AWBURST is INCR, the address must be aligned to the cache line size for Cache Line Size Transactions	ARM IHI 0022D, Section C3.1.5 on pg C3-163	Yes
master_aw_awaddr_for_wrbkclean	When AWBURST is INCR, the start and end address must be in the same cache line, for WriteBack/Clean Transactions	ARM IHI 0022D, Section C3.1.5 on pg C3-165	No

AXI ABVIP User Guide
Appendix E: ACE5 ABVIP Package Contents--ACE5 Checks

master_aw_awcache_for_coherent	awcache must be normal non-cacheable (AxCACHE=0010) for Barrier/DVM Transactions awcache must be modifiable (AxCACHE[1]=1) for all other coherent Transactions awcache must be modifiable (AxCACHE[1]=1) for all other coherent Transactions	ARM IHI 0022D, Section C3.1.5 on pg C3-163 - C3-166	Yes
master_aw_awlock_for_coherent	awlock must be normal access (AxLOCK=0) for all coherent Write Transactions	ARM IHI 0022D, Section C3.1.5 on pg C3-163 - C3-166	Yes
master_aw_awdomain_by_awcache	Device transactions, as indicated by AxCACHE[1] = 0, must only use AxDOMAIN = 11, System. Cacheable transactions, as indicated by AxCACHE[3:2] != 00, must not use AxDOMAIN = 11, System	ARM IHI 0022D, Section C3.1.1 on C3-158	Yes
master_aw_awdomain_for_coherent	awdomain must be INNER, OUTER for coherent transaction group and Evict awdomain must be INNER, OUTER, NON for memory update transaction group.	ARM IHI 0022D, Section C3.1.3 on pg C3-162, section C3.1.5 on pg C3-165	Yes
master_aw_awbar_for_coherent	awbar must be normal access for all Transactions except Barrier	ARM IHI 0022D, Section C3.1.5 on pg C3-163 - C3-166	Yes
master_aw_awsnoop_legal_values	Legal Write address channel combinations of awsnoop, awbar, awdomain, all unused encodings are reserved	ARM IHI 0022D, Section C3.1.3 on pg C3-162	Yes
master_aw_awsnoop_for_barrier	All barrier transactions, must have AWSNOOP[2:0] = 000	ARM IHI 0022D, Section C3.1.5 on pg C3-166	Yes
master_aw_ar_no_overlapping_addr	Current write request address should not overlap with any one of the outstanding read and write requests	None	Yes

AXI ABVIP User Guide
Appendix E: ACE5 ABVIP Package Contents--ACE5 Checks

master_aw_b_awsnoop_no_wrunique_wrbkclean	Complete any outstanding WriteBack or WriteClean transactions before issuing a WriteUnique or WriteLineUnique transaction	ARM IHI 0022D, Section C4.8.6 on pg C4-205	No
master_aw_b_awsnoop_no_wrbkclean_wrunique	No additional WriteBack or WriteClean transactions can be issued until all outstanding WriteUnique or WriteLineUnique transactions are completed.	ARM IHI 0022D, Section C4.8.6 on pg C4-205	No
master_aw_b_ar_r_awcache_no_write_while_maintenance_scl	The master must not issue any further shareable transactions to the same cache line until the cache maintenance transaction is complete.	ARM IHI 0022D, Section C7.2.1 on pg C7-243	Yes
slave_r_ar_ac_cr_rvalid_no_rresp_before_sn_finished_scl	If the interconnect sends a snoop transaction to a master, it must not provide the same master with a read response to a transaction to the same cache line, until it has received a snoop response on <code>CRRESP</code> to the snoop transaction.	ARM IHI 0022D, Section C7.2.1 on pg C7-243	No
slave_r_ar_ac_cr_rvalid_no_ns_rresp_before_sn_finished_scl	If the interconnect sends a snoop transaction to a master, it must not provide the same master with a non shareable read response to a transaction to the same cache line, until it has received a snoop response on <code>CRRESP</code> to the snoop transaction. To enable this check, set parameter <code>RECM_CHECKS_ON</code> to 1.	None	No
assert_aw_wr_tbl_no_overflow	ACE write Checker should never overflow otherwise the results become invalid to be used as an assertion only	None	Yes
master_aw_wr_tbl_no_overflow_ace	Master should not issue Evict and barrier transaction when write table is full	None	Yes
master_aw_awdomain_inner_shareable_dep	Master must not drive awdomain value 01, as it is deprecated in amba5 protocol	ARM IHI 0022F Section E1.1 on pg E1-332	Yes
master_aw_awnsaid_valid	The master must awnsaid 0, for secure transactions.	ARM IHI 0022F: Section E2.13 on pg E2-375	Yes
master_aw_width_check	AWID, ARID should be of same width for read and write channels for exclusive transactions	None	Yes
master_aw_atomicID_width_check	AWID, ARID should be of same width for read and write channels for atomic transactions	None	Yes

AXI ABVIP User Guide
Appendix E: ACE5 ABVIP Package Contents--ACE5 Checks

master_aw_w_awlen_exact_len_collision	If data phase and control starts simultaneously, the control phase must match that data transfer length.	None	Yes
master_w_aw_wvalid_no_dbc	It ensures that wvalid should only come from master if either its control phase has already done or control phase is coming simultaneously with data phase in non-dbc case (i.e. when DATA_BEFORE_CONTROL_ON is off)	None	Yes
master_w_aw_wlast_exact_len_collision	When data and control phase are coming simultaneously then wlast should only be asserted when awlen matches the data beat count.	None	Yes
master_aw_mpam_combination	If NS is 0, MPAN_NS cannot be 1, it is illegal combination for write address channel.	ARM IHI 0022G: section 6.4.2	Yes
master_awidunq_awid_no_outstanding_wr_trans	When AWIDUNQ is asserted, there must be no outstanding write transactions from this master with the same AWID value	ARM IHI 0022G: section 6.4.2	Yes
master_no_wr_req_with_same_awid	A master must not issue a write request with the same AWID as an outstanding write transaction that had AWIDUNQ asserted	ARM IHI 0022G: section 6.4.2	Yes
master_awidunq_bidunq_match	If AWIDUNQ is de-asserted or asserted for a request, the corresponding BIDUNQ signal must be de-asserted or asserted for all response beats for that transaction	ARM IHI 0022G: section 6.4.2	Yes
master_wr_cl_bk_evict_id_inuse	WriteClean, WriteBack, WriteEvict and Evict transactions uses an ID that is unique in flight	ARM IHI 0022G on pg G5.1 - 454	No
master_aw_awaddrchk_gen	awaddrchk should be generated whenever awvalid is asserted	ARM IHI 0022G on pg E2.6-392	Yes
master_aw_awvalidchk_gen	awvalidchk should be generated whenever aresetn is asserted	ARM IHI 0022G on pg E2.6-392	Yes
master_aw_awreadychk_gen	awreadychk should be generated whenever aresetn is asserted	ARM IHI 0022G on pg E2.6-392	Yes
master_aw_awidchk_gen	awidchk should be generated whenever awvalid is asserted	ARM IHI 0022G on pg E2.6-392	Yes
master_aw_awlenchk_gen	awlenchk should be generated whenever awvalid is asserted	ARM IHI 0022G on pg E2.6-392	Yes

AXI ABVIP User Guide
Appendix E: ACE5 ABVIP Package Contents--ACE5 Checks

master_aw_awloopchk_gen	awloopchk should be generated whenever awvalid is asserted	ARM IHI 0022G on pg E2.6-392	Yes
master_aw_awtracechk_gen	awtracechk should be generated whenever awvalid is asserted	ARM IHI 0022G on pg E2.6-392	Yes
master_aw_awuserchk_gen	awuserchk should be generated whenever awvalid is asserted	ARM IHI 0022G on pg E2.6-392	Yes
master_aw_awctlchk0_gen	awctlchk0 should be generated whenever awvalid is asserted	ARM IHI 0022G on pg E2.6-392	Yes
master_aw_awctlchk1_gen	awctlchk1 should be generated whenever awvalid is asserted	ARM IHI 0022G on pg E2.6-392	Yes
master_aw_awnsaidchk_gen	awnsaidchk should be generated whenever awvalid is asserted	ARM IHI 0022G on pg E2.6-392	Yes
master_aw_awmpamchk_gen	awmpamchk should be generated whenever awvalid is asserted	ARM IHI 0022G on pg E2.6-392	Yes
master_aw_awmmuchk_gen	awmmuchk should be generated whenever awvalid is asserted	ARM IHI 0022G on pg E2.6-392	No
master_aw_awmmusidchk_gen	awmmusidchk should be generated whenever awvalid is asserted	ARM IHI 0022G on pg E2.6-392	No
master_aw_awmussidchk_gen	awmussidchk should be generated whenever awvalid is asserted	ARM IHI 0022G on pg E2.6-392	No
master_aw_awcmo_for_coherent	awcmo must be 2'b00 for all other coherent Transactions	ARM IHI 0022G: Section E1.4.2 on pg E1-350	Yes
master_b_bready_persist_eventually	All bpersist response must be eventually accepted	ARM IHI 0022G: Section E1.5	Yes

AXI ABVIP User Guide
Appendix E: ACE5 ABVIP Package Contents--ACE5 Checks

master_ar_no_read_while_deep_maintenance_scl	The master must not issue any further shareable transactions to the same cache line until the cache maintenance for persistence transaction is complete	ARM IHI 0022D: Section C7.2.1 on pg C7-243	Yes
master_aw_awsnoop_no_persist_maintenance_while_pending_scl	The master must complete any outstanding shareable transactions to the same cache line before issuing the cache maintenance for persistence transaction	ARM IHI 0022D: Section C7.2.1 on pg C7-243	Yes
master_ar_no_write_while_persist_maintenance_scl	The master must not issue any further shareable transactions to the same cache line until the cache maintenance for persistence transaction is complete	ARM IHI 0022D: Section C7.2.1 on pg C7-243	Yes
master_aw_awburst_for_coherent	awburst must be INCR or WRAP for all Cmo Transactions	ARM IHI 0022D: Section C3.1.5 on pg C3-163, C3-166, section C12.6 on C12-287	Yes
master_aw_awcache_for_coherent	awcache must be modifiable (AxCACHE[1]=1) for all CMO Transactions	ARM IHI 0022D: Section C3.1.5 on pg C3-163 - C3-166, section C12.6 on C12-287	Yes
master_aw_awlen_for_coherent	awlen must be less than 16 for Cache Line Size Transactions	ARM IHI 0022D: Section C3.1.5 on pg C3-163, C3-166, section C12.6 on C12-287	Yes
master_aw_awbar_for_coherent	awbar must be normal access for Cache Line Size Transactions	ARM IHI 0022D: Section C3.1.5 on pg C3-163, C3-166, section C12.6 on C12-287	Yes

