AMBA® 4 ACE and ACE-Lite Protocol Checkers

Revision: r0p0

User Guide

ARM®

ARM DUI 0576A (ID092411)

AMBA 4 ACE and ACE-Lite Protocol Checkers
User Guide

Release Information

The following changes have been made to this book.

Change history

Date	Issue	Confidentiality	Change
30 August 2011	A	Non-Confidential	First issue for r0p0

Proprietary Notice

Confidentiality Status

This document is Non-Confidential. The right to use, copy and disclose this document may be subject to license restrictions in accordance with the terms of the agreement entered into by ARM and the party that ARM delivered this document to.

Product Status

The information in this document is final, that is for a developed product.

Web Address

`http://www.arm.com`

Contents
AMBA 4 ACE and ACE-Lite Protocol Checkers User Guide

Preface

This preface introduces the *AMBA® 4 ACE and ACE-Lite Protocol Checkers User Guide*. It contains the following sections:

About this book

This is the *User Guide* for the *AMBA® 4 ACE and ACE-Lite Protocol Checkers*.

Intended audience

This book is written for system designers, system integrators, and verification engineers who want to confirm that a design complies with the relevant AMBA4 protocol. This can be ACE or ACE-Lite.

Using this book

This book is organized into the following chapters:

Chapter 1 *Introduction*

Read this for a high-level description of the protocol checkers.

Chapter 2 *Implementation and Integration*

Read this for a description of where to locate the protocol checkers in your design, the integration flow, information about specific signal connections with an example file listing, and setting up your simulator.

Chapter 3 *Parameter Descriptions*

Read this for a description of the protocol checkers parameters.

Chapter 4 *ACE and ACE-Lite Protocol Assertion Descriptions*

Read this for a description of the protocol checkers module.

Appendix A *Example Usage*

Read this for an example of a design that does not comply with the protocol.

Appendix B *Revisions*

Read this for a description of the technical changes between released issues of this book.

Conventions

Conventions that this book can use are described in:

- *Typographical* on page vii
- *Timing diagrams* on page vii
- *Signals* on page vii.

Typographical

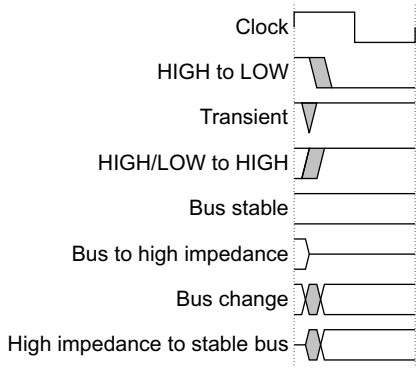
The typographical conventions are:

italic	Highlights important notes, introduces special terminology, denotes internal cross-references, and citations.
bold	Highlights interface elements, such as menu names. Denotes signal names. Also used for terms in descriptive lists, where appropriate.
`monospace`	Denotes text that you can enter at the keyboard, such as commands, file and program names, and source code.
`mono`space	Denotes a permitted abbreviation for a command or option. You can enter the underlined text instead of the full command or option name.
`monospace italic`	Denotes arguments to monospace text where the argument is to be replaced by a specific value.
`monospace bold`	Denotes language keywords when used outside example code.
< and >	Enclose replaceable terms for assembler syntax where they appear in code or code fragments. For example: `MRC p15, 0 <Rd>, <CRn>, <CRm>, <Opcode_2>`

Timing diagrams

The figure named *Key to timing diagram conventions* explains the components used in timing diagrams. Variations, when they occur, have clear labels. You must not assume any timing information that is not explicit in the diagrams.

Shaded bus and signal areas are undefined, so the bus or signal can assume any value within the shaded area at that time. The actual level is unimportant and does not affect normal operation.



Key to timing diagram conventions

Signals

The signal conventions are:

Signal level The level of an asserted signal depends on whether the signal is active-HIGH or active-LOW. Asserted means:

- HIGH for active-HIGH signals
- LOW for active-LOW signals.

Lower-case n At the start or end of a signal name denotes an active-LOW signal.

Additional reading

This section lists publications by ARM and by third parties.

See Infocenter, http://infocenter.arm.com, for access to ARM documentation.

See the glossary, http://infocenter.arm.com/help/topic/com.arm.doc.aeg0014-/index.html, for a list of terms and acronyms specific to ARM.

ARM publications

This book contains information that is specific to this product. See the following document for other relevant information:

- *AMBA® AXI and ACE Protocol Specification - AXI3, AXI4, and AXI4-Lite, ACE and ACE-Lite* (ARM IHI 0022D-2c)

Other publications

This section lists relevant documents published by third parties:

- SystemVerilog technical papers, tutorials, and downloads, http://www.systemverilog.org/
- Accellera *SystemVerilog 3.1a Language Reference Manual*, http://www.eda.org/sv/
- 1800-2005 *IEEE Standard for SystemVerilog: Unified Hardware Design, Specification and Verification Language*, http://www.systemverilog.org.

Feedback

ARM welcomes feedback on this product and its documentation.

Feedback on this product

If you have any comments or suggestions about this product, contact your supplier and give:

- The product name.
- The product revision or version.
- An explanation with as much information as you can provide. Include symptoms and diagnostic procedures if appropriate.

Feedback on content

If you have comments on content then send an e-mail to `errata@arm.com`. Give:

- the title
- the number, ARM DUI 0576A
- the page numbers to which your comments apply
- a concise explanation of your comments.

ARM also welcomes general suggestions for additions and improvements.

Chapter 1
Introduction

This chapter introduces the protocol checkers. It contains the following sections:

1.1 About the protocol checkers

You can use the protocol checkers that this user guide describes to test the ACE and ACE-Lite protocols that the *AMBA® AXI and ACE Protocol Specification - AXI3, AXI4, and AXI4-Lite, ACE, and ACE-Lite* defines. See *ARM publications* on page viii.

The behavior of the interface you test is checked against the protocol by a series of assertions.

This guide describes the contents of the SystemVerilog files, and how to integrate them into a design. It also describes the correct use of these assertions with simulators to flag errors, warnings, or both during design simulation.

1.2 Tools

The protocol checkers are written in SystemVerilog. SystemVerilog is a *Hardware Description and Verification Language* (HDVL) standard that extends the established Verilog language. SystemVerilog was developed to improve productivity in the design of large gate count, IP-based, bus-intensive chips. SystemVerilog is targeted at the chip implementation and verification flow, with links to the system level design flow.

Note

The version of System Verilog supported is IEEE 1800-2005.

Chapter 2
Implementation and Integration

This chapter describes the location of the protocol checkers and the integration flow. It contains the following sections:

2.1 Implementation and integration flow

Figure 2-1 shows the design flow for implementing and integrating the protocol checkers SystemVerilog file with a design.

