SYNOPSYS®

DesignWare® DW_axi_rs

Databook

DW_axi_rs – Product Code

2.04a
March 2020

Copyright Notice and Proprietary Information

Destination Control Statement

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

Disclaimer

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

Trademarks

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

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

Free and Open-Source Software Licensing Notices

If applicable, Free and Open-Source Software (FOSS) licensing notices are available in the product installation.

Third-Party Links

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

Synopsys, Inc.
www.synopsys.com

Contents

Revision History

This section tracks the significant documentation changes that occur from release-to-release and during a release from version 1.00c onward.

Date	Release	Description
2.04a	March 2020	■ Revision version change for 2020.03a release ■ Updated synthesis results in "Performance" on page 73 ■ Updated "Verification" on page 69 ■ "Signal Descriptions" on page 35, "Parameter Descriptions" on page 27, "Internal Parameter Descriptions" on page 67 auto-extracted from the RTL
2.03a	February 2018	■ Revision version change for 2018.02a release ■ Updated synthesis results in "Performance" on page 73 ■ Removed Chapter 2 Building and Verifying a Component or Subsystem from the databook and added the contents in the newly created user guide. ■ "Signal Descriptions" on page 35, "Parameter Descriptions" on page 27, "Internal Parameter Descriptions" on page 67 auto-extracted from the RTL with change bars
2.02a	March 2016	■ Revision version change for 2016.03a release ■ Added "Running SpyGlass® Lint and SpyGlass® CDC" section ■ Added "Running Spyglass on Generated Code with coreAssembler" section ■ "Signal Descriptions" on page 35 and "Parameter Descriptions" on page 27 auto-extracted from the RTL ■ Added chapter "Internal Parameter Descriptions" on page 67 ■ Updated area and power numbers in "Performance" on page 73
2.01a	October 2014	■ Version change for 2014.10a release. ■ Added "Integration Considerations" chapter.
2.00a	June 2013	Added information about new and modified signals and parameters to support the AMBA 4 AXI and ACE-Lite specification.
1.02c	May 2013	Version change for 2013.05a release.
1.02b	Oct 2012	Added the product code on the cover and in Table 1-1.
1.02b	Oct 2011	Version change for 2011.10a release.

Date	Release	Description
1.02a	Jun 2011	Updated system diagram in Figure 1-1; enhanced “Related Documents” section in Preface.
1.02a	Sep 2010	Corrected names of include files and vcs command used for simulation.
1.01b	Dec 2009	Updated databook to new template for consistency with other IIP/VIP/PHY databooks.
1.01b	May 2009	Removed references to Quickstarts, as they are no longer supported.
1.01b	Mar 2009	Corrected Figure 16 with a register in combinatorial path for payload multiplexer
1.01a	Oct 2008	Version change for 2008.10a release.
1.00d	Jun 2008	Version change for 2008.06a release.
1.00c	Jun 2007	Version change for 2007.06a release.

Preface

This databook provides information about the DesignWare AXI register slice (DW_axi_rs). This component conforms to the AMBA 3 AXI and AMBA 4 AXI specifications defined in the *AMBA AXI and ACE Protocol Specification* from ARM.

The information in this databook includes a functional description, signal and parameter descriptions, as well as information on how you can configure, create RTL for, simulate, and synthesize the component using Synopsys coreAssembler or coreConsultant. This manual also provides an overview of the component testbench, a description of the tests that are run to verify the coreKit, and synthesis information for the coreKit.

Organization

The chapters of this databook are organized as follows:

- Chapter 1, "Product Overview" provides a system overview, a component block diagram, basic features, and an overview of the verification environment.
- Chapter 2, "Functional Description" describes the functional operation of the DW_axi_rs.
- Chapter 3, "Parameter Descriptions" identifies the configurable parameters supported by the DW_axi_rs.
- Chapter 4, "Signal Descriptions" provides a list and description of the DW_axi_rs signals.
- Chapter 5, "Internal Parameter Descriptions" provides a list of internal parameter descriptions that might be indirectly referenced in expressions in the Signals and Parameters chapters.
- Chapter 6, "Verification" provides information on verifying the configured DW_axi_rs.
- Chapter 7, "Integration Considerations" includes information you need to integrate the configured DW_axi_rs into your design.
- Appendix A, "Glossary" provides a glossary of general terms.

Related Documentation

- *Using DesignWare Library IP in coreAssembler* – Contains information on getting started with using DesignWare SIP components for AMBA 2 and AMBA 3 AXI/AMBA 4 AXI components within coreTools
- *coreAssembler User Guide* – Contains information on using coreAssembler
- *coreConsultant User Guide* – Contains information on using coreConsultant

To see a complete listing of documentation within the DesignWare Synthesizable Components for AMBA 3 AXI and AMBA 4 AXI, see the *Guide to Documentation for DesignWare Synthesizable Components for AMBA 2, AMBA 3 AXI, and AMBA 4 AXI*.

Web Resources

- DesignWare IP product information: https://www.synopsys.com/designware-ip.html
- Your custom DesignWare IP page: https://www.synopsys.com/dw/mydesignware.php
- Documentation through SolvNetPlus: https://solvnetplus.synopsys.com (Synopsys password required)
- Synopsys Common Licensing (SCL): https://www.synopsys.com/keys

Customer Support

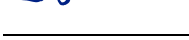
Synopsys provides the following various methods for contacting Customer Support:

- Prepare the following debug information, if applicable:
  - For environment set-up problems or failures with configuration, simulation, or synthesis that occur within coreConsultant or coreAssembler, select the following menu:

    File > Build Debug Tar-file

    Check all the boxes in the dialog box that apply to your issue. This option gathers all the Synopsys product data needed to begin debugging an issue and writes it to the <*core tool startup directory*>/debug.tar.gz file.
  - For simulation issues outside of coreConsultant or coreAssembler:
    - Create a waveforms file (such as VPD or VCD).
    - Identify the hierarchy path to the DesignWare instance.
    - Identify the timestamp of any signals or locations in the waveforms that are not understood.
- *For the fastest response*, enter a case through SolvNetPlus:

  a. https://solvnetplus.synopsys.com

  
  Note SolvNetPlus does not support Internet Explorer. Use a supported browser such as Microsoft Edge, Google Chrome, Mozilla Firefox, or Apple Safari.

  b. Click the **Cases** menu and then click **Create a New Case** (below the list of cases).

  c. Complete the mandatory fields that are marked with an asterisk and click **Save**.

  Ensure to include the following:
  - **Product L1:** DesignWare Library IP
  - **Product L2:** <name of L2>

  d. After creating the case, attach any debug files you created.

  For more information about general usage information, refer to the following article in SolvNetPlus:

  https://solvnetplus.synopsys.com/s/article/SolvNetPlus-Usage-Help-Resources

- Or, send an e-mail message to support_center@synopsys.com (your email will be queued and then, on a first-come, first-served basis, manually routed to the correct support engineer):
  - Include the Product L1 and Product L2 names, and Version number in your e-mail so it can be routed correctly.
  - For simulation issues, include the timestamp of any signals or locations in waveforms that are not understood
  - Attach any debug files you created.
- Or, telephone your local support center:
  - North America:
    Call 1-800-245-8005 from 7 AM to 5:30 PM Pacific time, Monday through Friday.
  - All other countries:
    https://www.synopsys.com/support/global-support-centers.html

Product Code

Table 1-1 lists all the components associated with the product code for DesignWare AMBA Fabric.

Table 1-1 DesignWare AMBA Fabric – Product Code: 3768-0

Component Name	Description
DW_ahb	High performance, low latency interconnect fabric for AMBA 2 AHB
DW_ahb_eh2h	High performance, high bandwidth AMBA 2 AHB to AHB bridge
DW_ahb_h2h	Area efficient, low bandwidth AMBA 2 AHB to AHB Bridge
DW_ahb_icm	Configurable multi-layer interconnection matrix
DW_ahb_ictl	Configurable vectored interrupt controllers for AHB bus systems
DW_apb	High performance, low latency interconnect fabric & bridge for AMBA 2 APB for direct connect to AMBA 2 AHB fabric
DW_apb_ictl	Configurable vectored interrupt controllers for APB bus systems
DW_axi	High performance, low latency interconnect fabric for AMBA 3 AXI and AMBA 4 AXI
DW_axi_a2x	Configurable bridge between AMBA 3 AXI/AMBA 4 AXI components and AHB components or between AMBA 3 AXI/AMBA 4 AXI components and AMBA 3 AXI/AMBA 4 AXI components.
DW_axi_gm	Simplify the connection of third party/custom master controllers to any AMBA 3 AXI or AMBA 4 AXI fabric
DW_axi_gs	Simplify the connection of third party/custom slave controllers to any AMBA 3 AXI or AMBA 4 AXI fabric
DW_axi_hmx	Configurable high performance interface from an AHB master to an AMBA 3 AXI or AMBA 4 AXI slave
DW_axi_rs	Configurable standalone pipelining stage for AMBA 3 AXI or AMBA 4 AXI subsystems

Table 1-1 DesignWare AMBA Fabric – Product Code: 3768-0 (Continued)

Component Name	Description
DW_axi_x2h	Bridge from AMBA 3 AXI or AMBA 4 AXI to AMBA 2.0 AHB, enabling easy integration of legacy AHB designs with newer AXI systems
DW_axi_x2p	High performance, low latency interconnect fabric and bridge for AMBA 2 & 3 APB for direct connect to AMBA 3 AXI or AMBA 4 AXI fabric
DW_axi_x2x	Flexible bridge between multiple AMBA 3 AXI components or busses

1 Product Overview

The DW_axi_rs component is an AMBA 3 AXI/AMBA 4 AXI/ACE-Lite register slice (rs) component. DW_axi_rs is designed to perform pipelining of the AXI channels without adversely affecting AXI throughput. Four different pipelining options are available and can be selected on a per AXI channel basis. Each side of DW_axi_rs conforms to the AXI protocol. DW_axi_rs propagates AXI transfers initiated by external AXI masters (for the read/write address and write data channels) and by external AXI slaves (for the read data and write response channels), according to the pipelined timing options set for that channel.

Note You must have a DWC-AMBA-Fabric-Source license to enable the AXI4 or ACE-Lite interface.