AXI ABVIP User Guide
Appendix E: ACE5 ABVIP Package Contents--ACE5 Checks

master_aw_awdomain_for_coherent	awdomain must be INNER, OUTER, NON for cache maintenance transaction group	ARM IHI 0022D: Section C3.1.3 on pg C3-161, section C3.1.5 on pg C3-163, C3-164, C12.6 on pg C12-287	Yes
master_aw_awsiz_awlen	Transaction size must be equal to cache line size for Cache Line Size Transactions	ARM IHI 0022D: Section C3.1.5 on pg C3-163	Yes
master_aw_awlock_for_coherent	awlock must be normal access (AxLOCK=0) for all CMO Transactions	ARM IHI 0022D: Section C3.1.5 on pg C3-163 - C3-166, section C12.6 on C12-287	Yes
master_aw_awaddr_for_cachelinesize	When AWBURST is INCR, the address must be aligned to the cache line size	ARM IHI 0022D: Section C3.1.5 on pg C3-163	Yes
master_aw_arsize_for_coherent_cachelinesize	awsiz must be equal to data bus width for Cache Line Size	ARM IHI 0022D: Section C3.1.5 on pg C3-163, C3-166, section C12.6 on C12-287	Yes
master_aw_awsnoop_no_maintenance_while_pending_scl	The master must complete any outstanding shareable transactions to the same cache line before issuing the cache maintenance transaction	ARM IHI 0022D: Section C7.2.1 on pg C7-243	Yes
master_ar_no_write_while_maintenance_scl	The master must not issue any further shareable transactions to the same cache line until the cache maintenance transaction is complete	ARM IHI 0022D: Section C7.2.1 on pg C7-243	Yes

AXI ABVIP User Guide
Appendix E: ACE5 ABVIP Package Contents--ACE5 Checks

master_ar_no_read_while_maintenance_scl	The master must not issue any further shareable transactions to the same cache line until the cache maintenance transaction is complete	ARM IHI 0022D: Section C7.2.1 on pg C7-243	Yes
master_aw_shareable_tr_ardomain	The master must not send Inner Shareable and Outer Shareable transactions	ARM IHI 0022H: Section E1.18.2 on pg E1-399	Yes
master_aw_trsize_never_cross_max_bytes	Write transaction must not cross maximum transaction bytes	ARM IHI 0022H: Section E1.18.3 on pg E1-400	Yes
master_aw_trsize_never_cross_max_boundary	Write transaction must not cross maximum transaction boundary	ARM IHI 0022H: Section E1.18.3 on pg E1-400	Yes
master_w_aw_wstrb_valid_collision	Check strobe for the first beat of a transfer which comes in the same cycle as control for the same transfer.	ARM IHI 0022D, Section A3.4.1 on pg A3.46-A3.47	Yes
master_w_aw_wstrb_valid_curr	Check strobe received in current cycle whose control is already received	ARM IHI 0022D, Section A3.4.1 on pg A3.46-A3.47	Yes
master_w_aw_wstrb_valid_dbc	In case where data beats are received before control, check the strobes of all the previously received beats when control is received.	ARM IHI 0022D, Section A3.4.1 on pg A3.46-A3.47	Yes
master_w_aw_wstrb_valid_curr_dbc	In case where data beats are received before control, check the strobes of the beat received in the same cycle as control.	ARM IHI 0022D, Section A3.4.1 on pg A3.46-A3.47	Yes
master_aw_awsizes_all_bytelanes_active	This assertion checks that the table never overflows. This ensures protocol verification results are valid. To be used as an ASSERTION only.	None	Yes
assert_w_wr_tbl_no_overflow	This assertion checks that the table never overflows. This ensures protocol verification results are valid. To be used as an ASSERTION only.	None	Yes

AXI ABVIP User Guide
Appendix E: ACE5 ABVIP Package Contents--ACE5 Checks

slave_b_aw_ac_cr_bvalid_no_bresp_before_sn_finished_scl	If the interconnect sends a snoop transaction to a master, it must not provide the same master with a response to a transaction to the same cache line, until it has received a snoop response on <code>CRRESP</code> to the snoop transaction (indicated by <code>snoop_in_prg_scl_b</code>).	ARM IHI 0022D, Section C6.2 on pg C6-227	No
slave_b_aw_ac_cr_bvalid_no_wns_bresp_before_sn_finished_scl	If the interconnect sends a snoop transaction to a master, it must not provide the same master with a response to a transaction to the same cache line, until it has received a snoop response on <code>CRRESP</code> to the snoop transaction. Set <code>RECM_CHECKS_ON</code> to enable this property.	None	No
master_b_wack_only_after_write	The WACK signal is asserted by a master to indicate that it has completed a write transaction. WACK is asserted for a single cycle and the interconnect is required to accept the WACK signal in a single cycle. The WACK signal must be asserted the cycle after the associated handshake or later.	ARM IHI 0022D, Section C3.5 on pg C3-172	No
master_b_wack_eventually	All pending write transactions will be acknowledged eventually.	None	No
slave_w_wr_tbl_no_overflow_ace	Slave should not issue <code>wready</code> when write table is full and no write data is pending	None	Yes
slave_w_aw_wready_dbc	If data before control is allowed for slave, it should issue <code>wready</code> with or after <code>awready</code> when <code>DATA_ACCEPT_WITH_OR_AFTER_CONTROL</code> is high	None	Yes
master_w_wack_wait_cycles	All pending write transactions will be acknowledged within <code>MAX_WAIT_CYCLES_WACK</code> cycles	None	No
slave_b_bresp_while_evict	An evict transaction does not propagate downstream and the interconnect is required to generate an <code>OKAY BRESP[1:0] = 0b00</code> write response	ARM IHI 0022D: Section C6.4 on pg C6-242	Yes
master_w_wvalidchk_gen	<code>wvalidchk</code> should be generated whenever <code>aresetn</code> is asserted	ARM IHI 0022G on pg E2.6-392	Yes
master_w_wreadychk_gen	<code>wreadychk</code> should be generated whenever <code>aresetn</code> is asserted	ARM IHI 0022G on pg E2.6-392	Yes
master_w_wdataachk_gen	<code>wdataachk</code> should be generated whenever <code>wvalid</code> is asserted	ARM IHI 0022G on pg E2.6-392	Yes
master_w_wlastchk_gen	<code>wlastchk</code> should be generated whenever <code>wvalid</code> is asserted	ARM IHI 0022G on pg E2.6-392	Yes

AXI ABVIP User Guide
Appendix E: ACE5 ABVIP Package Contents--ACE5 Checks

master_w_wloopchk_gen	wloopchk should be generated whenever wvalid is asserted	ARM IHI 0022G on pg E2.6-392	Yes
master_w_wtracechk_gen	wtracechk should be generated whenever wvalid is asserted	ARM IHI 0022G on pg E2.6-392	Yes
master_w_wuserchk_gen	wuserchk should be generated whenever wvalid is asserted	ARM IHI 0022G on pg E2.6-392	Yes
master_w_wpoisonchk_gen	wpoisonchk should be generated whenever wvalid is asserted	ARM IHI 0022G on pg E2.6-392	Yes
slave_b_bvalidchk_gen	bvalidchk should be generated whenever aresetn is asserted	ARM IHI 0022G on pg E2.6-392	Yes
slave_b_breadychk_gen	breadychk should be generated whenever aresetn is asserted	ARM IHI 0022G on pg E2.6-392	Yes
slave_b_bidchk_gen	bidchk should be generated whenever bvalid is asserted	ARM IHI 0022G on pg E2.6-392	Yes
slave_b_bpoisonchk_gen	bpoisonchk should be generated whenever bvalid is asserted	ARM IHI 0022G on pg E2.6-392	Yes
slave_b_brespchk_gen	brespchk should be generated whenever bvalid is asserted	ARM IHI 0022G on pg E2.6-392	Yes
slave_b_buserchk_gen	buserchk should be generated whenever bvalid is asserted	ARM IHI 0022G on pg E2.6-392	Yes
slave_b_btracechk_gen	btracechk should be generated whenever bvalid is asserted	ARM IHI 0022G on pg E2.6-392	Yes
slave_b_bloopchk_gen	bloopchk should be generated whenever bvalid is asserted	ARM IHI 0022G on pg E2.6-392	Yes
slave_b_aw_persist_bcomp_with_bvalid	Slaves are required to send bcomp or bpersist with each bvalid	ARM IHI 0022G: Section E1.5.5 on pg E1-353	Yes

AXI ABVIP User Guide
Appendix E: ACE5 ABVIP Package Contents--ACE5 Checks

slave_b_aw_persist_trn_bcomp_bid_match	Slaves are required to reflect on the appropriate BID response as an ACE ID received from the master	ARM IHI 0022G: Section E1.5.5 on pg E1-353	Yes
slave_b_aw_bpersist_eventually	All pending bpersist write responses must start eventually	ARM IHI 0022G: Section E1.5	Yes
slave_b_aw_persist_trn_bpersist_bid_match	Slaves are required to reflect on the appropriate BID as an ACE ID received from the master	ARM IHI 0022G: Section E1.5.5 on pg E1-353	Yes
master_wdata_recom_inactivebytelines	wdata is driven to zero for inactive byte lanes	ARM IHI0022H on pg A3-42	Yes
1.7 Atomic Access			
master_ar_excl_arcache_low	Exclusive access burst cannot be modified.	ARM IHI 0022D, Section A7.2.4 on pg A7-97	Yes
master_aw_excl_no_lock_trans_for_shareable	There should not be any lock transaction for shareable domain from the write channel.	None	No
slave_b_excl_bresp_no_exokay_supported	The slave must not send an EXOKAY response for a normal write transaction.	ARM IHI 0022D, Section A7.2.3 on pg A7.93	Yes
master_ar_excl_arlen_correct	Exclusive accesses are not permitted to use a burst.	ARM IHI 0022D, Section A7.2.4 on pg A7.93	Yes
master_ar_excl_araddr_aligned	The address of an exclusive access must be aligned to the total number of bytes in the transaction.	ARM IHI 0022D, Section A7.2.4 on pg A7.93	Yes
master_ar_excl_arlen_arsize_correct_bytes	Number of bytes in an exclusive burst.	ARM IHI 0022D, Section A7.2.4 on pg A7.93	Yes
master_ar_excl_arlen_correct	Exclusive accesses are not permitted to use a burst length greater than 16.	ARM IHI 0022D, Section A7.2.4 on pg A7.93	Yes