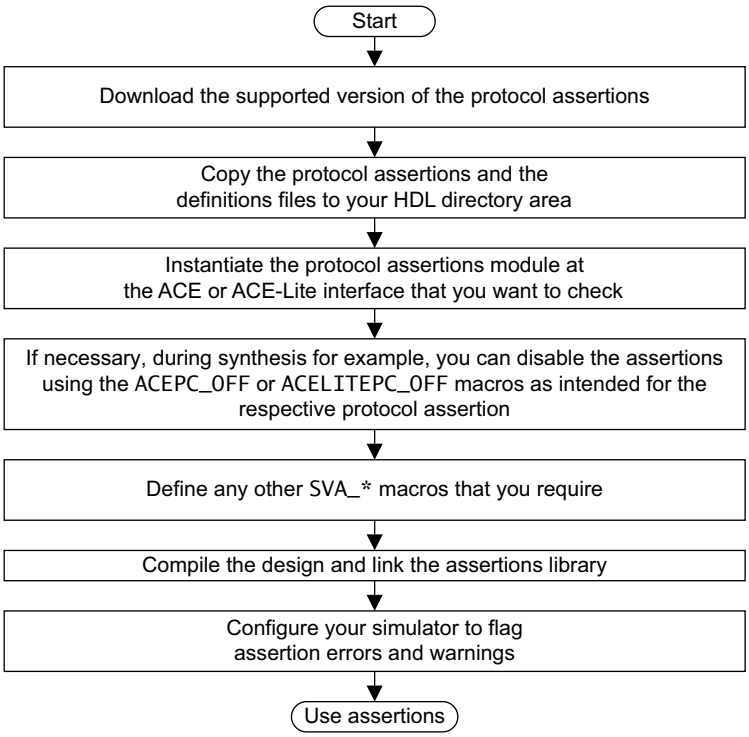


Figure 2-1 Integration flow

2.2 Implementing the protocol checkers in your design directory

You can implement the protocol checkers for:

- ACE
- ACE-Lite.

This section describes:

- *ACE and ACE-Lite protocol checker files*
- *Location of ACE and ACE-Lite protocol checker files*.

2.2.1 ACE and ACE-Lite protocol checker files

Figure 2-2 shows the contents of the directory that contains the protocol checkers. It shows the files that are required for each of the different protocols, ACE and ACE-Lite.

```
sva/
  ├──Axi4PC_ace.sv
  ├──Axi4PC_ace_defs.v
  ├──Axi4PC_ace_message_defs.v
  ├──Axi4PC_ace_message_undefs.v
  ├──Axi4PC_ace_undefs.v
  ├──AceLitePC.sv
  ├──AcePC.sv
  ├──AcePC_message_defs.v
  ├──AcePC_message_undefs.v
  ├──AcePC_defs.v
  ├──AcePC_undefs.v
  └──files.vc
```

Figure 2-2 Protocol checkers directory structure for ACE and ACE-Lite

2.2.2 Location of ACE and ACE-Lite protocol checker files

Figure 2-3 shows the location of the protocol checker SystemVerilog files in your design RTL.

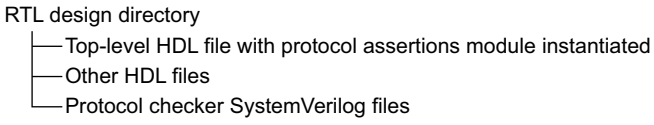


Figure 2-3 Location of the ACE and ACE-Lite protocol checker SystemVerilog files

2.3 Instantiating the protocol checker modules

The protocol checker modules contain a port list. Connect the ACE or ACE-Lite module ports to the corresponding signals in your design.

See *Example Verilog file listing for ACE protocol checker instantiation* for an example that shows how the module is instantiated in a top-level Verilog file.

See *ARM publications* on page viii for the specifications that describe the ACE or ACE-Lite signals.

The low-power interface signals of the AXI interface are defined as weak pull-up and you must tie them LOW. They are named:

CSYSREQ For the low-power request signal.

CSYSACK For the low-power request acknowledgement signal.

CACTIVE For the clock active signal.

The SystemVerilog files contain checks for user-configurable sideband signals. You must tie these signals LOW.

This section describes:

- *Example Verilog file listing for ACE protocol checker instantiation*
- *Example Verilog file listing for ACE-Lite protocol checker instantiation* on page 2-5.

2.3.1 Example Verilog file listing for ACE protocol checker instantiation

Example 2-1 shows part of a design HDL file instantiating the protocol checker module for ACE. You can, if necessary, override any of the protocol checker parameters by using `defparam` at this level.

Example 2-1 Example Verilog file listing for ACE

```
AcePC u_ace4_sva
      (
      .ACLK (ACLK),
      .ARESETn (ARESETn),
      .AWID (AWID),
      .AWADDR (AWADDR),
      .AWLEN (AWLEN),
      .AWSIZE (AWSIZE),
      .AWBURST (AWBURST),
      .AWLOCK (AWLOCK),
      .AWCACHE (AWCACHE),
      .AWPROT (AWPROT),
      .AWDOMAIN (AWDOMAIN),
      .AWSNOOP (AWSNOOP),
      .AWBAR (AWBAR),
      .AWQOS (AWQOS),
      .AWREGION (AWREGION),
      .AWUSER (AWUSER),
      .AWVALID (AWVALID),
      .AWREADY (AWREADY),
      .WLAST (WLAST),
      .WDATA (WDATA),
      .WSTRB (WSTRB),
      .WUSER (WUSER),
      .WVALID (WVALID),
```

```
.WREADY (WREADY),
.WACK (WACK),
.BID (BID),
.BRESP (BRESP),
.BUSER (BUSER),
.BVALID (BVALID),
.BREADY (BREADY),
.ARID (ARID),
.ARADDR (ARADDR),
.ARLEN (ARLEN),
.ARSIZE (ARSIZE),
.ARBURST (ARBURST),
.ARLOCK (ARLOCK),
.ARCACHE (ARCACHE),
.ARPROT (ARPROT),
.ARDOMAIN (ARDOMAIN),
.ARSNOOP (ARSNOOP),
.ARBAR (ARBAR),
.ARQOS (ARQOS),
.ARREGION (ARREGION),
.ARUSER (ARUSER),
.ARVALID (ARVALID),
.ARREADY (ARREADY),
.RID (RID),
.RLAST (RLAST),
.RDATA (RDATA),
.RRESP (RRESP),
.RUSER (RUSER),
.RVALID (RVALID),
.RREADY (RREADY),
.RACK (RACK),
.ACVALID (ACVALID),
.ACREADY (ACREADY),
.ACADDR (ACADDR),
.ACSNOOP (ACSNOOP),
.ACPROT (ACPROT),
.CRVALID (CRVALID),
.CRREADY (CRREADY),
.CRRESP (CRRESP),
.CDVALID (CDVALID),
.CDREADY (CDREADY),
.CDDATA (CDDATA),
.CDLAST (CDLAST),
.CACTIVE (CACTIVE),
.CSYSREQ (CSYSREQ),
.CSYSACK (CSYSACK)
);
```

2.3.2 Example Verilog file listing for ACE-Lite protocol checker instantiation

Example 2-2 shows part of a design HDL file instantiating the protocol checker module for ACE-Lite. You can, if necessary, override any of the protocol checker parameters by using `defparam` at this level.