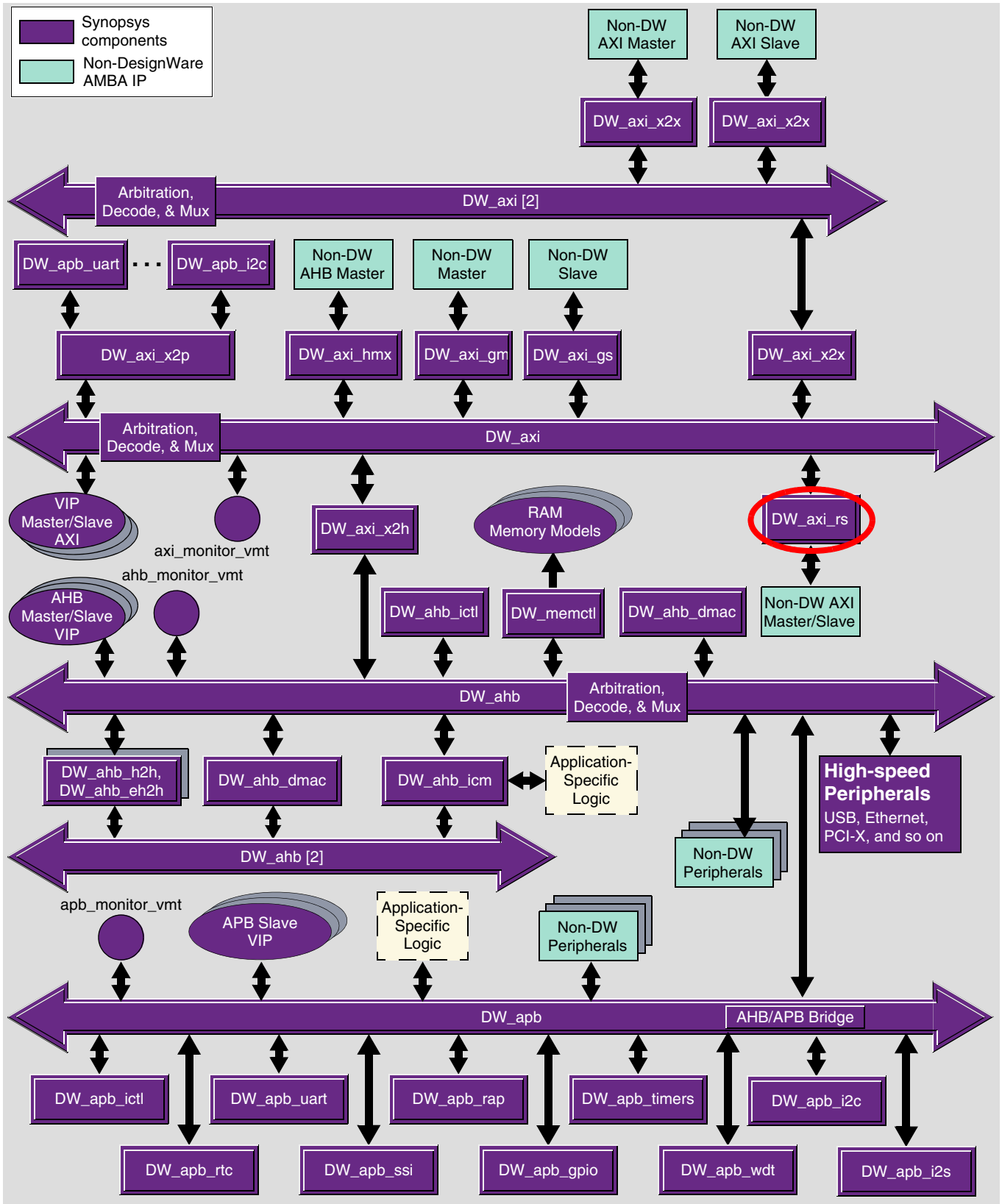
1.1 DesignWare Synthesizable Components for AMBA System Overview

The Synopsys DesignWare Synthesizable Components environment is a parameterizable bus system containing AMBA version 2.0 compliant AHB (Advanced High-performance Bus) and APB (Advanced Peripheral Bus) components, AMBA 3 AXI/AMBA 4 AXI (Advanced eXtensible Interface) compliant components., and ACE-Lite (AXI Coherence Extension) compliant components.

Figure 1-1 illustrates one example of this environment, including the AXI bus, the AHB bus, and the APB bus. Included in this subsystem are synthesizable IP for AXI/AHB/APB peripherals, bus bridges, and an AXI interconnect and AHB bus fabric. Also included are verification IP for AXI/AHB/APB master/slave models and bus monitors. In order to display the databook for a DW_* component, click on the corresponding component object in the illustration.

Attention Links resolve only if you are viewing this databook from your $DESIGNWARE_HOME tree, and to only those components that are installed in the tree.

Figure 1-1 Example of DW_axi_rs in a Complete System



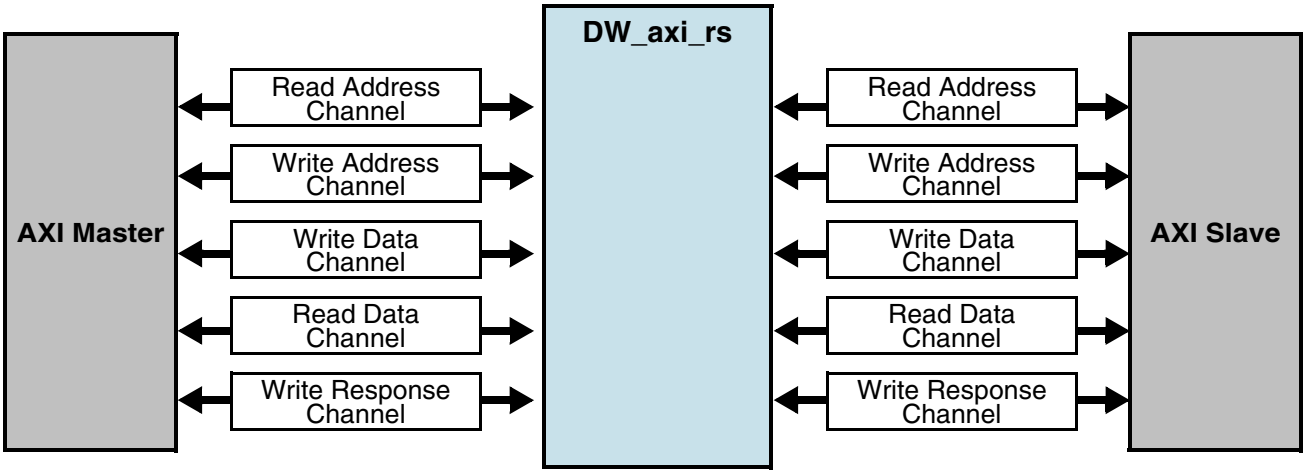
1.2 General Product Description

DW_axi_rs is used to isolate long timing paths. When timing analysis shows the critical path in a subsystem is between an AXI master and slave, DW_axi_rs can be inserted. DW_axi_rs can then be configured to insert a pipeline stage on one or more AXI channels to break the long timing paths. The inserted pipeline stage adds a cycle of latency to the channel, but the pipeline stage's registers break long combinatorial timing paths between the master and the slave. By breaking critical timing paths, the overall subsystem frequency can be increased at the expense of adding latency in an isolated portion of the subsystem.

There are four DW_axi_rs timing mode options available for each of the five AXI channels. For each instantiation of DW_axi_rs, each channel is independently configured to use one of the four timing modes: pass through, forward registered, backward registered, or fully registered.

Figure 1-2 shows a simple application of DW_axi_rs.

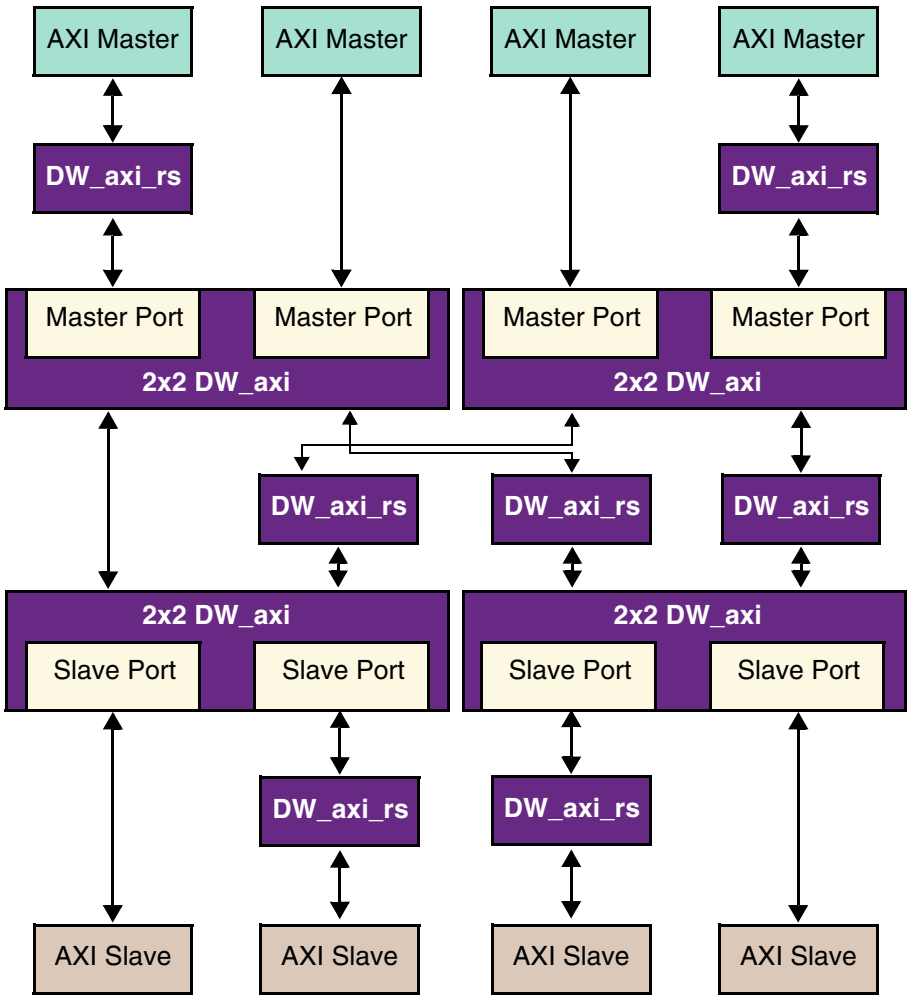
Figure 1-2 Example Usage of DW_axi_rs



DW_axi_rs can also be used between AXI interconnect ports and external masters and slaves. It is also possible to use DW_axi_rs between two connected AXI interconnects. Examples of both of these usages are illustrated in Figure 1-3.