AXI ABVIP User Guide
Appendix E: ACE5 ABVIP Package Contents--ACE5 Checks

master_ar_excl_araddr_aligned	The address of exclusive access must be aligned to total number of bytes in the transaction.	ARM IHI 0022D, Section A7.2.4 on pg A7.93	Yes
master_ar_excl_arlen_arsize_correct_bytes	Number of bytes in an exclusive burst should be 1,2,4,8,16,32,64 or 128.	ARM IHI 0022D, Section A7.2.4 on pg A7.93	Yes
master_ar_excl_load	An exclusive load must use either <code>ReadClean</code> or <code>ReadShared</code> transaction type An exclusive store must be a <code>CleanUnique</code> transaction.	ARM IHI 0022D, Section C9.6 on C9-268	No
master_aw_excl_write	The domain should be always System from write channel for exclusive transaction.	ARM IHI 0022D, Section C9.6 on C9-268	No
slave_r_excl_rresp_no_exokay	The slave must not send an EXOKAY response for a normal read transaction.	ARM IHI 0022D, Section A7.2.2 on pg A7.92	Yes
master_aw_excl_no_lock_trans_for_sharable	There should not be any lock transaction for sharable domain from the write channel	None	Yes
master_ar_excl_arsnoop	Exclusive accesses in case of shared domain is possible with <code>ReadClean/ReadShared/CleanUnique</code>	ARM IHI 0022F: Section D9.2.4 on pg D9.99	No
master_excl_ar_excl_store_when_excl_seq_prog	A master must not permit an Exclusive Store transaction at the same time as any transaction performing an Exclusive seq for ACE	ARM IHI 0022F: Section D9.2.3 on pg A7.99	No
master_ar_arlock_excl_shared_tbl_no_overflow	Master should not issue arlock when exclusive access table is full	ARM IHI 0022F	No
master_ar_arlock_excl_shared_domain_tbl_no_overflow	Master should not issue arlock when exclusive access table is full	ARM IHI 0022F	No
slave_r_excl_resp_shared_domain	The EXOKAY response is only permitted for a <code>ReadNoSnoop</code> , <code>ReadClean</code> , <code>ReadShared</code> or <code>CleanUnique</code> transaction	ARM IHI 0022F: Section D9.2.4 on pg D9.99	No
slave_excl_r_wr_resp_no_decerr	The slave can send an EXOKAY/OKAY/SLVERR response only when there is valid exclusive store request for ACE	ARM IHI 0022F: Section A7.2.2 on pg A7.98	No

AXI ABVIP User Guide
Appendix E: ACE5 ABVIP Package Contents--ACE5 Checks

slave_excl_r_shared_domain_resp_overlapping_addr	Exclusive Write fails when same location is updated since the exclusive read has happened for ACE	ARM IHI 0022F: Section A7.2.4 on pg A7.99	No
slave_ar_arready_excl_shared_domain_tbl_no_overflow	Slave should not accept exclusive access transaction when exclusive access table is full for shared domain	ARM IHI 0022F	No
1.8 Snoop Transaction			
slave_ac_aw_w_no_snoop_before_bresp_wack_scl	If an interconnect provides a master with a response to a write transaction, it should not send the same master a snoop transaction to the same cache line until it has received an acknowledgment of the transaction response on WACK.	ARM IHI 0022D, Section C6.2 on pg C6-227	No
slave_ac_aw_w_no_snoop_before_wns_bresp_wack_scl	If the interconnect provides a master with a response to a write transaction including <code>writenosnoop</code> , it must not send the same master a snoop transaction to the same cache line until it has received an acknowledgment of the transaction response. Set parameter <code>RECM_CHECK_ON</code> to 1, to enable this property.	None	No
slave_ac_acsnoop_legal_values	Legal Snoop address channel values of acsnoop, all unused encodings are reserved.	ARM IHI 0022D, Section C3.7 on pg C3-176	No
slave_ac_acaddr_wrap_aligned	The snoop address, ACADDR, must be aligned to the data transfer size, which is determined by the width of the snoop data bus in bytes.	ARM IHI 0022D, Section C3.6.2 on pg C3-174	No
slave_ac_ar_r_no_snoop_before_rresp_rack_scl	If the interconnect provides a master with a response to a read transaction, it must not send the same master a snoop transaction to the same cache line until it has received an acknowledgment of the transaction response on RACK.	ARM IHI 0022D, Section C6.2 on pg C6-227	No
slave_ac_ar_r_no_snoop_before_ns_rresp_rack_scl	If the interconnect provides a master with a response to a non shareable read transaction, it must not send the same master a snoop transaction to the same cache line until it has received an acknowledgment of the transaction response on RACK. Set <code>RECM_CHECKS_ON</code> to enable this property.	ARM IHI 0022D, Section C6.2 on pg C6-227	No
assert_ac_sn_tbl_no_overflow	Check that fifo is actually never overflowing.	None	No
master_cr_ac_crvalid	The master must wait for both <code>ACVALID</code> and <code>ACREADY</code> to be asserted before asserting <code>CRVALID</code> .	ARM IHI 0022D, Section C3.9 on pg C3-182	No

AXI ABVIP User Guide
Appendix E: ACE5 ABVIP Package Contents--ACE5 Checks

master_cr_ac_cd_crrresp_for_dbr	Data Transfer must be asserted if: - there is already a snoop data ongoing for the snoop burst (DBR). - there is already snoop data asserted for the same snoop burst (DWR).	ARM IHI 0022D, Section C3.9 on pg C3 182	No
master_cr_ac_cd_crrresp_legal_for_invalidate	Illegal responses are: - IsShared = 1 for a ReadUnique, CleanInvalid, MakeInvalid transaction.	ARM IHI 0022D, Section C3.7 on pg C3 179	No
master_cr_crrresp_legal	Illegal responses are: - DataTransfer = 0 and PassDirty = 1 for any transaction.	ARM IHI 0022D, Section C3.7 on pg C3 179	No
master_cr_aw_b_ac_crrresp_wrbkclean_scl	If a snooped master has a memory update in progress, using either a WriteBack or WriteClean transaction, when it receives a snoop transaction to the same line, then the snooped master must ensure that no other master can perform a memory update at the same time. This is done by giving a snoop response with PassDirty (CRRESP[2]) = 0 and IsShared (CRRESP[3]) = 1, which does not pass the permission to store to the line and does not pass responsibility for updating memory	ARM IHI 0022D, Section C5.2.3 on pg C5-216	No
assert_cr_sn_tbl_no_overflow	Check that fifo is actually never overflowing.	None	No
master_cd_ac_cr_cdvalid	The master must wait for both ACVALID and ACREADY to be asserted before asserting CDVALID.	ARM IHI 0022D, Section C3.9 on pg C3 182	No
master_cd_ac_cdlast_exact_len	CDLAST is asserted for the last data transfer of a snoop transaction.	ARM IHI 0022D, Section C3.8 on pg C3 180	No
assert_cd_sn_tbl_no_overflow	Check that fifo is actually never overflowing.	None	No
master_ac_acready_eventually	All snoop control requests must be eventually accepted.	None	No
slave_cr_crrready_eventually	All snoop response requests must be eventually given.	None	No
slave_cd_cdready_eventually	All snoop data requests must be eventually.	None	No

AXI ABVIP User Guide
Appendix E: ACE5 ABVIP Package Contents--ACE5 Checks

master_cr_crvalid_eventually	For every snoop address that is presented to a cached master on the snoop address channel, a corresponding response is required on the snoop response channel.	ARM IHI 0022D, Section C3.7 on pg C3 176	No
master_cd_cdvalid_eventually	All snoop request with data beats should come eventually.	None	No
master_cd_cdlast_eventually	All snoop request with data beats should finish eventually.	ARM IHI 0022D, Section C3.8 on pg C3 180	No
master_ac_sn_tbl_no_overflow	Master should not drive <code>acready</code> when snoop control table is full.	None	No
slave_ac_sn_tbl_no_overflow	Slave should not drive <code>acvalid</code> when snoop control table is full.	None	No
master_cr_sn_tbl_no_overflow	Master should not drive <code>crvalid</code> when snoop response table is full.	None	No
slave_cr_sn_tbl_no_overflow	Slave should not drive <code>crready</code> when snoop response table is full.	None	No
master_cd_sn_tbl_no_overflow	Master should not drive <code>cdvalid</code> when snoop data table is full.	None	Yes
slave_cd_sn_tbl_no_overflow	Slave should not drive <code>cdready</code> when snoop data table is full.	None	No
slave_ac_acready_wait_cycles	The maximum number of cycles for which <code>acready</code> remains low after <code>acvalid</code> going high must not exceed the <code>MAX_WAIT_CYCLES_AC</code> specified.	None	No
slave_cr_crready_wait_cycles	The maximum number of cycles for which <code>crready</code> remains low after <code>crvalid</code> going high must not exceed the <code>MAX_WAIT_CYCLES_CR</code> specified.	None	No
master_cr_snresp_crtrace	The master should respond with <code>crtrace</code> asserted if <code>actrace</code> was asserted in the snoop request	ARM IHI0022F Section E2.6 on page E2-356	No
master_cd_cr_dbr_cdtrace	The master should respond with <code>cdtrace</code> asserted if <code>actrace</code> was asserted in the snoop request	ARM IHI0022F Section E2.6 on page E2-356	No
master_cd_sn_rsp_done_cdtrace	The master should respond with <code>cdtrace</code> asserted if <code>actrace</code> was asserted in the snoop request	ARM IHI0022F Section E2.6 on page E2-356	No

AXI ABVIP User Guide
Appendix E: ACE5 ABVIP Package Contents--ACE5 Checks

master_cd_cr_valid_cdtrace	The master should respond with cdtrace asserted if actrace was asserted in the snoop request	ARM IHI0022F Section E2.6 on page E2-356	No
slave_ac_acvalidchk_gen	The slave should generate acvalidchk whenever aresetn is high	ARM IHI 0022G on pg E2.6-392	Yes
slave_ac_acreadychk_gen	The slave should generate acreadychk whenever aresetn is high	ARM IHI 0022G on pg E2.6-392	Yes
slave_ac_acaddrchk_gen	The slave should generate acaddrchk whenever acvalid is high	ARM IHI 0022G on pg E2.6-392	Yes
slave_ac_acctlchk_gen	The slave should generate acctlchk whenever acvalid is high	ARM IHI 0022G on pg E2.6-392	Yes
slave_ac_acvmidextchk_gen	The slave should generate acvmidextchk whenever acvalid is high	ARM IHI 0022G on pg E2.6-392	Yes
slave_ac_actracechk_gen	The slave should generate actracechk whenever acvalid is high	ARM IHI 0022G on pg E2.6-392	Yes
master_cd_cddatachk_gen	The Master should generate cddatachk whenever cdvalid is high	ARM IHI 0022G on pg E2.6-392	No
master_cd_cdvalidchk_gen	The Master should generate cdvalidchk whenever aresetn is high	ARM IHI 0022G on pg E2.6-392	No
master_cd_cdreadychk_gen	The Master should generate cdreadychk whenever aresetn is high	ARM IHI 0022G on pg E2.6-392	No
master_cd_cdlastchk_gen	The Master should generate cdlastchk whenever cdvalid is high	ARM IHI 0022G on pg E2.6-392	No
master_cd_cdpoisonchk_gen	The Master should generate cdpoisonchk whenever cdvalid is high	ARM IHI 0022G on pg E2.6-392	No
master_cr_crvalidchk_gen	The Master should generate crreadychk whenever aresetn is high	ARM IHI 0022G on pg E2.6-392	Yes