Example 2-2 Example Verilog file listing for ACE-Lite

```
Ace4LitePC u_acelite_sva
          (
```

```
.ACLK (ACLK),
.ARESETn (ARESETn),
.ARDOMAIN (ARDOMAIN),
.ARSNOOP (ARSNOOP),
.ARBAR (ARBAR),
.AWADDR (AWADDR),
.AWPROT (AWPROT),
.AWVALID (AWVALID),
.AWREADY (AWREADY),
.AWDOMAIN (AWDOMAIN),
.AWSNOOP (AWSNOOP),
.AWBAR (AWBAR),
.WDATA (WDATA),
.WSTRB (WSTRB),
.WVALID (WVALID),
.WREADY (WREADY),
.BRESP (BRESP),
.BVALID (BVALID),
.BREADY (BREADY),
.ARADDR (ARADDR),
.ARPROT (ARPROT),
.ARVALID (ARVALID),
.ARREADY (ARREADY),
.RDATA (RDATA),
.RRESP (RRESP),
.RVALID (RVALID),
.RREADY (RREADY)
);
```

2.4 Configuring your simulator

Most simulators support the use of assertions in RTL, and enable you to configure the simulator appropriately using command variables that define the available assertion options. These can include:

- suppress or enable assertion warnings
- select assertion report messages to display
- set a minimum severity level for which assertion report messages are output
- set a minimum severity level for which an assertion causes the simulator to stop.

The protocol checkers are written using SystemVerilog version 3.1a, and are tested with a number of simulators. Contact your simulator supplier and see your documentation for more information on using SystemVerilog Assertions.

Chapter 3
Parameter Descriptions

This chapter provides descriptions of the protocol checkers parameters. It contains the following sections:

Caution

An additional set of defined parameters are derived from the base set of parameters that this chapter describes. Do not modify them.

3.1 ACE and ACE-Lite interfaces

Table 3-1 shows the user-defined parameters for setting the interface characteristics for ACE and ACE-Lite. Change them to match your design specification.

Table 3-1 Interface parameters for ACE and ACE-Lite

Name	Description	ACE default	ACE-Lite default
DATA_WIDTH	Width of the system data buses.	64	64
ADDR_WIDTH	Width of the address bus.	64	64
CD_DATA_WIDTH	Width of the snoop data bus.	DATA_WIDTH	-
RID_WIDTH	Number of read channel ID bits required.	4	4
WID_WIDTH	Number of write channel ID bits required.	4	4
MAXRBURSTS	Size of FIFOs for storing outstanding read bursts. This must be equal to or greater than the maximum number of outstanding read bursts on the interface.	16	16
MAXWBURSTS	Size of FIFOs for storing outstanding write bursts. This must be equal to or greater than the maximum number of outstanding write bursts on the interface.	16	16
MAXCBURSTS	Size of FIFOs for storing outstanding snoop transactions. This must be equal to or greater than the maximum number of snoop transactions on the interface.	64	-
AWUSER_WIDTH	Width of the user AW sideband field.	32	32
WUSER_WIDTH	Width of the user W sideband field.	32	32
BUSER_WIDTH	Width of the user B sideband field.	32	32
ARUSER_WIDTH	Width of the user AR sideband field.	32	32
RUSER_WIDTH	Width of the user R sideband field.	32	32
CACHE_LINE_SIZE_BYTES	Cache line size, in bytes.	64	64
MAX_BARRIER	Maximum number of barriers on the ACE-Lite interface.	-	256
SINGLE_EXCL	Configures the interface to support only single exclusive threads.	1'b1	-
EXMON_WIDTH	Width of the exclusive access monitor required. The maximum ID width monitored for AXI3-style exclusive accesses.	4	4

3.2 Performance checking

Table 3-2 shows the user-defined parameter for performance checking.

Table 3-2 Performance checking parameter

Name	Description	Default
MAXWAITS	Maximum number of cycles between **VALID** to **READY** HIGH before a warning is generated	16

3.3 Disabling recommended rules

Table 3-3 shows the user-defined parameters for disabling recommended rules from the protocol checkers.

Table 3-3 Display parameters

Name	Description	Default
RecommendOn	Enable or disable reporting of protocol recommendations	1'b1, enabled
RecMaxWaitOn	Enable or disable the recommended MAX_WAIT rules	1'b1, enabled
RecommendOn_SW	Enable or disable the recommended software rules	1'b1, enabled

Note

RecMaxWaitOn is a subset of RecommendOn, and if RecommendOn is 1'b0, that is, disabled, then the MAX_WAIT rules are disabled regardless of the settings of RecMaxWaitOn.

If RecommendOn is disabled, the following warning is issued:

```
ACE_WARN: All recommended ACE rules have been disabled by the RecommendOn parameter
```

If RecommendOn_SW is disabled, the following warning is issued:

```
ACE_WARN: All recommended ACE software rules have been disabled by the RecommendOn_SW parameter
```

If RecommendOn is enabled, the default, but RecMaxWaitOn is disabled, the following warning is issued:

```
ACE_WARN: Five recommended MAX_WAIT rules have been disabled by the RecMaxWaitOn parameter
```

3.4 X-check rules

If you want to disable the X-propagation assertions on ACE or ACE-Lite interfaces, you must use the following rule when compiling:

```
+define+AXI4_XCHECK_OFF
```

3.5 Disabling protocol checkers

In circumstances where the protocol assertion module has been automatically inserted in a testbench, and you want to disable it without editing the testbench, you can compile with the following options:

`+define+ACE4PC_OFF`

To disable the `AcePC` protocol checker module.

`+define+ACELITEPC_OFF`

To disable the `AceLitePC` protocol checker module.

Chapter 4
ACE and ACE-Lite Protocol Assertion Descriptions

This chapter describes the protocol assertions and indicates the area of the *AMBA® AXI and ACE Protocol Specification - AXI3, AXI4, and AXI4-Lite, ACE and ACE-Lite* to which they apply.

It contains the following sections:

4.1 AW channel

Table 4-1 shows the AW channel protocol rules.