Figure 1-3 shows a unique usage of DW_axi interconnect fabric, where a 4x4 interconnect is separated into four 2x2 interconnects. This is known as interconnect tiling. For more information on tiling, see the *DesignWare DW_axi Databook*.

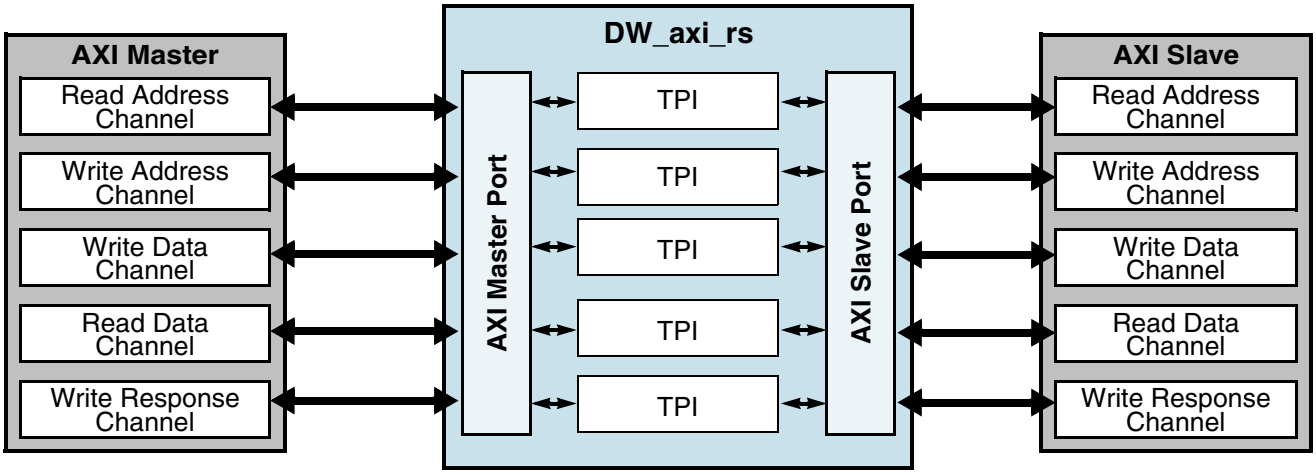
Figure 1-3 Example Usage of DW_axi_rs



1.2.1 DW_axi_rs Block Diagram

To implement the four timing modes for each channel, DW_axi_rs contains a sub-block called timing path isolation (TPI). This sub-block is instantiated for each channel, as shown in Figure 1-4. Each of the five instances of the timing path isolation block is statically configured with one of the four timing mode options. Each instance is configured independently of the other instances.

Figure 1-4 DW_axi_rs Block Diagram



1.3 Features

The DW_axi_rs supports the following features:

1.3.1 General

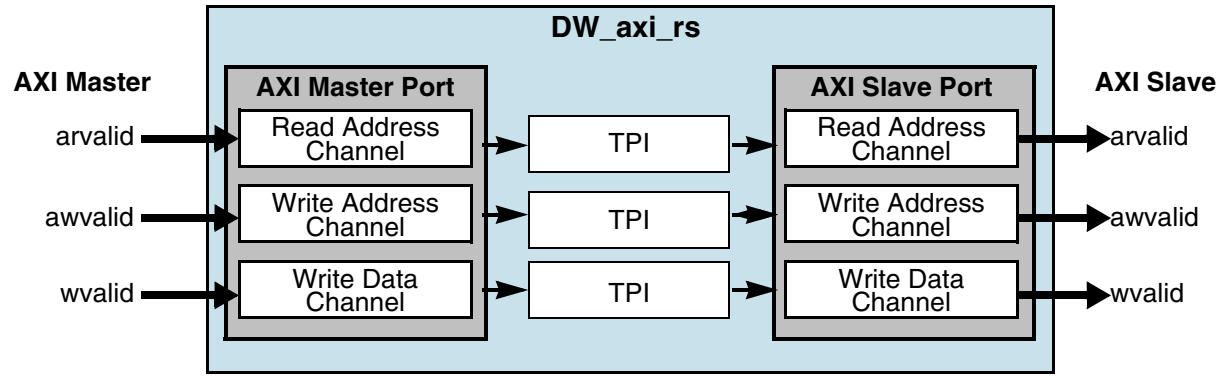
- Complies with the following specifications:
  - AMBA 3 AXI
  - AMBA 4 AXI
  - ACE-Lite
- Supports the following timing modes:
  - Pass through
  - Forward registered
  - Fully registered
  - Backward registered
- Configurable timing mode option on a per channel basis. The timing mode option selected for a channel is independent of the option selected for the other AXI channels.
- No throughput penalty for any of the timing mode options
- Sideband signaling support
- Master and slave have same address width, data width, ID width, and burst width
- Same clock domain for attached master and slave
- Adds one cycle of latency on registered paths

1.4 DW_axi_rs Terminology

The following terms are used throughout this databook.

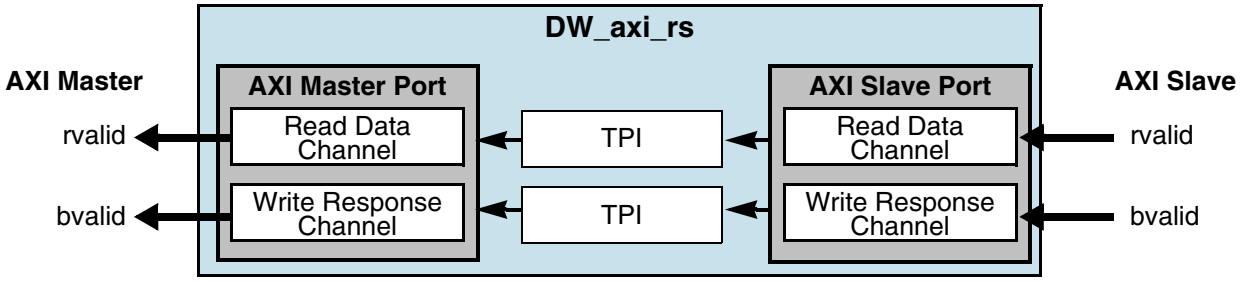
- **Forward control path** – For the read/write address channels and the write data channel, the forward control path is in the direction of arvalid/awvalid/wvalid from an AXI master to an AXI slave, as shown in Figure 1-5.

Figure 1-5 Forward Control Path – Read/Write Command and Write Data Channels



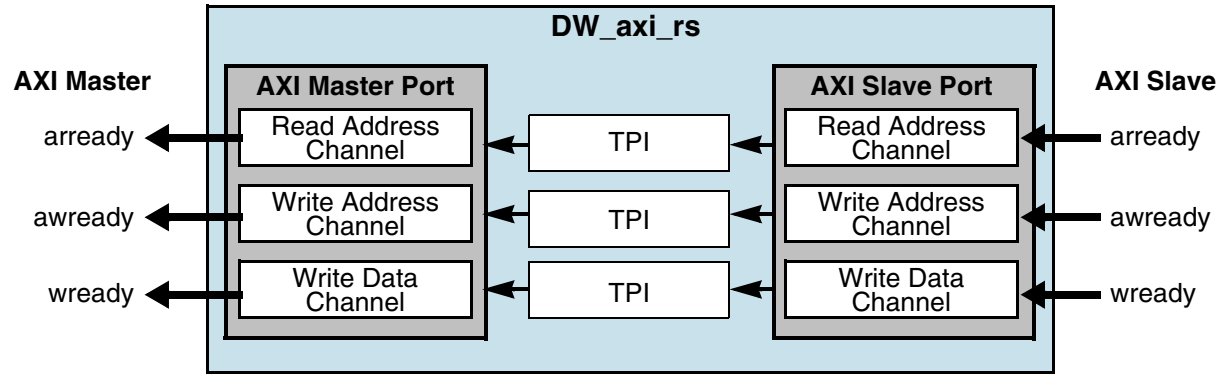
For the read data and write response channels, the forward control path is in the direction of rvalid/bvalid from an AXI slave to an AXI master, as shown in Figure 1-6.

Figure 1-6 Forward Control Path – Read Data and Write Response Channels



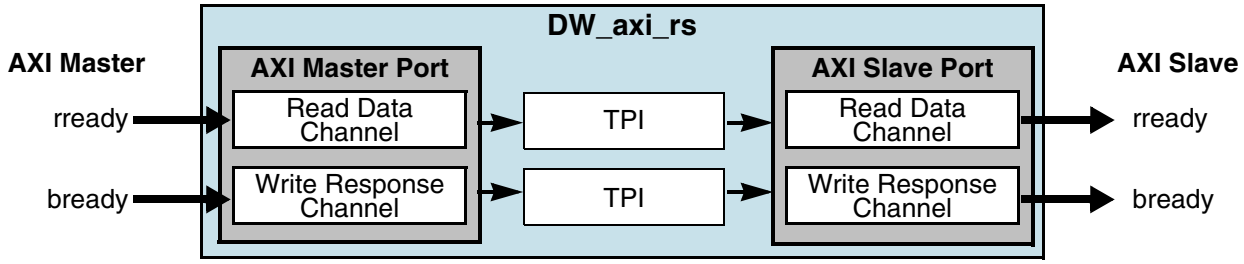
- **Backward control path** – For the read/write address channels and the write data channel, the backward control path is in the direction of aready/awready/wready from an AXI slave to an AXI master, as shown in Figure 1-7.

Figure 1-7 Backward Control Path – Read/Write Command and Write Data Channels



For the read data and write response channels, the backward control path is in the direction of rready/bready from an AXI master to an AXI slave, as shown in Figure 1-8.

Figure 1-8 Backward Control Path – Read Data and Write Response Channels



- **AXI payload –** The payload is the information being transferred from the payload source to the payload destination. The payload is valid when the *valid signal is asserted by the payload source. The payload is always in the direction of the forward control path. It is specific to the AXI channel. Table 1-1 provides the payload for each channel group.