AXI ABVIP User Guide
Appendix E: ACE5 ABVIP Package Contents--ACE5 Checks

master_cr_crreadychk_gen	The Master should generate creadychk whenever aresetn is high	ARM IHI 0022G on pg E2.6-392	Yes
master_cr_crrespchk_gen	The Master should generate crrespchk whenever crvalid is high	ARM IHI 0022G on pg E2.6-392	Yes
master_cr_crnsaidchk_gen	The Master should generate crnsaidchk whenever crvalid is high	ARM IHI 0022G on pg E2.6-392	Yes
master_cr_crtracechk_gen	The Master should generate crtracechk whenever crvalid is high	ARM IHI 0022G on pg E2.6-392	Yes
1.9 Barrier Transaction			
ARM IHI 0022G on pg E2.6-392master_ar_bar_check_aruser	ARUSER cannot be reliably transported along with barrier transactions. Therefore, it is recommended that user defined signals be zero for barrier transactions.	ARM IHI 0022E, Section C8.2.3 on pg C8-260	Yes
master_aw_bar_check_awuser	AWUSER cannot be reliably transported along with barrier transactions. Therefore, it is recommended that user defined signals be zero for barrier transactions.	ARM IHI 0022E, Section C8.2.3 on pg C8-260	Yes
slave_r_bar_check_ruser	RUSER cannot be reliably transported along with barrier transactions. Therefore, it is recommended that user defined signals be zero for barrier transactions.	ARM IHI 0022E, Section C8.2.3 on pg C8-260	Yes
slave_b_bar_check_buser	BUSER cannot be reliably transported along with barrier transactions. Therefore, it is recommended that user defined signals be zero for barrier transactions.	ARM IHI 0022E, Section C8.2.3 on pg C8-260	Yes
slave_r_ar_bar_rlast_for_barrier	Barrier responses must have <code>RLAST</code> asserted.	ARM IHI 0022E, Section C8.2.3 on pg C8-260	Yes
master_aw_ar_bar_write_barrier_match	All barrier transactions must be issued in pairs, with a barrier transaction on both the read address and write address channels. This is referred to as issuing a "barrier pair". The two barrier transactions in a barrier pair must have the same <code>AxID</code> , <code>AxBAR</code> , <code>AxDOMAIN</code> and <code>AxPROT</code> values.	ARM IHI 0022D, Section C8.4.1 on pg C8-254	Yes

AXI ABVIP User Guide
Appendix E: ACE5 ABVIP Package Contents--ACE5 Checks

master_ar_aw_bar_read_barrier_match	All barrier transactions must be issued in pairs, with a barrier transaction on both the read address and write address channels. This is referred to as issuing a "barrier pair". The two barrier transactions in a barrier pair must have the same <code>AxID</code> , <code>AxBAR</code> , <code>AxDOMAIN</code> and <code>AxPROT</code> values.	ARM IHI 0022D, Section C8.4.1 on pg C8-254	Yes
master_ar_aw_bar_read_barrier_id_unused	Barrier transactions are required to use different ID values than are in use for non-barrier transactions. It is permissible for barrier transactions and non-barrier transactions to use the same AXI ID value if one has fully completed before the other is issued.	ARM IHI 0022D, Section C8.4.1 on pg C8-254	Yes
master_ar_aw_bar_read_normal_id_unused	Barrier transactions are required to use different ID values than are in use for non-barrier transactions. It is permissible for barrier transactions and non-barrier transactions to use the same AXI ID value if one has fully completed before the other is issued.	ARM IHI 0022D, Section C8.4.1 on pg C8-254	Yes
master_aw_ar_bar_write_barrier_id_unused	Barrier transactions are required to use different ID values than are in use for non-barrier transactions. It is permissible for barrier transactions and non-barrier transactions to use the same AXI ID value if one has fully completed before the other is issued.	ARM IHI 0022D, Section C8.4.1 on pg C8-254	Yes
master_aw_ar_bar_write_normal_id_unused	Barrier transactions are required to use different ID values than are in use for non-barrier transactions. It is permissible for barrier transactions and non-barrier transactions to use the same AXI ID value if one has fully completed before the other is issued.	ARM IHI 0022D, Section C8.4.1 on pg C8-254	Yes
master_ar_aw_bar_read_barrier_id_match	Barrier pairs must be issued in the same sequence on the read address and write address channels.	ARM IHI 0022D, Section C8.4.1 on pg C8-254	Yes
master_aw_ar_bar_write_barrier_id_match	Barrier pairs must be issued in the same sequence on the read address and write address channels.	ARM IHI 0022D, Section C8.4.1 on pg C8-254	Yes
master_aw_ar_bar_start_mem_updt	Shareable transactions are limited to memory update transactions like <code>WriteBack</code> , <code>WriteClean</code> , and <code>Evict</code> operations during a barrier.	ARM IHI 0022D, Section C8.4.1 on pg C8-255	No
master_ar_aw_bar_read_barrier_eventually	A master interface that has issued a write barrier on the AW channel, must eventually issue the corresponding read barrier on the AR channel if all other transactions on the AR channel are progressed.	ARM IHI 0022D, Section C8.4.1 on pg C8-255	Yes

AXI ABVIP User Guide
Appendix E: ACE5 ABVIP Package Contents--ACE5 Checks

master_aw_ar_bar_write_barrier_eventually	Barrier responses must have <code>RLAST</code> asserted.	ARM IHI 0022D, Section C8.2.3 on pg C8-250	Yes
slave_b_aw_bar_bresp_for_barrier	Barrier responses must have <code>BRESP</code> = 'b00.	ARM IHI 0022D, Section C8.2.3 on pg C8-250	Yes
slave_r_ar_bar_rresp_for_barrier	Barrier responses must have <code>RRESP</code> = 'b0000.	ARM IHI 0022D, Section C8.2.3 on pg C8-250	Yes
master_aw_ar_bar_tbl_no_overflow	Master should not drive <code>awbar</code> when barrier write table is full.	None	Yes
master_ar_aw_bar_tbl_no_overflow	Master should not drive <code>arbar</code> when barrier read table is full.	None	Yes
slave_aw_ar_bar_tbl_no_overflow	Slave should not issue <code>awready</code> for write barrier when barrier write table is full.	None	Yes
slave_ar_aw_bar_tbl_no_overflow	Slave should not issue <code>arready</code> for read barrier when barrier read table is full.	None	Yes
1.10 DVM Transaction			
master_ar_dvm_reserved	DVM all reserved bit format for initial DVM request.	ARM IHI 0022D, Section C12.5 on pg C12-286	No
master_ar_dvm_msg_type_reserved	DVM message type reserved bits for initial DVM request.	ARM IHI 0022D, Section C12.5 on pg C12-286	No
master_ar_addl_dvm_msg_type_reserved	Reserved bits for additional DVM message.	ARM IHI 0022D, Section C12.5 on pg C12-286	No
master_ar_dvm_tlb_inv_msg_format	Supported msg for TLB invalidate.	ARM IHI 0022D, Section C12.7.1 on pg C12-289	No

AXI ABVIP User Guide
Appendix E: ACE5 ABVIP Package Contents--ACE5 Checks

master_ar_dvm_brn_prd_inv_msg_format	Supported msg for branch predictor invalidate.	ARM IHI 0022D, Section C12.7.2 on pg C12-290	No
master_ar_dvm_phy_inst_cache_inv_msg_format	Supported msg for physical instruction cache invalidate.	ARM IHI 0022D, Section C12.7.3 on pg C12-291	No
master_ar_dvm_virtual_inst_cache_inv_msg_format	Supported msg for virtual instruction cache invalidate	ARM IHI 0022D, Section C12.7.4 on pg C12-292	No
master_ar_dvm_sync_msg_format	Supported msg for DVM synchronization	ARM IHI 0022D, Section C12.7.5 on pg C12-293	No
master_ar_dvm_hint_msg_format	Supported msg for hint	ARM IHI 0022D, Section C12.7.6 on pg C12-293	No
master_ar_araddr_dvm_complete	For a DVM complete message ARADDR is defined to be all zeros.	ARM IHI 0022D, Section C12.6 on pg C12-287	Yes
master_ar_dvm_addl_msg_follows_dvm_msg	Multipart DVM messages are always sent as successive transactions.	ARM IHI 0022D, Section C12.6 on pg C12-287	No
master_cr_dvm_addl_msg_resp	For multi-part DVM transactions, a response is provided for each transaction. For such transactions, it is required that a component must provide the same response to each transaction.	ARM IHI 0022D, Section C12.3.4 on pg C12-283	No
slave_ar_r_dvm_mesg_resp_format	RRESP can take value of 0000 or 0010 during DVM message.	ARM IHI 0022D, Section C12.3.4 on pg C12-283	No

AXI ABVIP User Guide
Appendix E: ACE5 ABVIP Package Contents--ACE5 Checks

slave_ar_r_dvm_sync_comp_resp_format	An error response is not permitted for a DVM sync or a DVM complete.	ARM IHI 0022D, Section C12.3.4 on pg C12-283, Section C12.7.6 on pg C12-294	No
slave_ar_r_rresp_same_for_addl_dvm_msg	For multi-part DVM messages, a response is provided for each transaction. It is required that the response given to all parts of a multiple transaction message are the same.	ARM IHI 0022D, Section C12.3.4 on pg C12-283	No
master_ar_dvm_sync_msg	This is a synchronization transaction that a component issues to check that all previous DVM Operations that it has issued have completed.	ARM IHI 0022D, Section C12.1 on pg C12-280	No
master_ar_r_dvm_id_unused	DVM messages are required to use AXI ID values that are not in use by any normal (non-DVM) transactions issued on the AR channel.	ARM IHI 0022D, Section C12.3.5 on pg C12-283, C8.4.1, page C8-256	No
master_ar_r_dvm_sync_eventually	This is a synchronization transaction that a component issues to check that all previous DVM Operations that it has issued have completed.	ARM IHI 0022D, Section C12.2 on pg C12-281	No
slave_ar_r_dvm_complete_follows_dvm_sync_eventually	A DVM Sync must complete in a timely manner.	ARM IHI 0022D, Section C12.2 on pg C12-281	No
slave_ac_dvm_reserved	DVM all reserved bit format for initial DVM request.	ARM IHI 0022D, Section C12.5 on pg C12-286	Yes
slave_ac_dvm_msg_type_reserved	DVM reserved bit for message types for initial DVM request.	ARM IHI 0022D, Section C12.5 on pg C12-286	Yes