Table 4-1 AW channel

Assertion	Description	AMBA AXI and ACE Protocol Specification reference
ACE_ERRM_AWDOMAIN_STABLE	**AWDOMAIN** remains stable when **AWVALID** is asserted and **AWREADY** is LOW.	A3.2.1
ACE_ERRM_AWSNOOP_STABLE	**AWSNOOP** remains stable when **AWVALID** is asserted and **AWREADY** is LOW.	A3.2.1
ACE_ERRM_AWBAR_STABLE	**AWBAR** remains stable when **AWVALID** is asserted and **AWREADY** is LOW.	A3.2.1
ACE_ERRM_AWDOMAIN_X	A value of X on **AWDOMAIN** is not permitted when **AWVALID** is HIGH.	A3.2.2
ACE_ERRM_AWSNOOP_X	A value of X on **AWSNOOP** is not permitted when **AWVALID** is HIGH.	A3.2.2
ACE_ERRM_AWBAR_X	A value of X on **AWBAR** is not permitted when **AWVALID** is HIGH.	A3.2.2
ACE_ERRM_AWSNOOP	When **AWVALID** is HIGH, **AWSNOOP** must be one of the non-reserved values.	Table C3-7
ACE_ERRM_AWCACHE_DEVICE	When **AWVALID** is HIGH, if **AWCACHE[1]** is LOW, non-modifiable, device, **AWDOMAIN** must be 'b11, system shareable.	Table C3-3
ACE_ERRM_AWCACHE_SYSTEM	When **AWVALID** is HIGH, if **AWCACHE[3:2]** is not 'b00, then **AWDOMAIN** must not be 'b11, system.	Table C3-3
ACE_ERRM_AW_DOMAIN_1	For a WriteBack or a WriteClean, **AWDOMAIN** must not be system shareable.	Table C3-8
ACE_ERRM_AW_DOMAIN_2	For a WriteLineUnique or Evict the domain must be to the inner or outer domain.	Table C3-8
ACE_ERRM_AW_SHAREABLE_ALIGN_INCR	WriteLineUnique and Evict transactions of type INCR must be aligned to the cache line size.	Table C3-8
ACE_ERRM_AW_SHAREABLE_ALIGN_WRAP	WriteLineUnique and Evict transactions of type WRAP must be aligned to the width of the bus.	Table C3-8
ACE_ERRM_AW_BLOCK_1	When a WriteUnique or WriteLineUnique is in progress, AW to B, a master must not issue a WriteBack or WriteClean, AW to B.	C4.8.6
ACE_ERRM_AW_BLOCK_2	When a WriteBack or WriteClean, AW to B, is in progress, a master must not issue a WriteUnique or WriteLineUnique, AW to B.	C4.8.6
ACE_ERRM_AW_FULL_LINE	WriteLineUnique and Evict transactions are required to be a full cache line size.	C3.1.5, C6.7.2

Table 4-1 AW channel (continued)

Assertion	Description	AMBA AXI and ACE Protocol Specification reference
ACE_ERRM_AW_SHAREABLE_CTL	WriteUnique, WriteLineUnique, WriteBack, WriteClean, Evict transactions are required to have the following values: **AWBURST** != FIXED **AWCACHE[1]** = 'b1 **AWLOCK** = 'b0 **AWBAR[0]** = 'b0.	Tables C3-10, C3-11, C3-12
ACE_ERRM_WB_WC_CACHE_LINE_BOUNDARY_INCR	For a WriteBack or a WriteClean, the transaction must not cross a cache line boundary. The final beat of a transaction must fall within the same cache line as the first.	Table C3-12
ACE_ERRM_WB_WC_CACHE_LINE_BOUNDARY_WRAP	For a WriteBack or a WriteClean, the transaction must not cross a cache line boundary. The total number of bytes must not exceed the cache line size.	Table C3-12
ACE_ERRM_AW_NORMAL_ID	A normal transaction that shares its ID with an outstanding barrier or DVM message must not be issued.	C8.4.1
ACE_ERRM_AW_SHAREABLE_LOCK	All shareable accesses, as indicated by **AxDOMAIN** = Inner Shareable or Outer Shareable, must use **AxLOCK** = 0.	Tables C3-10, C3-11, C3-12

4.2 AR channel

Table 4-2 shows the AR channel protocol rules.

Table 4-2 AR channel

Assertion	Description	AMBA AXI and ACE Protocol Specification reference
ACE_ERRM_ARDOMAIN_STABLE	**ARDOMAIN** remains stable when **ARVALID** is asserted and **ARREADY** is LOW	A3.2.1
ACE_ERRM_ARSNOOP_STABLE	**ARSNOOP** remains stable when **ARVALID** is asserted and **ARREADY** is LOW	A3.2.1
ACE_ERRM_ARBAR_STABLE	**ARBAR** remains stable when **ARVALID** is asserted and **ARREADY** is LOW	A3.2.1
ACE_ERRM_ARDOMAIN_X	A value of X on **ARDOMAIN** is not permitted when **ARVALID** is HIGH	A3.2.2
ACE_ERRM_ARSNOOP_X	A value of X on **ARSNOOP** is not permitted when **ARVALID** is HIGH	A3.2.2
ACE_ERRM_ARBAR_X	A value of X on **ARBAR** is not permitted when **ARVALID** is HIGH	A3.2.2
ACE_ERRM_ARSNOOP	When **ARVALID** is HIGH, **ARSNOOP** must be one of the non-reserved values	Table C3-7
ACE_ERRM_ARCACHE_DEVICE	When **ARVALID** is HIGH, if **ARCACHE[1]** is Low, non-modifiable, device, **ARDOMAIN** must be 2'b11, system shareable	Table C3-3
ACE_ERRM_ARCACHE_SYSTEM	When **ARVALID** is HIGH, if **ARCACHE[3:2]** is not 2'b00, then **ARDOMAIN** must not be 2'b11, system	Table C3-3
ACE_ERRM_AR_DOMAIN_1	CleanInvalid, CleanShared, and MakeInvalid transactions must not be system shareable	Table C3-10
ACE_ERRM_AR_DOMAIN_2	ReadShared, ReadClean, ReadNotSharedDirty, ReadUnique, CleanUnique, and MakeUnique transactions must be to the inner or outer domain	Table C3-10
ACE_ERRM_AR_SHAREABLE_ALIGN_INCR	ReadShared, ReadClean, ReadNotSharedDirty, Readunique, CleanInvalid, CleanShared, CleanUnique, MakeUnique, and MakeInvalid transactions of type INCR must be aligned to the cache line size	Table C3-10
ACE_ERRM_AR_FULL_LINE	ReadShared, ReadClean, ReadNotSharedDirty, Readunique, CleanInvalid, CleanShared, CleanUnique, MakeUnique, and MakeInvalid transactions are required to be a full cache line size	Table C3-10
ACE_ERRM_AR_SHAREABLE_CTL	ReadShared, ReadClean, ReadNotSharedDirty, Readunique, CleanInvalid, CleanShared, CleanUnique, MakeUnique, and MakeInvalid transactions must have the following properties: **ARSIZE** == Data bus width **ARBURST** != FIXED **ARBAR[0]** == 'b0 **ARCACHE[1]** = 'b1.	Table C3-10
ACE_ERRM_AR_NORMAL_ID	A normal transaction that shares its ID with an outstanding barrier or DVM message must not be issued	C8.4.1, C12.3.5
ACE_ERRM_AR_SHAREABLE_LOCK	ReadNotSharedDirty, ReadOnce, Readunique, CleanInvalid, CleanShared, MakeUnique, and MakeInvalid must have **ARLOCK** = 1'b0	Table C3-10

4.3 W channel

Table 4-3 shows the W channel protocol rules.