Table 1-1 DW_axi_rs Payload[a]

Channel	Payload
Read/Write Address	Read/Write address and control signals
Read/Write Data	Read/Write data and control signals
Write Response	Write response and control signals

a. The control signals mentioned do not included the flow control signals *valid and *ready.

- **Payload source –** The payload source is the sender of the AXI channel payload.
- **Payload destination –** The payload destination is the receiver of the AXI channel payload.

  Table 1-2 provides the payload source and destination for each channel.

Table 1-2 DW_axi_rs Payload Source and Destination

Channel	Payload Source	Payload Destination
Read/Write Address	AXI Master	AXI Slave
Write Data	AXI Master	AXI Slave
Read Data	AXI Slave	AXI Master
Write Response	AXI Slave	AXI Master

- **Transfer –** A single channel sequence initiated by valid and ended by ready. A write address, read address, write data, read data, and write response phase are each, by themselves, a transfer.
- **Sideband Signals –** Optional control/status signals, not defined by the AXI protocol, on each channel. These are set by configuration parameters. An example use would be parity for data.

1.5 Standards Compliance

The DW_axi_rs component conforms to the *AMBA AXI and ACE Protocol Specification* from ARM. Readers are assumed to be familiar with this specification.

1.6 Verification Environment Overview

The DW_axi_rs includes an extensive verification environment, which sets up and invokes your selected simulation tool to execute tests that verify the functionality of the configured component. You can then analyze the results of the simulation.

The Verification chapter discusses the testbench environment and provides an overview of the tests that are used to verify DW_axi_rs when you run component-level simulation.

1.7 Licenses

Before you begin using the DW_axi_rs, you must have a valid license. For more information, see "Licenses" section in the DesignWare Synthesizable Components for AMBA 2, AMBA 3 AXI, and AMBA 4 AXI Installation Guide.

Source code for this component is available on a per-project basis as a DesignWare Core. Contact your local sales office for the details.

1.8 Where To Go From Here

At this point, you may want to get started working with the DW_axi_rs component within a subsystem or by itself. Synopsys provides several tools within its coreTools suite of products for the purposes of configuration, synthesis, and verification of single or multiple synthesizable IP components—coreConsultant and coreAssembler. For information on the different coreTools, see *Guide to coreTools Documentation*.

For more information about configuring, synthesizing, and verifying just your DW_axi_rs component, see *"Overview of the coreConsultant Configuration and Integration Process"* section in *DesignWare Synthesizable Components for AMBA 2 User Guide*.

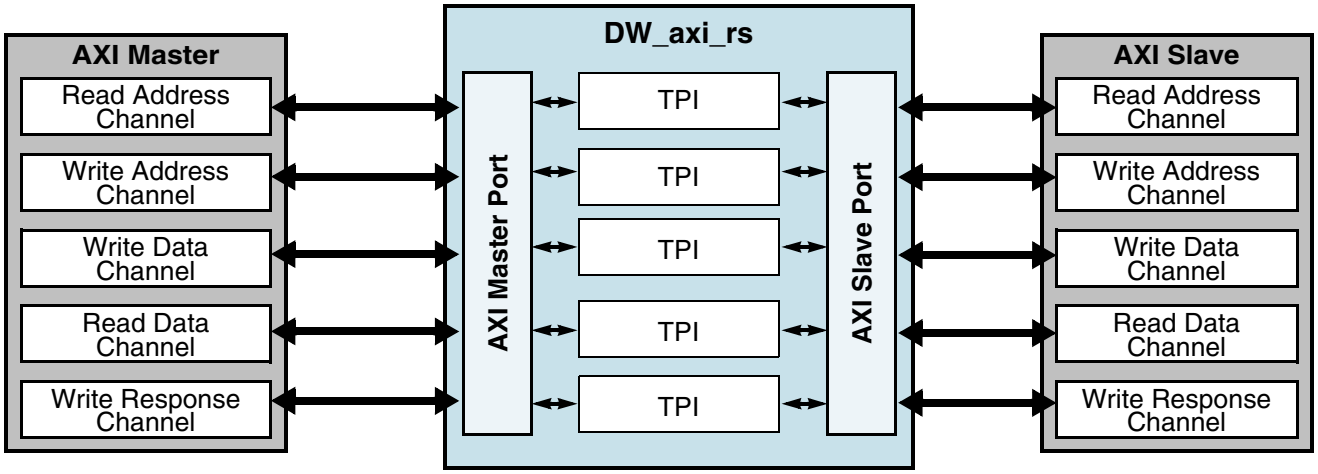
For more information about implementing your DW_axi_rs component within a DesignWare subsystem using coreAssembler, see *"Overview of the coreAssembler Configuration and Integration Process"* section in *DesignWare Synthesizable Components for AMBA 2 User Guide*.

2 Functional Description

The DW_axi_rs is used to isolate long timing paths. When timing analysis shows the critical path in a subsystem is between an AXI master and slave, the DW_axi_rs can be inserted. The DW_axi_rs can then be configured to insert a pipeline stage on one or more AXI channels to break the long timing paths. To achieve this, the component has a timing path isolation sub-block (TPI) that is instantiated for each AXI channel. Each instance of the TPI can be statically configured with one of the four timing mode options.

Figure 2-1 shows the block diagram for DW_axi_rs.

Figure 2-1 DW_axi_rs Block Diagram



This chapter provides detailed information about the timing mode options and how DW_axi_rs functions when configured for each mode. It also includes information about the sideband and register slice signals.

Note In the following timing mode descriptions and timing diagrams, the DW_axi_rs I/O signals are described generically using the terms ready/valid (to/from source) and ready/valid (from/to destination), rather than referring to a specific AXI channel's signal names. Since both masters and slaves can qualify as sources for transfers, depending on the channel, the generic terms source and destination are used to refer to the transfer instead. For more information, see Table 1-2.

It is highly recommended that you review "DW_axi_rs Terminology" on page 15 before reading this chapter, because the terms are used throughout the remainder of this databook.

2.1 Pass Through Timing Mode

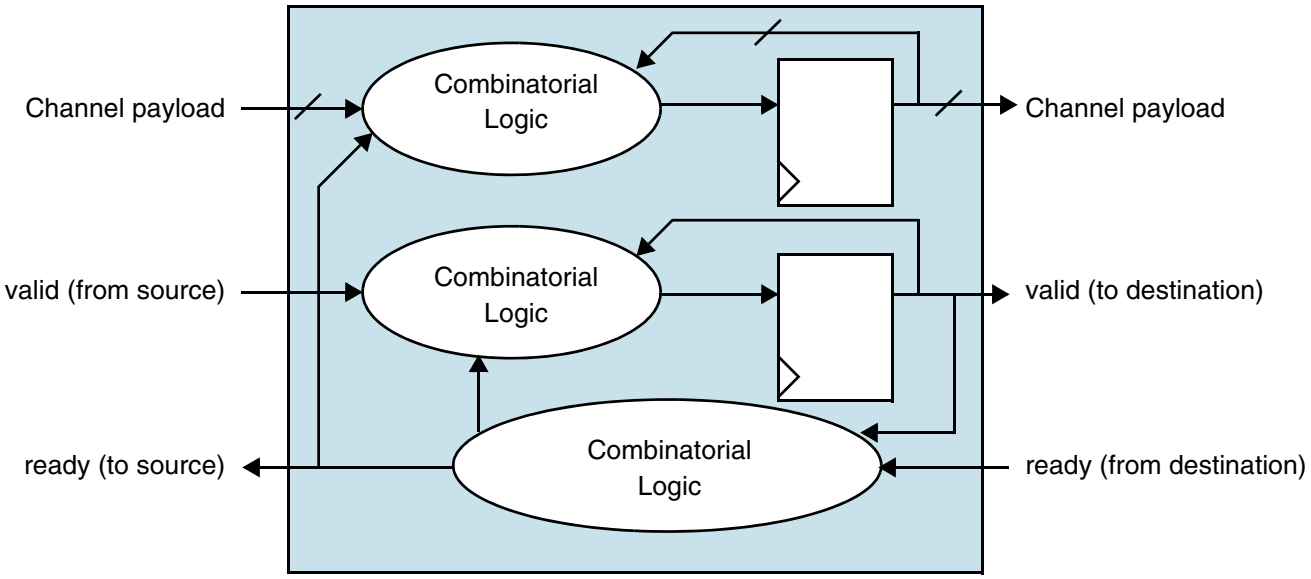
No registers are added to break the timing paths, including the forward control path, corresponding channel payload, and backward control path. DW_axi_rs input signals on the channels that are configured as the Pass Through Timing Mode are directly connected to the corresponding output signals. For this timing mode option, no additional cycle of latency is required as no logic exists on the signal paths.

Note It is illegal to configure all five channels to the Pass Through timing mode because, in such a case, an instance of DW_axi_rs is not required. If you try to configure all five channels with this timing mode, an error occurs.

2.2 Forward Registered Timing Mode

Registers are inserted into the forward control path and the corresponding channel payload to break the timing paths. No timing isolation register is inserted on the backward control path. Figure 2-2 shows a block diagram of this timing mode.