AXI ABVIP User Guide
Appendix E: ACE5 ABVIP Package Contents--ACE5 Checks

slave_ac_addl_dvm_msg_type_reserved	Reserved bits for additional DVM message.	ARM IHI 0022D, Section C12.5 on pg C12-286.	Yes
slave_ac_dvm_tlb_inv_msg_format	Supported msg for TLB invalidate.	ARM IHI 0022D, Section C12.7.1 on pg C12-289	Yes
slave_ac_dvm_brn_prd_inv_msg_format	Supported msg for branch predictor invalidate.	ARM IHI 0022D, Section C12.7.2 on pg C12-290	Yes
slave_ac_dvm_phy_inst_cache_inv_msg_format	Supported msg for physical instruction cache invalidate.	ARM IHI 0022D, Section C12.7.3 on pg C12-291	Yes
slave_ac_dvm_virtual_inst_cache_inv_msg_format	Supported msg for virtual instruction cache invalidate.	ARM IHI 0022D, Section C12.7.4 on pg C12-292	Yes
slave_ac_dvm_sync_msg_format	Supported msg for DVM Synchronization.	ARM IHI 0022D, Section C12.7.5 on pg C12-293	Yes
slave_ac_dvm_hint_msg_format	Supported msg for hint.	ARM IHI 0022D, Section C12.7.6 on pg C12-293	Yes
slave_ac_acaddr_dvm_complete	For a DVM complete message ACADDR is defined to be all zeros.	ARM IHI 0022D, Section C12.6 on pg C12-287	Yes
slave_ac_dvm_addl_msg_follows_dvm_msg	Multipart DVM messages are always sent as successive transactions.	ARM IHI 0022D, Section C12.3.3 on pg C12-282	Yes

AXI ABVIP User Guide
Appendix E: ACE5 ABVIP Package Contents--ACE5 Checks

master_cr_ac_dvm_crresp	<p>If the component can perform the requested DVM action, it must respond by setting <code>CRRESP</code> = "00000". If the component is unable to perform the requested DVM action, it must respond by setting <code>CRRESP</code> = "00010" to indicate that it is an unsupported message. A component is not permitted to set <code>CRRESP</code> to 00010 in response to DVM Sync or a DVM Complete</p> <p>No data transfer is associated with DVM transactions (<code>CRRESP[0] = 0</code>) All Hint Messages (<code>acaddr[15]=0</code>) must respond with response value <code>CRRESP</code> set to 0.</p>	ARM IHI 0022D, Section C12.3.4 on pg C12-283, C12.7.6 on pg C12-293	Yes
master_ar_ac_dvm_complete_after_handshake	A DVM Complete on the AR channel must only be issued after the handshake of the associated DVM Sync on the same master's AC channel.	ARM IHI 0022D, Section C12.3.2 on pg C12-282	No
master_ar_ac_dvm_complete_after_dvm_sync_eventually	DVM complete is issued by a component that has received a number of DVM operations and a DVM Sync.	ARM IHI 0022D, Section C12.2 on pg C12-281	No
master_ar_dvm_araddr_23_16	If <code>ARADDR[5]</code> of the first transaction is de-asserted, <code>ARADDR[23:16]</code> of the first transaction must be all zeros for all defined message types except Hint.	ARM IHI 0022D, Section C12.6 on pg C12-298	Yes
master_ar_dvm_araddr_39_32	If <code>ARADDR[5]</code> of the first transaction is de-asserted, <code>ARADDR[39:32]</code> of the first transaction must be all zeros for all defined message types except Hint.	ARM IHI 0022D, Section C12.6 on pg C12-298	Yes
master_ar_araddr_31_0	If <code>ARADDR[6]</code> of the first transaction is de-asserted, <code>ARADDR[31:24]</code> of the first transaction must be all zeros for all defined message types except Hint.	ARM IHI 0022D, Section C12.6 on pg C12-298	Yes
master_ar_r_dvm_sync_wait_cycles	Pending DVM Sync transactions will be acknowledged within <code>MAX_WAIT_CYCLES_DVM_SYNC</code> cycles.	None	Yes
master_ar_dvm_msg_vmidext	DVM message <code>araddr[6]</code> is de-asserted, <code>arvmidext</code> must be all zeros except dvm hint.	ARM IHI 0022F : Section E2.11 on page E2-368	No

AXI ABVIP User Guide
Appendix E: ACE5 ABVIP Package Contents--ACE5 Checks

master_ar_dvm_addl_msg_vmidext	DVM message araddr[6] is de-asserted, arvmidext must be all zeros except dvm hint.	ARM IHI 0022F : Section E2.11 on page E2-368	No
slave_ac_dvm_msg_vmidext	DVM message acaddr[6] is de-asserted, acvmidext must be all zeros except dvm hint	ARM IHI 0022F : Section E2.11 on page E2-368	No
slave_ac_dvm_addl_msg_vmidext	DVM message acaddr[6] is de-asserted, acvmidext must be all zeros except dvm hint.	ARM IHI 0022F : Section E2.11 on page E2-368	No
1.11 Low Power Checks			
master_rst_syscoreq_low_reset	Master should drive syscoreq low during reset	ARM IHI 0022F : Section E2.10 on page E2-362	Yes
master_rst_syscoack_low_reset	Master should drive syscoack low during reset	ARM IHI 0022F : Section E2.10 on page E2-362	Yes
slave_lpi_acwakeup_cycle_before_acvalid	It is recommended that acwakeup is asserted a cycle before assertion of valid	ARM IHI 0022F Section E2.9.2 on page E2-361	No
slave_lpi_acwakeup_acvalid_stable	acwakeup must remain asserted until associated valid/ready handshake	ARM IHI 0022F Section E2.9.2 on page E2-361	No
master_lpi_syscoreq_asserted	Master can only change syscoreq when syscoack is at the same level	ARM IHI 0022F section E2.10.1 on page E2-363	Yes
master_lpi_syscoreq_deasserted	Master can only change syscoreq when syscoack is at the same level	ARM IHI 0022F section E2.10.1 on page E2-363	Yes

AXI ABVIP User Guide
Appendix E: ACE5 ABVIP Package Contents--ACE5 Checks

master_lpi_syscoack_asserted	Master can only change syscoack when syscoreq is at the same level	ARM IHI 0022F section E2.10.1 on page E2-363	Yes
master_lpi_syscoack_deasserted	Master can only change syscoack when syscoreq is at the same level	ARM IHI 0022F section E2.10.1 on page E2-363	Yes
master_lpi_aw_coherent_ccs_enabled	Master can issue a transaction to a coherent location only when coherency is enabled state	ARM IHI 0022F section E2.10.2 table E2.16 on page E2-365	Yes
master_lpi_ar_coherent_ccs_enabled	Master can issue a transaction to a coherent location only when coherency is enabled state	ARM IHI 0022F section E2.10.2 table E2.16 on page E2-365	Yes
slave_lpi_acvalid_ccs_enabled	Interconnect can not issue snoop transaction when coherency is disabled	ARM IHI 0022F section E2.10.2 table E2.16 on page E2-365	No
master_lpi_ccs_awakeup_asserted	It is required that the AWAKEUP signal is asserted to guarantee forward progress of a transition on the Coherency connect signaling.	ARM IHI 0022F section E2.9.1 on page E-360	Yes
master_lpi_ccs_awakeup_asserted_stable	When AWAKEUP is asserted with SYSCOREQ de-asserted and SYSCOACK asserted, it must remain asserted until SYSCOACK is de-asserted .When AWAKEUP is asserted with SYSCOREQ asserted and SYSCOACK de-asserted, it must remain asserted until SYSCOACK is asserted	ARM IHI 0022F section E2.9.1 on page E-360	Yes
Master_syscoreqchk_gen	Master should always generate syscoreqchk	ARM IHI 0022G on pg E2.6-392	Yes
slave_syscoackchk_gen	Interconnect should always generate syscoackchk	ARM IHI 0022G on pg E2.6-392	Yes
1.12 DSU Feature Checks			

AXI ABVIP User Guide
Appendix E: ACE5 ABVIP Package Contents--ACE5 Checks

master_bready_high	ACE master always accepts write responses without any delay by holding bready high.	ARM DSUTRM Section A6.9 on pg A6-96	No
master_wack_high	ACE master asserts write acknowledge signal high in the cycle following acceptance of a write response.	ARM DSUTRM Section A6.12 on pg A6-96	No
master_awprot0_high	ACE master always drive axprot0[0] signal high indicating that access could be a privileged access.	ARM DSUTRM Section A6.12 on pg A6-96	No
master_awid_deviceWr	ACE master allows only Device write for a particular AWID encodings.	ARM DSUTRM Section A6.4 Table A6-5 on pg A6-86	No
master_awid_000t0nnn_count	ACE master allows only one issuing capability for this id.	ARM DSUTRM Section A6.4 Table A6-5 on pg A6-86	No
master_awid_001t0nnn_count	ACE master allow maximum of 15 issuing capability for this id.	ARM DSUTRM Section A6.4 Table A6-5 on pg A6-86	No
master_awid_1xxxxxxx_count	ACE master allow maximum 1 issuing capability for this id.	ARM DSUTRM Section A6.4 Table A6-5 on pg A6-86	No
master_awid_valid	ACE master allow only a few AWID encodings.	ARM DSUTRM Section A6.4 Table A6-5 on pg A6-86	No
master_awid_excl_valid	ACE master allow only particular AWID encoding for system domain store exclusive only	ARM DSUTRM Section A6.4 Table A6-5 on pg A6-86	No

AXI ABVIP User Guide
Appendix E: ACE5 ABVIP Package Contents--ACE5 Checks

master_aw_not_allowed	ACE master doesn't allow WriteUniquePtl, WriteUniqueFull and, Barrier transactions	ARM DSUTRM Section A6.6 Table A6-8 on pg A6-90	No
master_aw_stashNot_allowed	ACE master doesn't allow stash transactions.	ARM DSUTRM Section A6.6 Table A6-8 on pg A6-90	No
master_incr2or4_128bit_wr	ACE master allow INCR N(N:2 or 4) 128-bit write transfers for normal non-cacheable or device transactions	ARM DSUTRM Section A6.6 on pg A6-89	No
master_incr1_8to128bit_wr	ACE master allow INCR 1 8-bit, 16-bit, 32-bit, 64bit, 128-bit write or exclusive write transfers for normal non-cacheable or device transactions	ARM DSUTRM Section A6.6 on pg A6-89	No
master_incr4_128bit_wr_evict	ACE master allow INCR 4 128-bit write transfers for writeback cacheable transactions.	ARM DSUTRM Section A6.6 on pg A6-89	NO
master_aw_awaddr_never_cross_64B_boundary	Bursts must not cross cache line boundary that is 64B.	ARM DSUTRM Section A6.6 on pg A6-89	No
master_rack_high	ACE master asserts read acknowledge signal high in the cycle following acceptance of the last read data channel transfer.	ARM DSUTRM Section A6.8 on pg A6-92	No
master_arprot0_high	ACE master always drive axprot0[0] signal high indicating that access could be a privileged access.	ARM DSUTRM Section A6.12 on pg A6-96	No
master_arid_deviceRd	ACE master allows only device read for a particular ARID encoding.	ARM DSUTRM Section A6.4 Table A6-6 on pg A6-87	No