Table 4-3 W channel

Assertion	Description	AMBA AXI and ACE Protocol Specification reference
ACE_ERRM_WLU_STRB	WriteLineUnique transactions are not permitted to have sparse strobes	C3.1.5

4.4 R channel

Table 4-4 shows the R channel protocol rules.

Table 4-4 R channel

Assertion	Description	AMBA AXI and ACE Protocol Specification reference
ACE_ERRS_RRESP_SHARED	A slave must not give an IsShared (**RESP[3]** = 'b1) response to a ReadNoSnoop, ReadUnique, CleanUnique, CleanInvalid, MakeInvalid, or MakeUnique transaction.	C3.2.1
ACE_ERRS_RRESP_DIRTY	A slave must not give a PassDirty (**RESP[2]** = 1'b1) response to a ReadNoSnoop, ReadOnce, ReadClean, CleanUnique, CleanInvalid, MakeUnique, MakeInvalid, or CleanShared transaction.	C3.2.1
ACE_ERRS_RRESP_RNSD	A slave must not give an IsShared and PassDirty response to a ReadNotSharedDirty.	C3.2.1
ACE_ERRS_RDATALESS	A slave must return exactly one data beat for Make or Clean transactions.	C3.2.1
ACE_ERRM_RACK_X	When not in reset, a value of X on **RACK** is not permitted.	C3.3
ACE_ERRS_RRESP_BAR	A barrier response must use **RRESP** = 'b0000.	Table C8-2
ACE_ERRS_RRESP_CONST	**RRESP[3:2]** are required to be constant for all data transfers within a burst.	C3.2.1
ACE_ERRS_RRESP_ACE_EXOKAY	The EXOKAY response is only permitted for ReadNoSnoop, ReadClean, ReadShared, and CleanUnique transactions.	C3.2.1
ACE_ERRM_RACK	**RACK** is not asserted until at least the cycle after the **RLAST** handshake. There must be only one **RACK** for each **RLAST** handshake.	C3.3
ACE_ERRS_RRESP_DVM	A DVM response must use **RRESP[3:2]** = 2'b00 && **RRESP[0]** = 'b0	C12.3.4
ACE_ERRS_RRESP_DVM_ERROR	An error response is not permitted for a: • DVM Sync • DVM Complete • DVM Hint.	C12.3.4, C12.7.6

4.5 B channel

Table 4-5 shows the B channel protocol rules.

Table 4-5 B channel

Assertion	Description	AMBA AXI and ACE Protocol Specification reference
ACE_ERRM_WACK	**WACK** is not asserted until at least the cycle after the B channel handshake. There must be only one **WACK** for each B handshake.	C3.5
ACE_ERRM_WACK_X	When not in reset, a value of X on **WACK** is not permitted.	C3.5
ACE_ERRS_BRESP_BAR	Barrier responses must be 'b00.	Table C8-2
ACE_ERRS_BRESP_WNS_EXOKAY	The EXOKAY response is only permitted for a WriteNoSnoop transaction.	C8.2.3
ACE_ERRS_BRESP_AW_WLAST	**BVALID** is not permitted until after the AW handshake and **WLAST**.	Table C8-2

4.6 Reset

Table 4-6 shows the reset channel protocol rules.

Table 4-6 Reset

Assertion	Description	AMBA AXI and ACE Protocol Specification reference
ACE_ERRS_ACVALID_RESET	**ACVALID** must be LOW for the first cycle after **ARESETn** goes HIGH	C2.3.3
ACE_ERRM_CRVALID_RESET	**CRVALID** must be LOW for the first cycle after **ARESETn** goes HIGH	C2.3.3
ACE_ERRM_CDVALID_RESET	**CDVALID** must be LOW for the first cycle after **ARESETn** goes HIGH	C2.3.3

4.7 Barriers

Table 4-7 shows the barriers channel protocol rules.

Table 4-7 Barriers

Assertion	Description	AMBA AXI and ACE Protocol Specification reference
ACE_ERRM_R_W_BARRIER_CTL	The ID, DOMAIN BAR, and PROT values of corresponding read and write barrier transactions must be the same. Barrier transactions must be issued in the same order on the read and write address channels.	C8.4.1
ACE_ERRM_AR_BARRIER_ID	The ID of a read barrier must not be the same as any outstanding non-barrier transaction.	C8.4.1
ACE_ERRM_AW_BARRIER_ID	The ID of a write barrier must not be the same as any outstanding non-barrier transaction.	C8.4.1
ACE_ERRM_AR_BARRIER_CTL	Barrier transactions must have the following payload values: **AxADDR** = All zeros **AxLEN** = `8'b0` **AxBURST** = `2'b01` **AxSIZE** = Data bus width **AxSNOOP** = All zeros **AxCACHE** = `'b0010` **AxLOCK** = `'b0`	Table C3-13
ACE_ERRM_AW_BARRIER_CTL	Barrier transactions must have the following payload values: **AxADDR** = All zeros **AxLEN** = All zeros **AxBURST** = `'b01` **AxSIZE** = Data bus width **AxSNOOP** = All zeros **AxSIZE** = Data bus width **AxCACHE** = `'b0010` **AxLOCK** = `'b0`	Table C3-13
ACE_ERRM_R_BARRIER_NUM	A master interface must not issue more than 256 outstanding barrier transactions.	C8.4.1
ACE_ERRM_W_BARRIER_NUM	A master interface must not issue more than 256 outstanding barrier transactions.	C8.4.1
ACE_ERRS_R_BARRIER_LAST	A read barrier response must always have **RLAST** asserted.	Table C8-2

4.8 Configuration

Table 4-8 shows the configuration rules.