Figure 2-2 Forward Registered Timing Mode



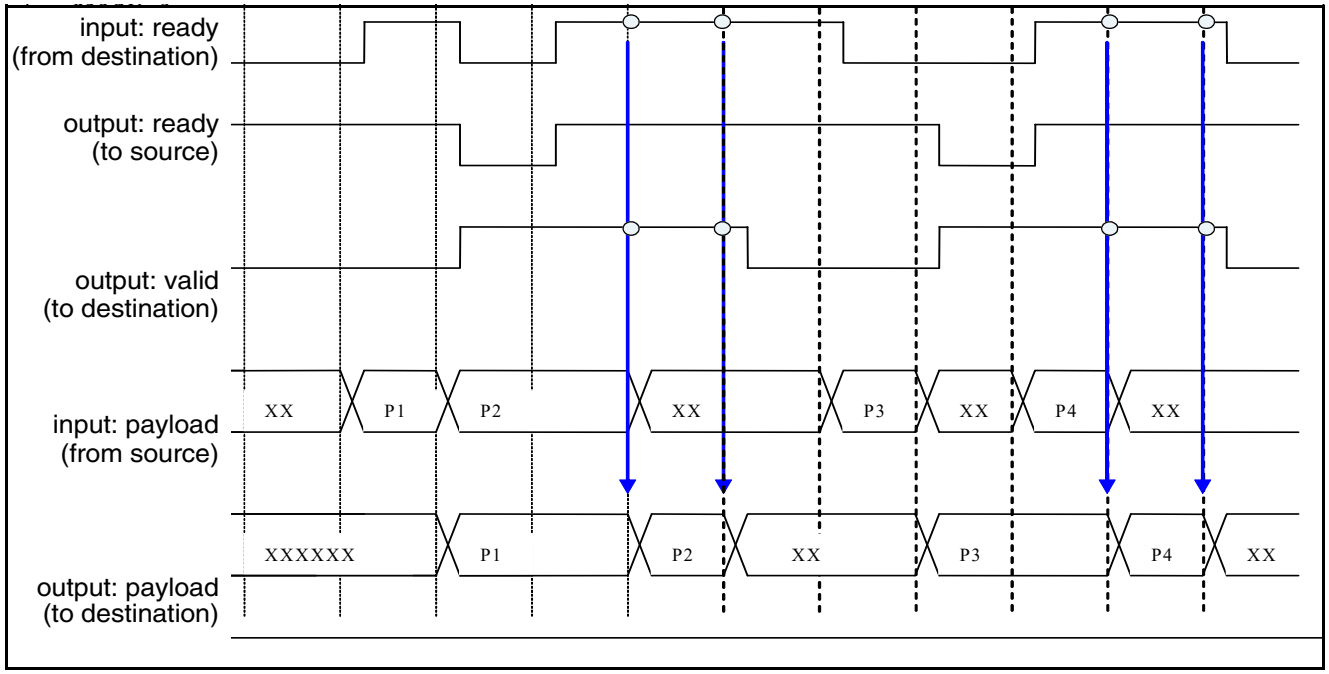
To ensure no throughput penalty and to break the forward control path and its channel payload, one pipeline stage must be used to keep one set of payload in advance when DW_axi_rs cannot forward a transfer. This pipeline stage causes the one cycle of additional latency on the forward control path and its channel payload. The basic scheme is as follows:

- If the pipeline stage is empty, then DW_axi_rs asserts ready (to source). The DW_axi_rs pipeline stage stores the AXI payload from the source, when valid (from source) is asserted.
- If the pipeline stage is full, then DW_axi_rs does not take in a new AXI payload from the source until the current contents of the pipeline stage are transferred out. There are two scenarios to consider:
  - **Pipeline stage not transferred out during current cycle –** If ready (from destination) is de-asserted in this cycle, then the pipeline stage is not transferred out. Therefore, ready (to source) is de-asserted by DW_axi_rs in this cycle and the AXI payload of the source is not loaded into the pipeline stage.
  - **Pipeline stage transferred out during current cycle –** If ready (from destination) is asserted in this cycle, then the pipeline stage is transferred out. Therefore, ready (to source) is asserted in this cycle. If valid (from source) is asserted, then the AXI payload of the source is transferred to DW_axi_rs in this cycle.

As the forward control path and its corresponding channel payload are registered, there is one cycle of latency between valid (from source) and valid (to destination) and between payload (from source) and payload (to destination).

After reset, the pipeline stage is empty, which means DW_axi_rs can accept payload from the source. Therefore, ready (to source) signal of the source is asserted, as illustrated in Figure 2-3. The blue arrows indicate the cycles that the destination valid/ready signals are both asserted. Therefore, those are the cycles that the pipeline stage is transferred out to the destination.

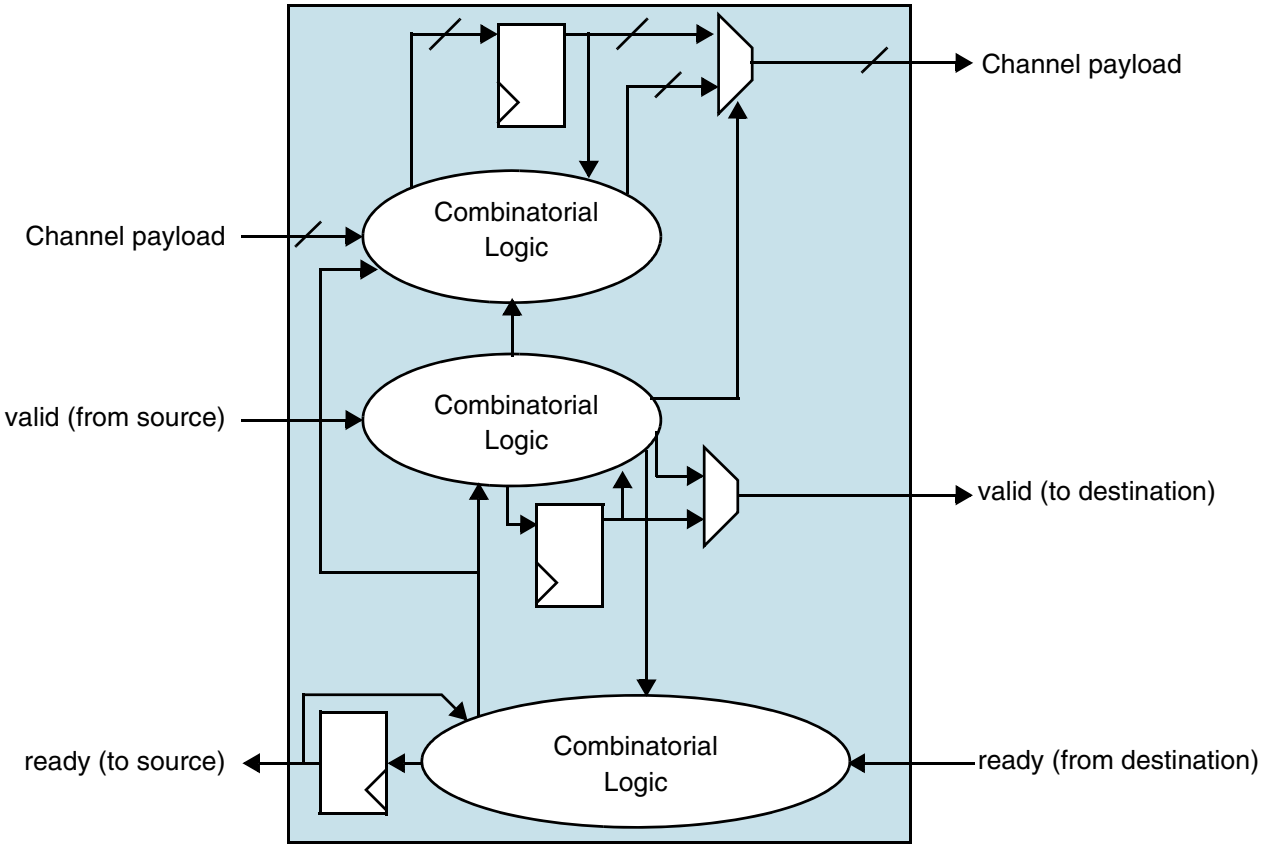
Figure 2-3 Example Timing Diagram of Forward Registered Timing Mode



2.3 Backward Registered Timing Mode

No registers are inserted to break the forward control path and the corresponding channel payload. However, the backward control path, specifically the ready control signal path shown in Figure 2-4, is broken by inserting a register. For this timing mode option, one additional cycle of latency is needed for the backward control path.

Figure 2-4 Backward Registered Timing Mode



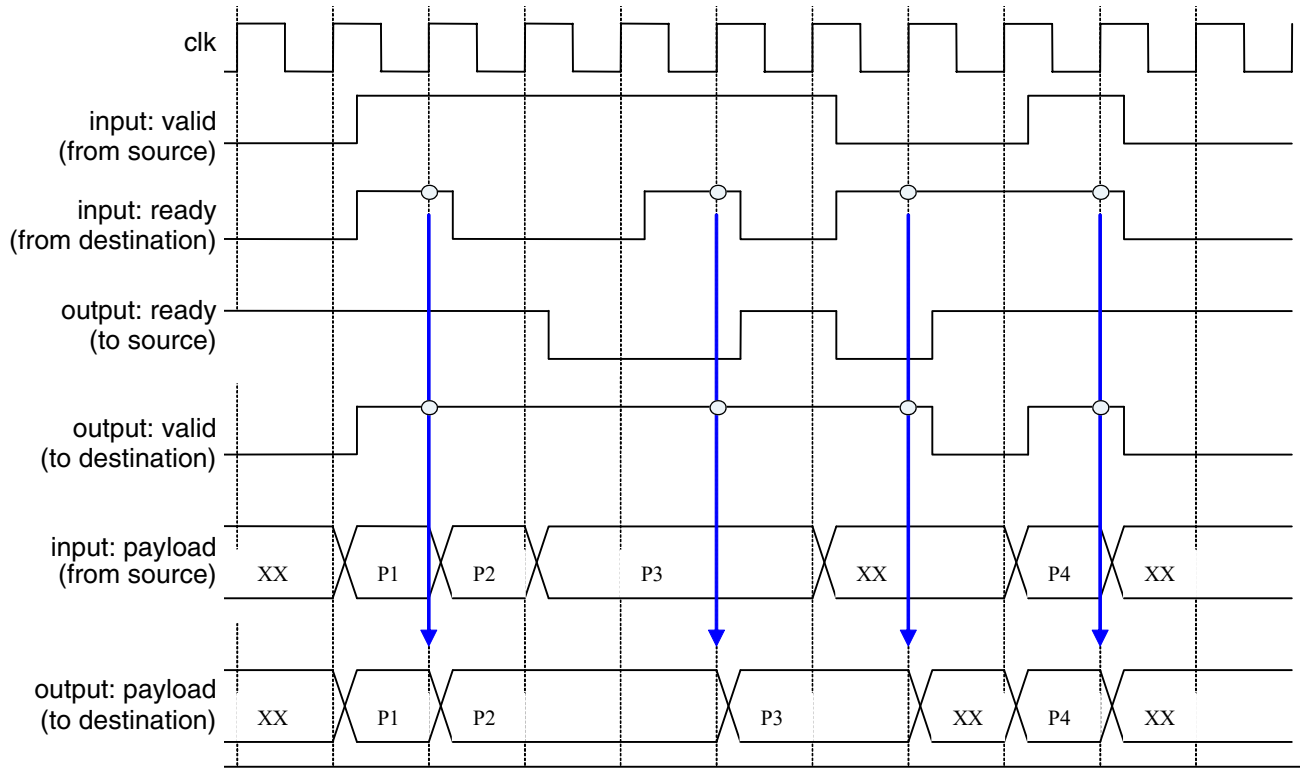
In this mode, a pipeline stage is needed to store payload to maintain maximum throughput on the channel. This pipeline stage does not eliminate the combinatorial path on the forward control path or channel payload path. The basic scheme is as follows:

- If the pipeline stage is empty, DW_axi_rs asserts the ready (to source). There are two scenarios to consider for whether the DW_axi_rs pipeline stage is loaded or not, when valid (from source) is asserted:
  - If ready (from destination) is asserted, then the destination accepts the payload in this cycle, and the pipeline stage is not loaded.
  - If ready (from destination) is de-asserted, then the pipeline stage is loaded with the payload and valid (to destination) is asserted on the next cycle. The pipeline stage remains full until ready (from destination) is asserted. Latency is inserted in this case, but only because the destination is not ready to accept the payload.