AXI ABVIP User Guide
Appendix E: ACE5 ABVIP Package Contents--ACE5 Checks

master_arid_0000t0nnn_count	ACE master allow only one issuing capability for this id.	ARM DSUTRM Section A6.4 Table A6-6 on pg A6-87	No
master_arid_0001t0nnn_count	ACE master allow maximum 17 issuing capability for this id	ARM DSUTRM Section A6.4 Table A6-6 on pg A6-87	No
master_arid_001000000_count	ACE master allows a maximum of only one dvm sync message for this id	ARM DSUTRM Section A6.4 Table A6-6 on pg A6-87	No
master_arid_001000001_count	ACE master allows a maximum of 256 dvm complete message for this id.	ARM DSUTRM Section A6.4 Table A6-6 on pg A6-87	No
master_arid_01xxxxxxx_count	ACE master allow a maximum of 1 issuing capability for this id.	ARM DSUTRM Section A6.4 Table A6-6 on pg A6-87	No
master_arid_1xxxxxxx_count	ACE master allow a maximum of 1 issuing capability for this id.	ARM DSUTRM Section A6.4 Table A6-6 on pg A6-87	No
master_arid_valid	ACE master allow only a few ARID encoding for read.	ARM DSUTRM Section A6.4 Table A6-6 on pg A6-87	No
master_arid_excl_valid	ACE master allow this ARID encoding for load exclusives and cacheable shareable store exclusives	ARM DSUTRM Section A6.4 Table A6-6 on pg A6-87	No

AXI ABVIP User Guide
Appendix E: ACE5 ABVIP Package Contents--ACE5 Checks

master_dvmsync_valid	ACE master allow dvm sync message for a particular id.	ARM DSUTRM Section A6.4 Table A6-6 on pg A6-87	No
master_dvmcomplete_valid	ACE master allow dvm complete message for a particular id.	ARM DSUTRM Section A6.4 Table A6-6 on pg A6-87	No
master_ar_not_allowed	ACE master doesn't allow MakeInvalid, ReadShared, ReadOnceCleanInvalid, ReadOnceMakeInvalid	ARM DSUTRM Section A6.4 Table A6-6 Table A6-8 on pg A6-90	No
master_incr2or4_128bit_rd	ACE master allow INCR N(N:2 or 4) 128-bit read transfers for normal non-cacheable or device transactions.	ARM DSUTRM Section A6.4 Table A6-6 on pg A6-89	No
master_wrap2or4_128bit_rd	ACE master allow WRAP N(N:2 or 4) 128-bit read transfers for normal non-cacheable or device transactions.	ARM DSUTRM Section A6.6 on pg A6-89	No
master_incr1_8to_128bit_rd	ACE master allow INCR 1 8-bit, 16bit, 32-bit, 64-bit, 128-bit read or exclusive read transfers for normal non-cacheable or device transactions.	ARM DSUTRM Section A6.6 on pg A6-89	No
master_wrap4_128bit_rd_linefills	ACE master allow WRAP 4 128-bit read transfers for writeback cacheable transactions.	ARM DSUTRM Section A6.6 on pg A6-89	No
master_incr4_128bit_rd_linefills	ACE master allow INCR 4 128-bit read transfers for writeback cacheable transactions.	ARM DSUTRM Section A6.6 on pg A6-89	No
master_ar_araddr_never_cross_64B_boundary	Bursts must not cross cache line boundary i.e 64B.	ARM DSUTRM Section A6.6 on pg A6-89	No

AXI ABVIP User Guide
Appendix E: ACE5 ABVIP Package Contents--ACE5 Checks

master_ac_not_allowed	ACE master doesn't allow Readshared and MakeInvalid.	ARM DSUTRM Section A6.6 Table A6-8 on pg A6-90	No
1.13 Read Data Chunking Checks			
master_ar_chunken_size_arlen	ARCHUNKEN must only be asserted for transactions with ARSIZE is equal to the data bus width or ARLEN is one beat.	ARM IHI0022G: section E1.14.1	Yes
master_ar_chunken_size	ARCHUNKEN must only be asserted for transactions with ARSIZE is equal or greater than 128 bits.	ARM IHI0022G: section E1.14.1	Yes
master_ar_chunken_araddr	ARCHUNKEN must only be asserted for transactions with ARADDR is aligned to 16 bytes.	ARM IHI0022G: section E1.14.1	Yes
master_ar_chunken_arburst	ARCHUNKEN must only be asserted for transactions with ARBURST is INCR or WRAP.	ARM IHI0022G: section E1.14.1	Yes
master_no_rd_req_with_same_arid	The master must not issue a request on the read channel with the same ARID as an outstanding request that had ARCHUNKEN asserted	ARM IHI0022G: section E1.14.1	Yes
master_chunkid_no_outstanding_rd_trans	ARCHUNKEN can only be asserted if there are no outstanding read transactions using the same ARID value	ARM IHI0022G: section E1.14.1	Yes
master_ar_chunken_aridunq	If present on the interface, ARIDUNQ must be asserted	ARM IHI0022G: section E1.14.1	Yes
slave_r_rchunkv_atomic	RCHUNKV cannot be asserted for ARCHUNKEN, Atomic transaction are sent on write channel.	None	Yes
slave_r_archunken_rchunkv_match	If ARCHUNKEN is asserted ,the corresponding RCHUNKV can be asserted response beats of the transaction.	ARM IHI0022G: section E1.14.1	Yes
slave_r_rchunkv_same_all_beat	RCHUNKV must be the same for every response beat of a transaction.	ARM IHI0022G: section E1.14.1	Yes
slave_r_ar_chunken_rchunknum_limit	When RVALID and RCHUNKV are asserted, RCHUNKNUM must be between zero and ARLEN.	ARM IHI0022G: section E1.14.1	Yes

AXI ABVIP User Guide
Appendix E: ACE5 ABVIP Package Contents--ACE5 Checks

slave_r_ar_chunken_rchunkstrb	When RVALID and RCHUNKV are asserted, RCHUNKSTRB must not be zero.	ARM IHI0022G: section E1.14.1	Yes
slave_r_ar_chunken_deassert	When RVALID is asserted and RCHUNKV is de-asserted, RCHUNKNUM and RCHUNKSTRB must be zero.	ARM IHI0022G: section E1.14.1	Yes
slave_r_ar_chunkstrb_match	The strobes of data chunk transferred must be unique for every chunk number	ARM IHI0022G: section E1.14.1	Yes
slave_r_ar_rlast_chunklen	The slave must assert the rlast signal when it drives the final read transfer in the burst	ARM IHI0022G: section E1.14.1	Yes
1.13 Regular Transaction Checks			
master_regular_arlen_arburst	For regular transactions- ARLEN is 1, 2, 4, 8 or 16. ARBURST is INCR or WRAP, not fixed.	ARM IHI0022H: section A3.4.3	Yes
master_regular_arsize	For regular transactions- ARSIZE is the same as data bus width, if ARLEN is greater than 1.	ARM IHI0022H: section A3.4.3	Yes
master_regular_arsize	For regular transactions- ARSIZE is the same as data bus width, if ARLEN is greater than 1.	ARM IHI0022H: section A3.4.3	Yes
master_regular_araddr_incr_aligned	For regular transactions- ARADDR is aligned to the transaction container for INCR transactions.	ARM IHI0022H: section A3.4.3	Yes
master_regular_araddr_wrap_aligned	For regular transactions- ARADDR is aligned to ARSIZE for WRAP transactions.	ARM IHI0022H: section A3.4.3	Yes
master_regular_awlen_awburst	For regular transactions- AWLEN is 1, 2, 4, 8 or 16. AWBURST is INCR or WRAP, not fixed.	ARM IHI0022H: section A3.4.3	Yes
master_regular_awsized	For regular transactions- AWSIZE is the same as data bus width, if ARLEN is greater than 1.	ARM IHI0022H: section A3.4.3	Yes
master_regular_awaddr_incr_aligned	For regular transactions- AWADDR is aligned to the transaction container for INCR transactions.	ARM IHI0022H: section A3.4.3	Yes
master_regular_awaddr_wrap_aligned	For regular transactions- AWADDR is aligned to AWSIZE for WRAP transactions.	ARM IHI0022H: section A3.4.3	Yes

1.14 UNTRANSLATED TRANSACTION VERSION 2 CHECKS

master_ar_armmuatst_zero_v2	This signal is not present for Untranslated Transactions v2. If not present the default value is 0	ARM IHI 0022H: Section E1.9.1 on pg E1-370	No
master_ar_atst_flow_mmusecid_low	When the flow is ATST, AxMMUSECSID must be LOW	ARM IHI 0022F: Section E2.12.2 on pg E2-371	No
master_ar_atst_flow_mmuSSIDV_low	When the flow is ATST, AxMMUSSIDV must be LOW	ARM IHI 0022F: Section E2.12.2 on pg E2-371	No
slave_r_ar_resp_consistent_transfault	If TRANSFAULT is used for one response beat, it must be used for all response beats of a transaction;	ARM IHI 0022F: Section E2.12.2 on pg E2-371	No
slave_r_resp_transfault_check	If ARMMUFLOW is not PRI, RRESP must not be TRANSFAULT	ARM IHI 0022H: Section E1.9.6 on pg E1-374	No
slave_r_resp_legal_val	RRESP can be normal response or transfault	ARM IHI 0022H: Section E1.9.1 on pg E1-370	No
master_aw_awmmuatst_zero_v2	this signal is not present for Untranslated Transactions v2. If not present the default value is 0;	ARM IHI 0022H: Section E1.9.1 on pg E1-370	No
master_aw_atst_flow_mmusecid_low	When the flow is ATST, AxMMUSECSID must be LOW	ARM IHI 0022F: Section E2.12.2 on pg E2-371	No
master_aw_atst_flow_mmuSSIDV_low	When the flow is ATST, AxMMUSSIDV must be LOW	ARM IHI 0022F: Section E2.12.2 on pg E2-371	No