Table 4-8 Configuration

Assertion	Description	AMBA AXI and ACE Protocol Specification reference
ACE_AUX_CACHE_LINE_SIZE	Cache line size must be defined as 16, 32, 64, 128, 256, 1024, or 2048 bytes.	-
ACE_AUX_CACHE_DATA_WIDTH32	For a data width or snoop data width of 32, the minimum cache line size is 16 bytes and the maximum is 64 bytes.	C3.1.4
ACE_AUX_CACHE_DATA_WIDTH64	For a data width or snoop data width of 64, the minimum cache line size is 16 bytes and the maximum is 128 bytes.	C3.1.4
ACE_AUX_CACHE_DATA_WIDTH128	For a data width or snoop data width of 128, the minimum cache line size is 16 bytes and the maximum is 256 bytes.	C3.1.4
ACE_AUX_CACHE_DATA_WIDTH256	For a data width or snoop data width of 256, the minimum cache line size is 32 bytes and the maximum is 512 bytes.	C3.1.4
ACE_AUX_CACHE_DATA_WIDTH512	For a data width or snoop data width of 512, the minimum cache line size is 64 bytes and the maximum is 1024 bytes.	C3.1.4
ACE_AUX_CACHE_DATA_WIDTH1024	For a data width or snoop data width of 1024, the minimum cache line size is 128 bytes and the maximum is 2048 bytes.	C3.1.4
ACE_AUX_MAXCBURSTS	The MAXCBURSTS parameter must be greater than or equal to 1.	-
ACE_AUX_ARCAM_OVERFLOW	Read CAM overflow, increase the MAXRBURSTS parameter.	-
ACE_AUX_AWCAM_OVERFLOW	Write CAM overflow, increase the MAXWBURSTS parameter.	-
ACE_AUX_ACCAM_OVERFLOW	Snoop CAM overflow, increase the MAXCBURSTS parameter.	-
ACE_AUX_ARCAM_UNDERFLOW	Read CAM underflow.	-
ACE_AUX_AWCAM_UNDERFLOW	Write CAM underflow.	-
ACE_AUX_ACCAM_UNDERFLOW	Snoop CAM underflow.	-
ACE_AUX_CD_DATA_WIDTH	The CD_DATA_WIDTH parameter must be 32, 64, 128, 256, 512, or 1024 bits.	-
ACELITE_AUX_MAX_BARRIERS	The MAX_BARRIERS parameter for the ACE-Lite protocol checker must be greater than or equal to 1.	-

4.9 End of simulation

Table 4-9 shows the end of simulation protocol rules.

Table 4-9 End of simulation

Assertion	Description	AMBA AXI and ACE Protocol Specification reference
ACE_ERRM_AC_EOS	At the end of simulation, all snoop transactions have completed	-
ACE_ERRM_R_W_BARRIER_EOS	A master must issue barrier transactions on both the read and write channels	C8.4.1
ACE_ERRM_RACK_EOS	A master must complete all read transactions using the **RACK** signal	-
ACE_ERRM_WACK_EOS	A master must complete all write transactions using the **WACK** signal	-
ACE_ERR_W_EOS	At the end of simulation, all write transactions have completed	-

4.10 ACE AC channel

Table 4-10 shows the ACE AC channel protocol rules.

Table 4-10 ACE AC channel

Assertion	Description	AMBA AXI and ACE Protocol Specification reference
ACE_ERRS_ACVALID_STABLE	When **ACVALID** is asserted, it must remain asserted until **ACREADY** is asserted	C2.3.3
ACE_ERRS_ACADDR_STABLE	AC payload signals must remain stable while **ACVALID** is asserted, and **ACREADY** is de-asserted	C3.6.2
ACE_ERRS_ACSNOOP_STABLE	**ACSNOOP** must be stable when **ACVALID** is asserted	C3.6.2
ACE_ERRS_ACPROT_STABLE	**ACPROT** must be stable when **ACVALID** is asserted	C3.6.2
ACE_ERRS_ACADDR_X	A value of X on **ACADDR** is not permitted when **ACVALID** is HIGH	A3.2.2
ACE_ERRS_ACSNOOP_X	A value of X on **ACSNOOP** is not permitted when **ACVALID** is HIGH	A3.2.2
ACE_ERRS_ACPROT_X	A value of X on **ACPROT** is not permitted when **ACVALID** is HIGH	A3.2.2
ACE_ERRS_ACVALID_X	When not in reset, a value of X on **ACVALID** is not permitted	A3.2.1
ACE_ERRM_ACREADY_X	When not in reset, a value of X on **ACREADY** is not permitted	A3.2.1
ACE_ERRS_ACSNOOP	When **ACVALID** is asserted, **ACSNOOP** must be one of the legal values	Table C3-19
ACE_ERRS_AC_ALIGN	When **ACVALID** is asserted, **ACADDR** must be data width-aligned	C3.6.2
ACE_RECM_ACREADY_MAX_WAIT	**ACREADY** should be asserted within MAXWAITS cycles of **ACVALID** being asserted	-

4.11 ACE CR channel

Table 4-11 shows the ACE CR channel protocol rules.

Table 4-11 ACE CR channel

Assertion	Description	AMBA AXI and ACE Protocol Specification reference
ACE_ERRM_CRVALID_X	When not in reset, a value of X on **CRVALID** is not permitted	A3.2.2
ACE_ERRS_CRREADY_X	When not in reset, a value of X on **CRREADY** is not permitted	A3.2.2
ACE_ERRM_CRVALID_STABLE	When **CRVALID** is asserted, it must remain asserted until **CRREADY** is asserted	C3.7
ACE_ERRM_CRRESP_STABLE	CR payload signals must remain stable while **CRVALID** is asserted and **CRREADY** is de-asserted	C3.7
ACE_ERRM_CRRESP_DIRTY	When **CRVALID** is asserted and **CRRESP[2]**, PassDirty, is asserted, then the **CRRESP[0]**, data transfer, signal must be asserted	Table C3-22
ACE_ERRM_CRRESP_SHARED	When **CRVALID** is asserted in response to an AC transfer, where **ACSNOOP** was ReadUnique, CleanInvalid, or MakeInvalid, the **CRRESP[3]**, IsShared, signal must be de-asserted	C3.7
ACE_ERRM_CR_ORDER	**CRVALID** must not be asserted before the corresponding AC handshake	Figure C3-1
ACE_ERRM_CRRESP_X	When **CRVALID** is asserted, a value of X on **CRRESP** is not permitted	A3.2.2
ACE_ERRM_CRRESP_DVM	A DVM response must use **CRRESP[4:0]** = 5'b000x0	C12.3.4
ACE_ERRM_CRRESP_DVM_ERROR	An error response is not permitted for a: • DVM Sync • DVM Complete • DVM Hint.	C12.3.4, C12.7.6
ACE_RECS_CRREADY_MAX_WAIT	**CRREADY** should be asserted within MAXWAITS cycles of **CRVALID** being asserted	-

4.12 ACE CD channel

Table 4-12 shows the ACE CD channel protocol rules.