- If the pipeline stage is full, DW_axi_rs stalls the source by de-asserting the ready (to source) signal of the source. The pipeline stage empties only after the ready (from destination) is asserted, and the pipeline stage is transferred out to the destination. The pipeline stage will be empty in the following cycle, so ready (to source) signal is re-asserted at that time.

The backward registered timing mode only breaks the backward control path, specifically the ready control signal path. The ready (to source) signal is asserted as the pipeline stage is empty after reset. The timing diagram for this timing mode is illustrated in Figure 2-5. The blue arrows indicate the cycles in which the destination valid/ready signals are both asserted. Therefore, those are the cycles that the destination payload is accepted by the destination.

Figure 2-5 Example Timing Diagram of Backward Registered Timing Mode

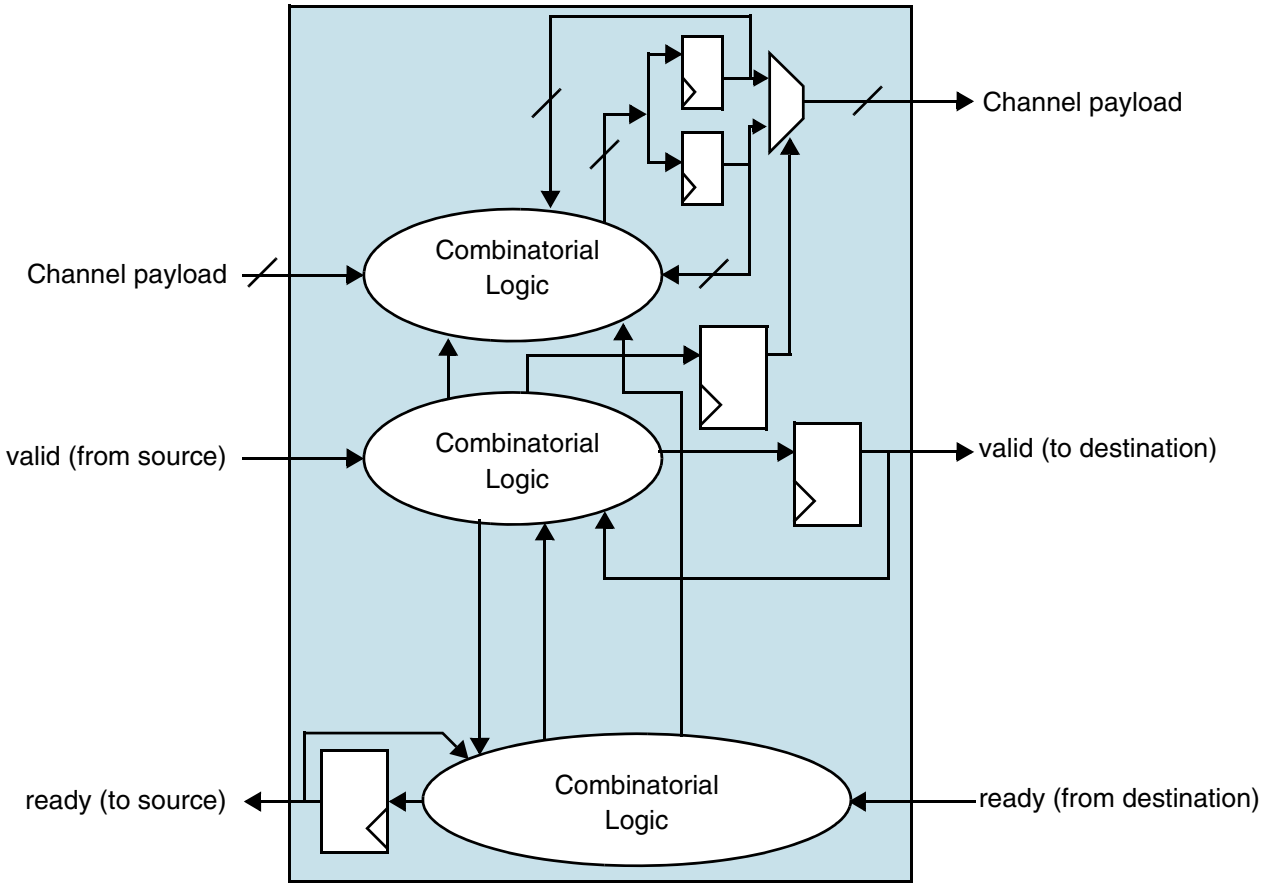


2.4 Fully Registered Timing Mode

Registers are inserted to break the timing paths of all signals, including the forward control path and its corresponding channel payload and the backward control path. In fully registered timing mode, no combinatorial paths exist between the signal source and destination.

The fully registered timing mode provides the best possible timing performance improvement to a system because it registers both the forward and backward control paths. The block diagram for this mode is illustrated in Figure 2-6.

Figure 2-6 Fully Registered Timing Mode



In this mode, a double pipeline stage is needed to store the channel payload of two transfers to ensure no throughput penalty. The basic scheme is as follows:

- If both pipeline stages are empty, then DW_axi_rs asserts the ready (to source) signal. This means the DW_axi_rs pipeline stage stores the AXI payload from the source, when valid (from source) is asserted. The cycle after valid (from source) is asserted, one pipeline stage becomes full.
- If one pipeline stage is full, then DW_axi_rs asserts the ready (to source) signal. This means the DW_axi_rs pipeline stage stores the AXI payload from the source, when valid (from source) is asserted. There are four scenarios to consider:
  - **Transfer pipeline stage out (to destination), no transfer pipeline stage in (from source)** –If the ready (from destination) is asserted, then the destination accepts the one full pipeline stage's payload in this cycle. This empties the one full pipeline stage. If valid (from source) is not asserted in this cycle, no pipeline stage is filled from the DW_axi_rs source this cycle. The result is both pipeline stages are empty at the completion of the cycle.
  - **Transfer pipeline stage out (to destination), transfer pipeline stage in (from source)** – If the ready (from destination) is asserted, then the destination accepts the one full pipeline stage's payload this cycle. This empties the one full pipeline stage. But if valid (from source) is asserted in this cycle, one pipeline stage is filled in this cycle, as well. The result is one pipeline stage is full at the completion of the cycle.

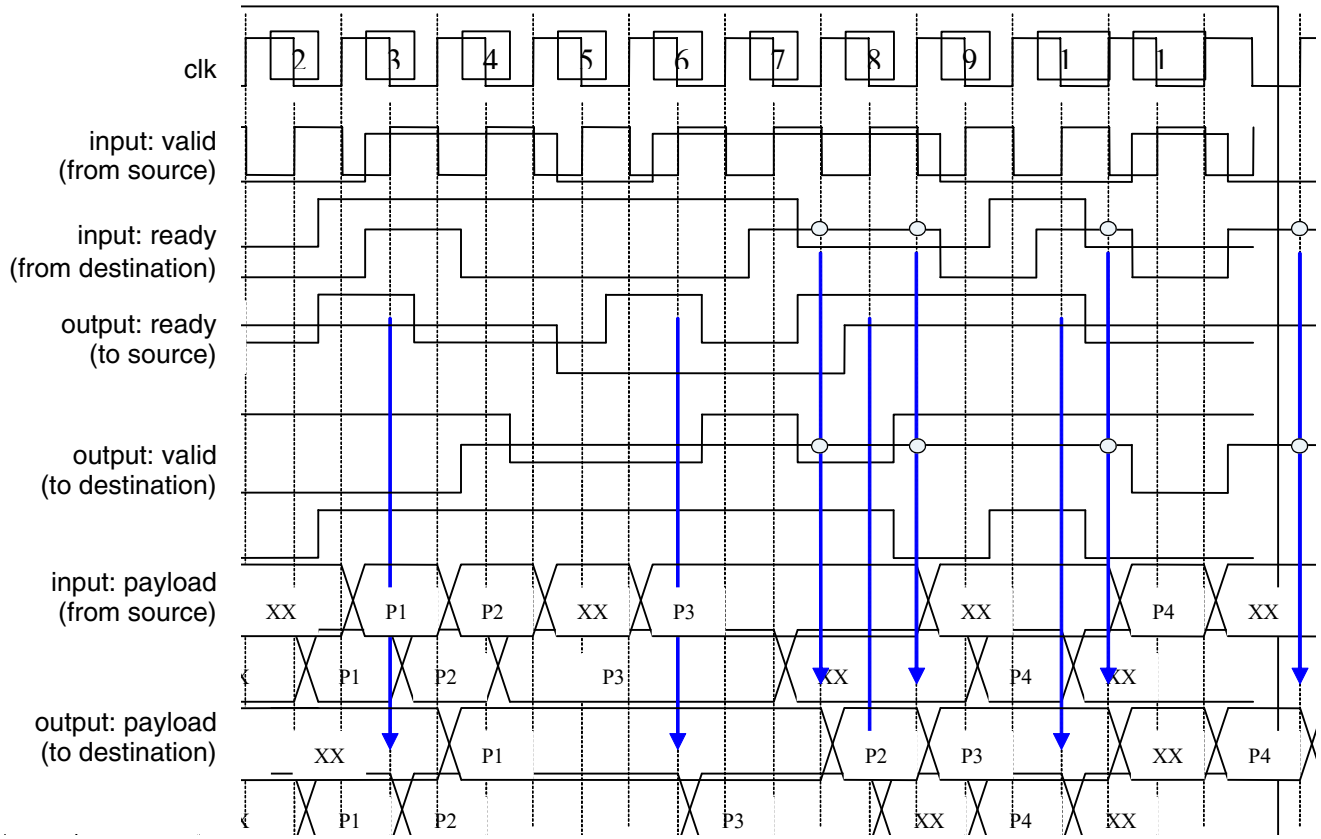
- **No transfer pipeline stage out (to destination), no transfer pipeline stage in (from source)** If the ready (from destination) is de-asserted, then the full pipeline stage is not transferred out in this cycle. If valid (from source) is not asserted in this cycle, then there is no change to the DW_axi_rs pipeline stages this cycle. The pipeline stage remains full until ready (from destination) is asserted. The result is one pipeline stage is full at the completion of the cycle.
- **No transfer pipeline stage out (to destination), transfer pipeline stage in (from source) –** If the ready (from destination) is de-asserted, then the full pipeline stage is not transferred out in this cycle. But if valid (from source) is asserted this cycle, the second pipeline stage is filled in this cycle. The first filled pipeline stage remains full until ready (from destination) is asserted. The second filled pipeline stage is not presented to the destination until the first filled pipeline stage is transferred out. The result is both pipeline stages are full at the completion of the cycle.