AXI ABVIP User Guide
Appendix E: ACE5 ABVIP Package Contents--ACE5 Checks

slave_b_resp_transfault_check	If AWMMUFLOW is not PRI, BRESP must not be TRANSFAULT	ARM IHI 0022H: Section E1.9.6 on pg E1-374	No
slave_b_resp_legal_val	BRESP can be normal response or transfault	ARM IHI 0022H: Section E1.9.1 on pg E1-371	No
master_aw_awmmuflow_zero_v1	this signal is not present for Untranslated Transactions v1. If not present the default value is 0	ARM IHI 0022H: Section E1.9.1 on pg E1-371	No
master_ar_armmuflow_zero_v1	this signal is not present for Untranslated Transactions v1. If not present the default value is 0	ARM IHI 0022H: Section E1.9.1 on pg E1-371	No
1.15 Memory Tagging Checks			
master_ar_artagop_arburst	AxBURST must be INCR or WRAP, not FIXED	ARM IHI 0022H: Section E1.15.4 on pg E1-389	Yes
master_aw_awtagop_awburst	AxBURST must be INCR or WRAP, not FIXED	ARM IHI 0022H: Section E1.15.4 on pg E1-389	Yes
master_ar_artagop_arcache	The transaction must be to Normal Write-Back memory	ARM IHI 0022H: Section E1.15.4 on pg E1-389	Yes
master_aw_awtagop_awcache	The transaction must be to Normal Write-Back memory	ARM IHI 0022H: Section E1.15.4 on pg E1-389	Yes
master_ar_artagop_aridunq	ARIDUNQ must be asserted for a read with tag Transfer or Fetch	ARM IHI 0022H: Section E1.15.4 on pg E1-389	Yes

AXI ABVIP User Guide
Appendix E: ACE5 ABVIP Package Contents--ACE5 Checks

master_aw_awtagop_awidunq	AWIDUNQ must be asserted for a write with tag Transfer or Fetch	ARM IHI 0022H: Section E1.15.4 on pg E1-389	Yes
master_ar_artagop_no_matching_id	A read with tag Transfer or Fetch can only be issued if there are no outstanding read transactions using the same ARID value	ARM IHI 0022H: Section E1.15.4 on pg E1-389	Yes
master_aw_awtagop_no_matching_id	A write with tag Transfer or Fetch can only be issued if there are no outstanding write transactions using the same AWID value	ARM IHI 0022H: Section E1.15.4 on pg E1-389	Yes
master_ar_read_no_matching_artagop_id	master must not issue a request on the read channel with the same ARID as an outstanding read with tag Transfer or Fetch	ARM IHI 0022H: Section E1.15.4 on pg E1-389	Yes
master_aw_write_no_matching_awtagop_idYes	Yes master must not issue a request on the write channel with the same AWID as an outstanding Write with tag Transfer or Fetch	ARM IHI 0022H: Section E1.15.4 on pg E1-389	Yes
slave_r_rtag_strb	For read transactions that use strobes, only tags which correspond to valid strobes are required to be valid;	ARM IHI 0022H: Section E1.15.4 on pg E1-390	Yes
slave_r_rtag_invalid_op	For read transactions that has invalid tag operation, rtags RTAG is invalid and must be zero	ARM IHI 0022H: Section E1.15.5 on pg E1-390	Yes
master_ar_artagop_legal_val	Read request tag operation, encoded as:0 Invalid, 1 Transfer, 2 Reserved, 3 Fetch	ARM IHI 0022H: Section E1.15.2 on pg E1-388	Yes
master_w_wtag_invalid_op	For write transactions invalid tag operation, WTAG is invalid and must be zero and WTAGUPDATE must be de-asserted	ARM IHI 0022H: Section E1.15.6 on pg E1-392	Yes

AXI ABVIP User Guide
Appendix E: ACE5 ABVIP Package Contents--ACE5 Checks

master_w_wtag_invalid_op_curr	For write transactions invalid tag operation, WTAG is invalid and must be zero and WTAGUPDATE must be de-asserted	ARM IHI 0022H: Section E1.15.6 on pg E1-392	Yes
master_w_wtag_invalid_op_dbc	For write transactions invalid tag operation, WTAG is invalid and must be zero and WTAGUPDATE must be de-asserted	ARM IHI 0022H: Section E1.15.6 on pg E1-392	Yes
master_w_wtagupdate_transfer_match	For write transactions transfer tag operation, WTAG bits must be valid for every byte in the transaction container, WTAGUPDATE must be de-asserted	ARM IHI 0022H: Section E1.15.6 on pg E1-392	Yes
master_w_wtagupdate_transfer_match_curr	For write transactions transfer tag operation, WTAG bits must be valid for every byte in the transaction container, WTAGUPDATE must be de-asserted	ARM IHI 0022H: Section E1.15.6 on pg E1-392	Yes
master_w_wtagupdate_transfer_match_dbc	For write transactions transfer tag operation, WTAG bits must be valid for every byte in the transaction container, WTAGUPDATE must be de-asserted	ARM IHI 0022H: Section E1.15.6 on pg E1-392	Yes
master_wtag_wtagupdate_same_for_less_data_width	multiple beats address the same tag, WTAG and WTAGUPDATE values must be consistent for each 4-bit tag that is accessed by the transaction	ARM IHI 0022H: Section E1.15.4 on pg E1-390	Yes
master_awtag_transfer_update_no_awtop	Atomic transactions cannot be used with Transfer or Update operations	ARM IHI 0022H: Section E1.15.7 on pg E1-394	Yes
master_awtag_awtop_compare_wtag_same_cmp_swp_bytes	For AtomicCompare transactions of 32 bytes, the same tag value must be used for tag bits associated with the compare and swap bytes	ARM IHI 0022H: Section E1.15.7 on pg E1-394	Yes
master_awtag_awtop_compare_wtag_same_cmp_swp_bytes_dbc	AtomicCompare transactions of 32 bytes, the same tag value must be used for tag bits associated with the compare and swap bytes	ARM IHI 0022H: Section E1.15.7 on pg E1-394	Yes

AXI ABVIP User Guide
Appendix E: ACE5 ABVIP Package Contents--ACE5 Checks

master_awtag_awtop_compare_wtag_same_cmp_swp_bytes_curr	AtomicCompare transactions of 32 bytes, the same tag value must be used for tag bits associated with the compare and swap bytes	ARM IHI 0022H: Section E1.15.7 on pg E1-394	Yes
master_awtag_match_awtop_compare_size	AtomicCompare transactions with Match can be 16 bytes or 32 bytes	ARM IHI 0022H: Section E1.15.7 on pg E1-394	Yes
slave_r_rtag_atomic_resp	Read data that is returned within an Atomic Transaction does not have valid RTAG values, so RTAG is recommended to be zero	ARM IHI 0022H: Section E1.15.7 on pg E1-394	Yes
master_w_wtagupdate_update_op	Tags that are only partially addressed in the transaction must have WTAGUPDATE deasserted. WriteUniqueFull and WriteFullCMO with Update must have all associated WTAGUPDATE bits asserted	ARM IHI 0022H: Section E1.15.6 on pg E1-392	Yes
slave_b_aw_bvalid_for_match_tagop	Slaves are required to send bcomp or btagmatch with each bvalid	ARM IHI 0022H: Section E1.15.6 on pg E1-393	Yes
slave_b_aw_bvalid_btagmatch_part2_rep	BID must have the same value as AWID	ARM IHI 0022H: Section E1.15.6 on pg E1-393	Yes
slave_b_aw_bcomp_btagmatch_valid_value_tagop	Valid combinations of BCOMP and BTAGMATCH	ARM IHI 0022H: Section E1.15.6 on pg E1-394	Yes
slave_b_aw_bcomp_btagmatch_valid_value_no_tagop	Valid combinations of BCOMP and BTAGMATCH	ARM IHI 0022H: Section E1.15.6 on pg E1-394	Yes
slave_b_aw_btagmatch_bresp_no_exokay	BRESP can be OKAY, SLVERR, or DECERR	ARM IHI 0022H: Section E1.15.6 on pg E1-393	Yes

slave_b_aw_bcomp_bresp_no_transfault	BRESP can be OKAY, SLVERR, EXOAKY or DECERR;	ARM IHI 0022H: Section E1.15.6 on pg E1-393	Yes
slave_b_aw_btagmatch_eventually	All pending btagmatch write responses must start eventually	ARM IHI 0022G: Section E1.5	Yes
master_b_bready_btagmatch_eventually	All btagmatch must be eventually accepted	ARM IHI 0022G: Section E1.5	Yes
master_aw_awtagop_legal_val	Write request tag operation, encoded as: 0 Invalid, 1 Transfer, 2 Update, 3 Match	ARM IHI 0022H: Section E1.15.2 on pg E1-388	Yes

Coverage Checks

The monitor provides coverage checks to collect coverage information for possible scenarios in an ACE-based design. The coverage information is required for complete verification of the DUV.

To view AXI4 coverage checks, refer to [AXI Monitor Coverage Points](#). To view additional ACE checks, refer to [Table C-7](#).

Table C-7 ACE Monitor Coverage Points

Coverage Assertions	Description
cover_awdomain_nonshared	awdomain can be of non-shared type
cover_awdomain_inner	awdomain can be of inner type
cover_awdomain_outer	awdomain can be of outer type
cover_awdomain_system	awdomain can be of system type
cover_ardomain_nonshared	ardomain can be of non-shared type
cover_ardomain_inner	ardomain can be of inner type
cover_ardomain_outer	ardomain can be of outer type
cover_ardomain_system	ardomain can be of system type
cover_awcache_x_awdomain_device_system	Cover proper awdomain and awcache combination
cover_awcache_x_awdomain_cacheable_notshared	Cover proper awdomain and awcache combination
cover_awcache_x_awdomain_cacheable_inner	Cover proper awdomain and awcache combination
cover_awcache_x_awdomain_cacheable_outer	Cover proper awdomain and awcache combination
cover_awcache_x_awdomain_cacheable_system	Cover proper awdomain and awcache combination
cover_awcache_x_awdomain_writeThBck_notshared	Cover proper awdomain and awcache combination
cover_awcache_x_awdomain_writeThBck_inner	Cover proper awdomain and awcache combination
cover_awcache_x_awdomain_writeThBck_outer	Cover proper awdomain and awcache combination