Table 4-12 ACE CD channel

Assertion	Description	AMBA AXI and ACE Protocol Specification reference
ACE_ERRM_CDVALID_X	When not in reset, a value of X on **CDVALID** is not permitted.	A3.2.2
ACE_ERRM_CDVALID_STABLE	When **CDVALID** is asserted, it must remain asserted until **CDREADY** is asserted.	C3.8
ACE_ERRS_CDREADY_X	When not in reset, a value of X on **CDREADY** is not permitted.	A3.2.2
ACE_ERRM_CDDATA_STABLE	CD payload signals must remain stable while **CDVALID** is asserted and **CDREADY** is de-asserted.	C3.8
ACE_ERRM_CDLAST_STABLE	CD payload signals must remain stable while **CDVALID** is asserted and **CDREADY** is de-asserted.	C3.8
ACE_ERRM_CDDATA_NUM	For each snoop that had **CRRESP[2]**, data transfer, asserted, exactly the number of CD transfers indicated by the cache line size must exist. When **CDVALID** is asserted, **CDLAST** must be asserted on and only on the last CD transfer in a transaction.	C3.8, C3.7
ACE_ERRM_CD_ORDER	**CDVALID** must not be asserted before the corresponding AC handshake.	Figure C3-1
ACE_ERRM_CDLAST_X	When **CDVALID** is asserted, a value of X on **CDLAST** is not permitted.	A3.2.2
ACE_ERRM_CDDATA_X	When **CDVALID** is asserted, a value of X on **CDDATA** is not permitted.	A3.2.2
ACE_RECS_CDREADY_MAX_WAIT	**CDREADY** should be asserted within MAXWAITS cycles of **CDVALID** being asserted.	-

4.13 ACE intertransaction

Table 4-13 shows the ACE intertransaction rules.

Table 4-13 ACE intertransaction rules

Assertion	Description	AMBA AXI and ACE Protocol Specification reference
ACE_ERRM_CRRESP_IN_WB_WC	If a WriteBack or WriteClean is in progress for a cacheline, from **AWVALID** being asserted until **BVALID** && **BREADY**, then any snoops to the same or overlapping addresses must respond with IsShared = 1 and PassDirty = 0 on the **CRRESP** pins.	C5.2.3
ACE_ERRM_AR_IN_CMAINT	A cacheable read transaction must not be issued on an interface that has an outstanding, overlapping cache maintenance transaction.	C7.2.1
ACE_ERRM_AW_IN_CMAINT	A cacheable write transaction must be issued on an interface that has an outstanding, overlapping cache maintenance transaction.	C7.2.1
ACE_ERRM_CMAINT_IN_READ	Cache maintenance operations cannot be performed if the master has outstanding cacheable read transactions to the same cache line.	C7.2.1
ACE_ERRM_CMAINT_IN_WRITE	Cache maintenance operations cannot be performed if the master has outstanding cacheable write transactions to the same cache line.	C7.2.1
ACE_ERRS_BRESP_IN_SNOOP	If the interconnect sends a snoop transaction to a master, it must not provide the same master with a response to a WriteUnique or WriteLineUnique transaction to the same cache line until it has received a snoop response on **CRRESP** to the snoop transaction.	C6.2
ACE_ERRS_AC_IN_BRESP	If the interconnect provides a master with a response to a WriteUnique or WriteLineUnique transaction, it must not send the same master a snoop transaction to the same cache line until it has received an acknowledgement of the transaction response on **WACK**.	C6.2
ACE_ERRS_RRESP_IN_SNOOP	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.	C6.2
ACE_ERRS_AC_IN_RRESP	If the interconnect provides a master with a response to a transaction, it must not send the same master a snoop transaction to the same cache line until it has received an acknowledgement of the transaction response on **RACK**.	C6.2
ACE_RECM_R_W_HAZARD	A master should not issue a read transaction to a memory region to which it is writing.	-
ACE_RECM_W_R_HAZARD	A master should not issue a write transaction to a memory region to which it is reading.	-
ACE_REC_SW_RRESP_IN_SNOOP	At any point in time, different agents should have a consistent view of the shareability of a memory region. A read response to a non-shareable transaction has been sent to the same cache line as an active snoop.	A4.5
ACE_REC_SW_AC_IN_RRESP	At any point in time, different agents should have a consistent view of the shareability of a memory region. A snoop has been received to a region that has an outstanding non-shareable read.	A4.5

Table 4-13 ACE intertransaction rules (continued)

Assertion	Description	AMBA AXI and ACE Protocol Specification reference
ACE_REC_SW_BRESP_IN_SNOOP	At any point in time, different agents should have a consistent view of the shareability of a memory region. A write response to a non-shareable transaction has been sent to the same cache line as an active snoop.	A4.5
ACE_REC_SW_AC_IN_BRESP	At any point in time, different agents should have a consistent view of the shareability of a memory region. A snoop has been received to a region that has an outstanding non-shareable write.	A4.5
ACE_ERRM_XSTORE_IN_XSEQ	A single exclusive-capable thread must not have an exclusive store transaction in progress at the same time as any transaction that registers that a master is performing an exclusive sequence.	C9.6

4.14 DVM

Table 4-14 shows the DVM rules.

Table 4-14 DVM rules

Assertion	Description	AMBA AXI and ACE Protocol Specification reference
ACE_ERRS_DVM_COMPLETE	DVM complete message received on the AC channel, but there is no outstanding DVM message that requires a completion message.	C12.1
ACE_ERRM_DVM_COMPLETE	DVM complete message sent on the AR channel, but there is no outstanding DVM message that requires a completion message.	C12.1
ACE_ERRM_DVM_TYPES	The DVM message type indicated by **ARADDR[14:12]** is not supported.	Table C12-2
ACE_ERRS_DVM_TYPES	The DVM message type indicated by **ACADDR[14:12]** is not supported.	Table C12-2
ACE_ERRM_DVM_RESVD_1	**ARADDR[4:1]** and **ARADDR[7]** are reserved and should be zero for DVM other than DVM Hint.	Table C12-2
ACE_ERRM_DVM_RESVD_2	The **ARADDR[3:0]** are reserved and should be zero when conveying additional DVM address information.	Table C12-3
ACE_ERRM_DVM_RESVD_3	The **ARADDR[15]** should be HIGH for DVM sync messages and **ARADDR[11:0]** should be LOW.	Table C12-13
ACE_ERRM_DVM_RESVD_4	The **ARADDR[15]** should be LOW for DVM messages other than sync messages.	Table C12-2
ACE_ERRS_DVM_RESVD_1	The **ACADDR[4:1]** and **ACADDR[7]** are reserved and should be zero for DVM other than DVM Hint.	Table C12-2
ACE_ERRS_DVM_RESVD_2	The **ACADDR[3:0]** are reserved and should be zero when conveying DVM address information.	Table C12-3
ACE_ERRS_DVM_RESVD_3	The **ACADDR[15]** should be HIGH for DVM sync messages and **ACADDR[11:0]** should be LOW.	Table C12-13
ACE_ERRS_DVM_RESVD_4	The **ACADDR[15]** should be LOW for DVM messages other than sync messages.	Table C12-2
ACE_ERRM_DVM_TLB_INV	The TLB Invalidate message indicated by **ARADDR** is not supported.	Table C12-6
ACE_ERRS_DVM_TLB_INV	The TLB Invalidate message indicated by **ACADDR** is not supported.	Table C12-6
ACE_ERRM_DVM_BP_INV	The Branch Predictor Invalidate message indicated by **ARADDR** is not supported.	Table C12-8
ACE_ERRS_DVM_BP_INV	The Branch Predictor Invalidate message indicated by **ACADDR** is not supported.	Table C12-8
ACE_ERRM_DVM_PHY_INV	The Physical Instruction Cache Invalidate message indicated by **ARADDR** is not supported.	Table C12-10
ACE_ERRS_DVM_PHY_INV	The Physical Instruction Cache Invalidate message indicated by **ACADDR** is not supported.	Table C12-10
ACE_ERRM_DVM_VIR_INV	The Virtual Instruction Cache Invalidate message indicated by **ARADDR** is not supported.	Table C12-12