- If both pipeline stages are full, then DW_axi_rs de-asserts the ready (to source) signal. This means the DW_axi_rs pipeline stage cannot store the AXI payload from the source, when valid (from source) is asserted.
- If either pipeline stage is full, valid (to destination) is asserted.
- Transfers are issued to the destination in the same order as received from the source by DW_axi_rs.

The Fully Registered timing mode breaks the forward control path, the corresponding channel, and the backward control path by inserting registers. Therefore, there is one cycle of latency between all the forward control path input signals/payload and their corresponding output signals/payload and between all the backward control path input signals and their corresponding output signals. After reset, the ready (to source) is asserted because the two pipeline stages are empty after reset.

The timing diagram for the fully registered timing mode is shown in Figure 2-7. The blue arrows indicate the cycles in which the destination valid/ready signals are both asserted. Therefore, those are the cycles that the destination payload is accepted by the destination.

Figure 2-7 Example Timing Diagram of Fully Registered Timing Mode



2.5 Sideband/User Signals

DW_axi_rs has optional sideband/user signals that can be included on the I/O by setting configuration parameters. These signals are named as sideband signals in AXI3 configurations and user signals in AXI4 configurations. (While sideband signals are not defined by the AMBA 3 AXI protocol, user signals are defined by AMBA 4 AXI protocol.) Sideband/user signals can be configured for each channel independently. An example use of these signals is parity for data. DW_axi_rs does nothing with these signals except pass them to/from the master and slave along with the other payload information.

For more information about configuring DW_axi_rs to include these signals, see "Parameter Descriptions" on page 27. For more information about the sideband signals when included on the I/O, see "Signal Descriptions" on page 35.

2.6 Register Slice Interface Signal Options

DW_axi_rs has four configuration parameters that set ID width, address width, data width, and burst length width for all five AXI channels. The clock for both AXI master and slave must be the same. For more information about these parameters, see "Register Slice Interface Options Parameters" on page 30.

3 Parameter Descriptions

This chapter details all the configuration parameters. **You can use the coreConsultant GUI configuration reports to determine the actual configured state of the controller.** Some expressions might refer to TCL functions or procedures (sometimes identified as **<functionof>)** that coreConsultant uses to make calculations. The exact formula used by these TCL functions is not provided in this chapter. However, when you configure the controller in coreConsultant, all TCL functions and parameters are evaluated completely; and the resulting values are displayed where appropriate in the coreConsultant GUI reports.

The parameter descriptions in this chapter include the **Enabled:** attribute which indicates the values required to be set on other parameters before you can change the value of this parameter.

These tables define all of the configuration options for this component.

3.1 Timing Mode Options Parameters

Table 3-1 Timing Mode Options Parameters

Label	Description
Timing Mode Options	
Read Address Channel	Timing mode option for read address channel. **Values:** ■ PASS-THROUGH (0) ■ FORWARD (1) ■ FULL (2) ■ BACKWARD (3) **Default Value:** FORWARD **Enabled:** Always **Parameter Name:** RS_AR_TMO
Write Address Channel	Timing mode option for write address channel. **Values:** ■ PASS-THROUGH (0) ■ FORWARD (1) ■ FULL (2) ■ BACKWARD (3) **Default Value:** FORWARD **Enabled:** Always **Parameter Name:** RS_AW_TMO
Read Data Channel	Timing mode option for read data channel. **Values:** ■ PASS-THROUGH (0) ■ FORWARD (1) ■ FULL (2) ■ BACKWARD (3) **Default Value:** FORWARD **Enabled:** Always **Parameter Name:** RS_R_TMO

Table 3-1 Timing Mode Options Parameters (Continued)

Label	Description
Write Data Channel	Timing mode option for write data channel. **Values:** ■ PASS-THROUGH (0) ■ FORWARD (1) ■ FULL (2) ■ BACKWARD (3) **Default Value:** FORWARD **Enabled:** Always **Parameter Name:** RS_W_TMO
Write Response Channel	Timing mode option for the write response channel. **Values:** ■ PASS-THROUGH (0) ■ FORWARD (1) ■ FULL (2) ■ BACKWARD (3) **Default Value:** FORWARD **Enabled:** Always **Parameter Name:** RS_B_TMO

3.2 Register Slice Interface Options Parameters

Table 3-2 Register Slice Interface Options Parameters

Label	Description
Interface	
Select Interface AXI3/AXI4/ACELITE.	Select RS Interface Type as AXI3, AXI4 or ACE-Lite. By default, DW_axi_rs supports AXI3. **Values:** ■ AXI3 (0) ■ AXI4 (1) ■ ACELITE (2) **Default Value:** AXI3 **Enabled:** DWC-AMBA-Fabric-Source License required **Parameter Name:** RS_AXI_INTERFACE_TYPE
ID Width	Width of transaction ID field on DW_axi_rs. **Values:** 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16 **Default Value:** 4 **Enabled:** Always **Parameter Name:** RS_IDW
Address Width	Read/write address width. **Values:** 32, ..., 64 **Default Value:** 32 **Enabled:** Always **Parameter Name:** RS_AW
Data Width	Read/write data width for read and write channels. This sets the width of the arlen/awlen signals. No distinction is made between read and write channels. **Values:** 8, 16, 32, 64, 128, 256, 512 **Default Value:** 32 **Enabled:** Always **Parameter Name:** RS_DW
AXI Burst Length Width	This is the width of the burst length bus for both read and write address channels i.e. Sets the width of the arlen/awlen bus's. **Values:** 4, 5, 6, 7, 8 **Default Value:** 4 **Enabled:** Always **Parameter Name:** RS_BLW

Table 3-2 Register Slice Interface Options Parameters (Continued)

Label	Description
Include QoS Signals?	If set true, the primary port master and secondary port slave write and read address channels have QoS signals (awqos_*/arqos_*) on the I/O. **Values:** ■ false (0) ■ true (1) **Default Value:** false **Enabled:** DWC-AMBA-Fabric-Source License && RS_AXI_INTERFACE_TYPE != 0 **Parameter Name:** RS_HAS_QOS
Include Region Signals?	If set true, the primary port master and secondary port slave write and read address channels have region signals (awregion_*/arregion_*) on the I/O. **Values:** ■ false (0) ■ true (1) **Default Value:** false **Enabled:** DWC-AMBA-Fabric-Source License && RS_AXI_INTERFACE_TYPE != 0 **Parameter Name:** RS_HAS_REGION

3.3 SideBand/User Bus Support Parameters

Table 3-3 SideBand/User Bus Support Parameters

Label	Description
AXI Channel SideBand/User Bus Support	
Write Address Channel Sideband/User	If true, all master and slave write address channels have an associated sideband (AXI3) or user (AXI4/ACE-Lite) bus. **Values:** ■ Disable (0) ■ Enable (1) **Default Value:** Disable **Enabled:** Always **Parameter Name:** RS_HAS_AWSB
Write Address Channel Sideband/User Width	When the RS_HAS_AWSB parameter is set to True, the write address channel user/sideband signal bus width can be set. **Values:** 1, ..., 256 **Default Value:** 4 **Enabled:** RS_HAS_AWSB == 1 **Parameter Name:** RS_AW_SBW
Write Data Channel Sideband/User	If true, all master and slave write data channels have an associated sideband (AXI3) or user (AXI4/ACE-Lite) bus. **Values:** ■ Disable (0) ■ Enable (1) **Default Value:** Disable **Enabled:** Always **Parameter Name:** RS_HAS_WSB
Write Data Channel Sideband/User Width	When the RS_HAS_WSB parameter is set to True, the write data channel user/sideband signal bus width can be set. **Values:** 1, ..., 256 **Default Value:** 4 **Enabled:** RS_HAS_WSB == 1 **Parameter Name:** RS_W_SBW

Table 3-3 SideBand/User Bus Support Parameters (Continued)

Label	Description
Write Response Channel Sideband/User	If true, all master and slave write response channels have an associated sideband (AXI3) or user (AXI4/ACE-Lite) bus. **Values:** ■ Disable (0) ■ Enable (1) **Default Value:** Disable **Enabled:** Always **Parameter Name:** RS_HAS_BSB
Write Response Channel Sideband/User Width	When the RS_HAS_BSB parameter is set to True, the write response channel user/sideband signal bus width can be set. **Values:** 1, ..., 256 **Default Value:** 4 **Enabled:** RS_HAS_BSB == 1 **Parameter Name:** RS_B_SBW
Read Address Channel Sideband/User	If true, all master and slave read address channels have an associated sideband (AXI3) or user (AXI4/ACE-Lite) bus. **Values:** ■ Disable (0) ■ Enable (1) **Default Value:** Disable **Enabled:** Always **Parameter Name:** RS_HAS_ARSB
Read Address Channel Sideband/User Width	When the RS_HAS_ARSB parameter is set to True, the read address channel user/sideband signal bus width can be set. **Values:** 1, ..., 256 **Default Value:** 4 **Enabled:** RS_HAS_ARSB == 1 **Parameter Name:** RS_AR_SBW
Read Data Channel Sideband/User	If true, all master and slave read data channels have an associated sideband (AXI3) or user (AXI4/ACE-Lite) bus. **Values:** ■ Disable (0) ■ Enable (1) **Default Value:** Disable **Enabled:** Always **Parameter Name:** RS_HAS_RSB

Table 3-3 SideBand/User Bus Support Parameters (Continued)

Label	Description
Read Data Channel Sideband/User Width	When the RS_HAS_RSB parameter is set to True, the read address channel user/sideband signal bus width can be set. **Values:** 1, ..., 256 **Default Value:** 4 **Enabled:** RS_HAS_RSB == 1 **Parameter Name:** RS_R_SBW

4
Signal Descriptions

This chapter details all possible I/O signals in the controller. For configurable IP titles, your actual configuration might not contain all of these signals.