AXI ABVIP User Guide
Appendix E: ACE5 ABVIP Package Contents--Coverage Checks

cover_arcache_x_ardomain_device_system	Cover proper awdomain and awcache combination
cover_arcache_x_ardomain_cacheable_notshared	Cover proper awdomain and awcache combination
cover_arcache_x_ardomain_cacheable_inner	Cover proper awdomain and awcache combination
cover_arcache_x_ardomain_cacheable_outer	Cover proper awdomain and awcache combination
cover_arcache_x_ardomain_cacheable_system	Cover proper awdomain and awcache combination
cover_arcache_x_ardomain_writeThBck_notshared	Cover proper awdomain and awcache combination
cover_arcache_x_ardomain_writeThBck_inner	Cover proper awdomain and awcache combination
cover_arcache_x_ardomain_writeThBck_outer	Cover proper awdomain and awcache combination
cover_ar_trgrp_NonSnoop	Cover read non snoop transaction
cover_ar_trgrp_Coherent Cover	read coherent transaction
cover_ar_trgrp_Maintenance	Cover read maintenance transaction
cover_ar_trgrp_DVM	Cover read DVM transaction
cover_ar_trgrp_Barrier	Cover read Barrier transaction
cover_aw_trgrp_NonSnoop	Cover write non snoop transaction
cover_aw_trgrp_Coherent	Cover write coherent transaction
cover_aw_trgrp_Memory	Cover write maintenance transaction
cover_aw_trgrp_Barrier	Cover write DVM transaction
cover_arsnoop_ReadNoSnoop	Cover ReadNoSnoop transaction
cover_arsnoop_ReadOnce	Cover ReadOnce transaction
cover_arsnoop_ReadShared	Cover ReadShared transaction
cover_arsnoop_ReadClean	Cover ReadClean transaction
cover_arsnoop_ReadNotSharedDirty	Cover ReadNotSharedDirty transaction
cover_arsnoop_ReadUnique	Cover ReadUnique transaction
cover_arsnoop_CleanUnique	Cover CleanUnique transaction
cover_arsnoop_MakeUnique	Cover MakeUnique transaction
cover_arsnoop_CleanShared	Cover CleanShared transaction
cover_arsnoop_CleanInvalid	Cover CleanInvalid transaction
cover_arsnoop_MakeInvalid	Cover MakeInvalid transaction
cover_arsnoop_DVM_Complete	Cover DVM Complete transaction
cover_arsnoop_DVM_Message	Cover DVM Message transaction
cover_arsnoop_Barrier	Cover Barrier transaction
cover_rsnoop_ReadNoSnoop_finished	Cover ReadNoSnoop transaction finished
cover_rsnoop_ReadOnce_finished	Cover ReadOnce transaction finished
cover_rsnoop_ReadShared_finished	Cover ReadShared transaction finished
cover_rsnoop_ReadClean_finished	Cover ReadClean transaction finished
cover_rsnoop_ReadNotSharedDirty_finished	Cover ReadNotSharedDirty transaction finished

AXI ABVIP User Guide
Appendix E: ACE5 ABVIP Package Contents--Coverage Checks

cover_rsnoop_ReadUnique_finished	Cover ReadUnique transaction finished
cover_rsnoop_CleanUnique_finished	Cover CleanUnique transaction finished
cover_rsnoop_MakeUnique_finished	Cover MakeUnique transaction finished
cover_rsnoop_CleanShared_finished	Cover CleanShared transaction finished
cover_rsnoop_CleanInvalid_finished	Cover CleanInvalid transaction finished
cover_rsnoop_MakeInvalid_finished	Cover MakeInvalid transaction finished
cover_rsnoop_DVM_Complete_finished	Cover DVM Complete transaction finished
cover_rsnoop_DVM_Message_finished	Cover DVM Message transaction finished
cover_rsnoop_Barrier_finished	Cover Barrier transaction finished
cover_awsnoop_WriteNoSnoop	Cover WriteNoSnoop transaction
cover_awsnoop_WriteUnique	Cover WriteUnique transaction
cover_awsnoop_WriteLineunique	Cover WriteLineUnique transaction
cover_awsnoop_WriteClean	Cover WriteClean transaction
cover_awsnoop_WriteBack	Cover WriteBack transaction
cover_awsnoop_Evict	Cover Evict transaction
cover_awsnoop_Barrier	Cover Barrier transaction
cover_bsnoop_WriteNoSnoop_finished	Cover WriteNoSnoop transaction finished
cover_bsnoop_WriteUnique_finished	Cover WriteUnique transaction finished
cover_bsnoop_WriteLineunique_finished	Cover WriteLineUnique transaction finished
cover_bsnoop_WriteClean_finished	Cover WriteClean transaction finished
cover_bsnoop_WriteBack_finished	Cover WriteBack transaction finished
cover_bsnoop_WriteBack_finished	Cover WriteBack transaction finished
cover_bsnoop_Barrier_finished	Cover Barrier transaction finished
cover_arbar_NormalRespect	Cover proper read barrier combination
cover_arbar_MemoryBarrier	Cover proper read barrier combination
cover_arbar_NormalIgnore	Cover proper read barrier combination
cover_arbar_SyncBarrier	Cover proper read barrier combination
cover_awbar_NormalRespect	Cover proper write barrier combination
cover_awbar_MemoryBarrier	Cover proper write barrier combination
cover_awbar_NormalIgnore	Cover proper write barrier combination
cover_awbar_SyncBarrier	Cover proper write barrier combination
cover_rresp_PassDirty	Cover rresp with PassDirty bit set
cover_rresp_PassClean	Cover rresp with PassClean bit set
cover_rresp_IsShared	Cover rresp with IsShared bit set
cover_rresp_IsUnique	Cover rresp with IsUnique bit set
cover_acprot_privileged	Cover proper acprot combination

AXI ABVIP User Guide
Appendix E: ACE5 ABVIP Package Contents--Coverage Checks

cover_acprot_normal	Cover proper acprot combination
cover_acprot_nonsecure	Cover proper acprot combination
cover_acprot_secure	Cover proper acprot combination
cover_acprot_instruction	Cover proper acprot combination
cover_acprot_data	Cover proper acprot combination
cover_acsnoop_ReadOnce	Cover ReadOnce snoop transaction
cover_acsnoop_ReadShared	Cover ReadShared snoop transaction
cover_acsnoop_ReadClean	Cover ReadClean snoop transaction
cover_acsnoop_ReadNotSharedDirty	Cover ReadNotSharedDirty snoop transaction
cover_acsnoop_ReadUnique	Cover ReadUnique snoop transaction
cover_acsnoop_CleanShared	Cover CleanShared snoop transaction
cover_acsnoop_CleanInvalid	Cover CleanInvalid snoop transaction
cover_acsnoop_MakeInvalid	Cover MakeInvalid snoop transaction
cover_acsnoop_DVM_Complete	Cover DVM Complete snoop transaction
cover_acsnoop_DVM_Message	Cover DVM Message snoop transaction
cover_crresp_DataTransfer	Cover crresp with data transfer bit set
cover_crresp_NoDataTransfer	Cover crresp with no data transfer
cover_crresp_Error	Cover crresp with error
cover_crresp_NoError	Cover crresp with no error
cover_crresp_PassDirty	Cover crresp with PassDirty bit set
cover_crresp_PassClean	Cover crresp with PassClean bit set
cover_crresp_IsShared	Cover crresp with IsShared bit set
cover_crresp_IsUnique	Cover crresp with IsUnique bit set
cover_crresp_WasUnique	Cover crresp with WasUnique bit set
cover_crresp_WasShared	Cover crresp with WasShared bit set
cover_rack	Cover rack
cover_wack	Cover wack
cover_seq_bar_bresp_out_of_order	Cover barrier transaction with out of order response
cover_seq_evict_bresp_in_order	Cover evict transaction with in order response
cover_seq_normal_evict_pair	Cover normal and evict transaction pair with response in order
cover_seq_normal_barrier_pair	Cover barrier and normal transaction with out of order response (awlen=0)
cover_seq_normal_len2_barrier_pair	Cover barrier and normal transaction with out of order response (awlen=1)
cover_seq_snoop_dbr_order	Cover snoop data before request
cover_seq_dbr_barrier_collision	Cover DBC barrier collision
cover_seq_dbr_len2_barrier_pair	Cover DBC barrier pair with out of order response

AXI ABVIP User Guide
Appendix E: ACE5 ABVIP Package Contents--Coverage Checks

cover_data_chunking_aligned	Covers Read Data Chunking with DATA_WIDTH 256
cover_data_chunking_unaligned	Covers Read Data Chunking with DATA_WIDTH 256 with unaligned address
cover_unq11_rd_diff_id	Covers for unique ID indication, two transaction with high aridunq signal
cover_unq10_rd_diff_id	Covers for unique ID indication, two transaction with one high and one low aridunq signal
cover_unq01_rd_diff_id	Covers for unique ID indication, two transaction with one high and one low aridunq signal
cover_unq00_rd_diff_id	Covers for unique ID indication, two transaction with both low aridunq signal
cover_unq_rd_tx_same_cycle	Covers for unique ID indication, two read transaction at same cycle
cover_unq_rd_transaction	Cover for whole write transaction with aridunq
cover_unq_rd_tx_diff_cycle	Covers for unique ID indication, two read transaction at different cycle
cover_unq_aridunq_ridunq_same_low	Read request sends low aridunq, read response has low ridunq signal in return
cover_unq_aridunq_ridunq_same_high	Read request sends high aridunq, read response has high ridunq signal in return
cover_unq11_wr_diff_id	Covers for unique ID indication, two transaction with high awidunq signal
cover_unq10_wr_diff_id	Covers for unique ID indication, two transaction with one high and one low awidunq signal
cover_unq01_wr_diff_id	Covers for unique ID indication, two transaction with one high and one low awidunq signal
cover_unq00_wr_diff_id	Covers for unique ID indication, two transaction with both low awidunq signal
cover_unq_wr_tx_same_cycle	Covers for unique ID indication, two write transaction at same cycle
cover_unq_wr_transaction	Cover for whole write transaction with awidunq
cover_unq_wr_tx_diff_cycle	Covers for unique ID indication, two write transaction at different cycle
cover_unq_awidunq_bidunq_same_low	write request sends low awidunq, write response has low bidunq signal in return
cover_unq_awidunq_bidunq_same_high	write request sends high awidunq, write response has high bidunq signal in return
2.4 MTE Covers	
cover_invalid_tag_op_wrtransaction	Covers for invalid tag transaction
cover_update_tag_op_wrtransaction	Covers for update tag transaction
cover_wtag_wvalid_comb	Covers for all possible values of wtag
cover_wtag_wvalid_comb	Covers for all possible values of wtagupdate
cover_transfer_tag_op_wrtransaction	Covers for transfer tag transaction
cover_match_tag_op_wrtransaction	Covers for whole tagmatch transaction
cover_bvalid_bcomp_btagmatch_pass	Covers for pass match and combined response
cover_bvalid_btagmatch_pass	Covers for btagmatch pass response part
cover_bvalid_bcomp	Covers for match response part 1
cover_bvalid_bcomp_btagmatch_fail	Covers for fail match and combined response
cover_bvalid_btagmatch_fail	Covers for btagmatch fail response part
cover_bvalid_btagmatch	Covers for btagmatch response part
cover_invalid_tag_op_rdtransaction	Covers for INVALID tag operation
cover_transfer_tag_op_rdtransaction	Covers for TRANSFER tag operation
cover_transfer_tag_op_rdtransaction	Covers for FETCH tag operation

AXI ABVIP User Guide
Appendix E: ACE5 ABVIP Package Contents--Coverage Checks

cover_rtag_rvalid_comb	Covers for every value of rtag
------------------------	--------------------------------