Table 4-14 DVM rules (continued)

Assertion	Description	AMBA AXI and ACE Protocol Specification reference
ACE_ERRS_DVM_VIR_INV	The Virtual Instruction Cache Invalidate message indicated by **ACADDR** is not supported.	Table C12-12
ACE_ERRM_DVM_CTL	DVM must have the following payload values: **ARLEN** = All zeros **ARBURST** = 'b01 **ARSIZE** = data bus width **ARCACHE** = 'b0010 **ARLOCK** = 'b0 **ARSNOOP[3:1]** = 'b111 **ARDOMAIN** = inner shareable or outer shareable **ARBAR** = 'b00	Table C12-4
ACE_ERRM_DVM_COMPLETE_CTL	DVM Complete transactions must have **ARADDR** = all zeros.	Table C12-4
ACE_ERRS_DVM_COMPLETE_CTL	DVM Complete transactions must have **ACADDR** = all zeros.	Table C12-4
ACE_ERRM_DVM_ID	A DVM message must not share IDs with non-DVM read transactions or barriers.	C12.3.5
ACE_ERRM_DVM_MULTIPART_ID	Both parts of a multipart DVM transaction must have the same ID.	C12.3.3
ACE_ERRM_DVM_SYNC	A component can have only one outstanding message requiring a completion message.	C12.1
ACE_ERRS_DVM_MULTIPART_RRESP	It is required that the response given to all parts of a multiple transaction message are the same.	C12.3.4
ACE_ERRM_DVM_MULTIPART_CRRESP	It is required that the response given to all parts of a multiple transaction message are the same.	C12.3.4
ACE_ERRM_DVM_MULTIPART_SUCCESSIVE	Multi-part DVM messages are always sent as successive transactions, and no other transaction can be interposed between them.	C12.3.3
ACE_ERRS_DVM_MULTIPART_SUCCESSIVE	Multi-part DVM messages are always sent as successive transactions, and no other transaction can be interposed between them.	C12.3.3
ACE_ERRS_DVM_LAST	A DVM response must always have **RLAST** asserted.	-

4.15 ACE-Lite

Table 4-15 shows the ACE-Lite rules.

Table 4-15 ACE-Lite rules

Assertion	Description	AMBA AXI and ACE Protocol Specification reference
ACELITE_ERRM_AWSNOOP	When **AWVALID** is HIGH, **AWSNOOP** must be one of the non-reserved values.	Table C11-2
ACELITE_ERRM_ARSNOOP	When **ARVALID** is HIGH, **ARSNOOP** must be one of the non-reserved values.	Table C11-1

Appendix A
Example Usage

This appendix provides an example transcript from the protocol checker. It contains the following section:

- *RDATA stable failure* on page A-2.

A.1 RDATA stable failure

Figure A-1 shows the timing diagram for a failure of the AXI_ERRS_RDATA_STABLE check.

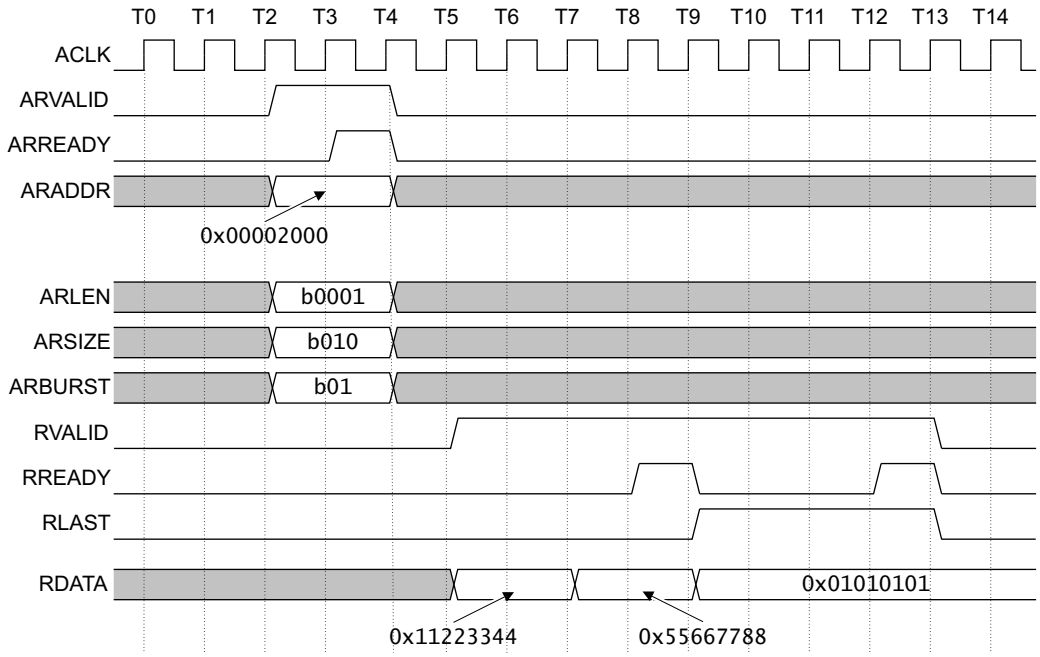


Figure A-1 RDATA stable failure

RDATA changes at T7 when **RVALID** is HIGH and **RREADY** is LOW. The protocol checker samples the change at T8.

Example A-1 shows the protocol checker transcript for this failure.

Example A-1 RDATA stable failure

```
# Loading sv_std.std
# Loading work.avip_testbench
# Loading work.Axi4PC
# Loading work.BaseClk
# do startup.do
# AXI4_INFO: Running Axi4PC_ace $State
# ** Error: AXI4_ERRS_RDATA_STABLE. RDATA must remain stable when RVALID is asserted and RREADY LOW.
     Spec: section 3.1, and figure 3-1 on page 3-2.
# Time: 1050 ns Started: 950 ns Scope: avip_testbench.uAxi4PC.axi4_errs_rdata_stable File: ../Axi4PC.sv
     Line: 2595 Expr: $stable(RDATA|~RdataMask)
# ** Note: $finish    : stim.svh(84)
# Time: 3960 ns  Iteration: 1  Instance: /avip_testbench
```

Appendix B
Revisions

This appendix describes the technical changes between released issues of this book.

Table B-1 Issue A

Change	Location	Affects
No changes, first release	-	-