Inputs are on the left of the signal diagrams; outputs are on the right.

Attention: For configurable IP titles, do not use this document to determine the exact I/O footprint of the controller. It is for reference purposes only.

When you configure the controller in coreConsultant, you must access the I/O signals for your actual configuration at workspace/report/IO.html or workspace/report/IO.xml after you have completed the report creation activity. That report comes from the exact same source as this chapter but removes all the I/O signals that are not in your actual configuration. This does not apply to non-configurable IP titles. In addition, all parameter expressions are evaluated to actual values. Therefore, the widths might change depending on your actual configuration.

Some expressions might refer to TCL functions or procedures (sometimes identified as **<functionof>)** that coreConsultant uses to make calculations. The exact formula used by these TCL functions is not provided in this chapter. However, when you configure the controller in coreConsultant, all TCL functions and parameters are evaluated completely; and the resulting values are displayed where appropriate in the coreConsultant GUI reports.

In addition to describing the function of each signal, the signal descriptions in this chapter include the following information:

Active State: Indicates whether the signal is active high or active low. When a signal is not intended to be used in a particular application, then this signal needs to be tied or driven to the inactive state (opposite of the active state).

Registered: Indicates whether or not the signal is registered directly inside the IP boundary without intervening logic (excluding simple buffers). A value of No does not imply that the signal is not synchronous, only that there is some combinatorial logic between the signal's origin or destination register and the boundary of the controller. A value of N/A indicates that this information is not provided for this IP title.

Synchronous to: Indicates which clocks in the IP sample this input (drive for an output) when considering all possible configurations. A particular configuration might not have all of the clocks listed. This clock might not be the same as the clock that your application logic should use to clock (sample/drive) this pin. For more details, consult the clock section in the databook.

Exists: Names of configuration parameters that populate this signal in your configuration.

Validated by: Assertion or de-assertion of signals that validates the signal being described.

Attributes used with Synchronous To

- Clock name - The name of the clock that samples an input or drive and output.
- None - This attribute may be used for clock inputs, hard-coded outputs, feed-through (direct or combinatorial), dangling inputs, unused inputs and asynchronous outputs.
- Asynchronous - This attribute is used for asynchronous inputs and asynchronous resets.

The I/O signals are grouped as follows:

- Clock and Resets on page 37
- Master Port Read Address Channel on page 38
- Master Port Write Address Channel on page 42
- Master Port Write Data Channel on page 46
- Master Port Read Data Channel on page 48
- Master Port Write Response Channel on page 50
- Slave Port Read Address Channel on page 52
- Slave Port Write Address Channel on page 56
- Slave Port Write Data Channel on page 60
- Slave Port Read Data Channel on page 63
- Slave Port Write Response Channel on page 65

4.1 Clock and Resets Signals

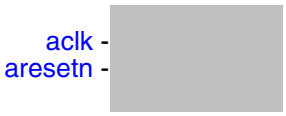


Table 4-1 Clock and Resets Signals

Port Name	I/O	Description
aclk	I	AXI Clock Signal. All input signals are sampled on the rising edge of aclk. All output signals must occur after the rising edge of aclk. There must be no combinatorial paths between input and output signals within any one master or slave interface, as specified by the AXI protocol. **Exists:** Always **Synchronous To:** None **Registered:** N/A **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
aresetn	I	AXI Reset Signal. The bus reset signal is active low and is used to reset the system and the bus. During reset, the following must occur: ■ A master interface must drive arvalid, awvalid, and wvalid low. ■ A slave interface must drive rvalid and bvalid low. Asynchronous assertion, synchronous de-assertion. The reset must be deasserted synchronously after the rising edge of aclk. DW_axi_rs does not contain logic to perform this synchronization, so it must be provided externally. **Exists:** Always **Synchronous To:** None **Registered:** N/A **Power Domain:** SINGLE_DOMAIN **Active State:** Low

4.2 Master Port Read Address Channel Signals

Table 4-2 Master Port Read Address Channel Signals

Port Name	I/O	Description
arready_m	O	Primary Port Read address ready. This signal indicates that the slave is ready to accept an address and associated control signals. ■ 1: Slave ready ■ 0: Slave not ready **Exists:** Always **Synchronous To:** aclk **Registered:** (RS_AR_TMO==2 || RS_AR_TMO==3) ? "Yes" : "No" **Power Domain:** SINGLE_DOMAIN **Active State:** High
arid_m[(RS_IDW-1):0]	I	Primary Port Read address ID. This signal is the identification tag for the read address group of signals. **Exists:** Always **Synchronous To:** aclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A

Table 4-2 Master Port Read Address Channel Signals (Continued)

Port Name	I/O	Description
araddr_m[(RS_AW-1):0]	I	Primary Port Read address. The read address bus gives the address of the first transfer in a read burst transaction. The associated control signals are used to determine the addresses of the remaining transfers in the burst. **Exists:** Always **Synchronous To:** aclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
arlen_m[(RS_BLW-1):0]	I	Primary Port Read Address Burst length. The burst length gives the exact number of transfers in a burst. This information determines the number of data transfers associated with the address. **Exists:** Always **Synchronous To:** aclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
arsize_m[2:0]	I	Primary Port Burst size. This signal indicates the size of each transfer in the burst. **Exists:** Always **Synchronous To:** aclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
arburst_m[1:0]	I	Primary Port Burst type for Read Address Channel. The burst type, coupled with the size information, details how the address for each transfer within the burst is calculated. **Exists:** Always **Synchronous To:** aclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
arlock_m[(RS_LTW-1):0]	I	Primary Port Lock type for Read Address Channel. This signal provides additional information about the atomic characteristics of the transfer. **Exists:** Always **Synchronous To:** aclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A

Table 4-2 Master Port Read Address Channel Signals (Continued)

Port Name	I/O	Description
arcache_m[3:0]	I	Primary Port Cache type for Read Address Channel. This signal indicates the bufferable, cacheable, write-through, write-back, and allocation attributes of the transaction. **Exists:** Always **Synchronous To:** aclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
arprot_m[2:0]	I	Primary Port Protection type for Read Address Channel. This signal indicates the normal, privileged, or secure protection level of the transaction and whether the transaction is a data access or an instruction access. **Exists:** Always **Synchronous To:** aclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
arvalid_m	I	Primary Port Read address valid. This signal indicates that a valid read address and control information are available. ■ 0: Address and control information unavailable ■ 1: Address and control information available **Exists:** Always **Synchronous To:** aclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** High
arqos_m[3:0]	I	Primary Port Quality of Service identifier, conveys priority information associated with each transaction at Read Address channel of Master port. **Exists:** (RS_HAS_QOS == 1) **Synchronous To:** aclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
arregion_m[3:0]	I	Primary Port Read Address Region identifier. Permits a single physical interface on a slave to be used for multiple logical interfaces. **Exists:** (RS_HAS_REGION == 1) **Synchronous To:** aclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A

Table 4-2 Master Port Read Address Channel Signals (Continued)

Port Name	I/O	Description
arsnoop_m[3:0]	I	Primary Port ACELITE signal. This signal indicates the transaction type for shareable read transactions. **Exists:** (RS_AXI_INTERFACE_TYPE == 2) **Synchronous To:** aclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
ardomain_m[1:0]	I	Primary Port ACELITE signal. This signal indicates the shareability domain of a read transaction. **Exists:** (RS_AXI_INTERFACE_TYPE == 2) **Synchronous To:** aclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
arbar_m[1:0]	I	Primary Port ACELITE signal. This signal indicates a read barrier transaction. **Exists:** (RS_AXI_INTERFACE_TYPE == 2) **Synchronous To:** aclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
arsideband_m[(RS_AR_SBW-1):0]	I	Primary Port Sideband signal for read address channel. **Exists:** (RS_HAS_ARSB == 1&&RS_AXI_INTERFACE_TYPE==0) **Synchronous To:** aclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
aruser_m[(RS_AR_SBW-1):0]	I	Primary Port User signal for read address channel. **Exists:** (RS_HAS_ARSB == 1&&RS_AXI_INTERFACE_TYPE!=0) **Synchronous To:** aclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A

4.3 Master Port Write Address Channel Signals

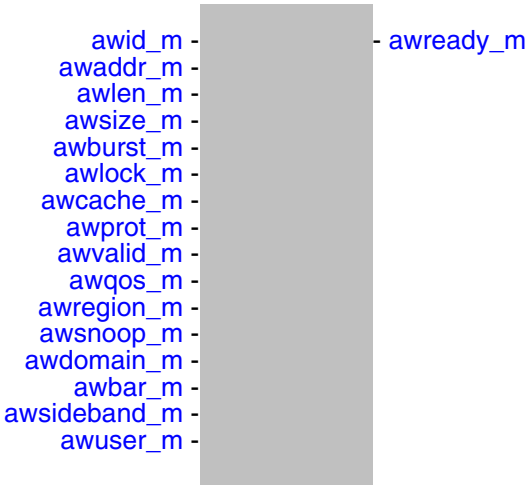


Table 4-3 Master Port Write Address Channel Signals

Port Name	I/O	Description
awready_m	O	Primary Port Write address ready. This signal indicates that the master is ready to accept an address and associated control signals. ■ 0: Master not ready ■ 1: Master ready **Exists:** Always **Synchronous To:** aclk **Registered:** (RS_AW_TMO==2 || RS_AW_TMO==3) ? "Yes" : "No" **Power Domain:** SINGLE_DOMAIN **Active State:** High
awid_m[(RS_IDW-1):0]	I	Primary Port Write address ID. This signal is the identification tag for the write address group of signals. **Exists:** Always **Synchronous To:** aclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A

Table 4-3 Master Port Write Address Channel Signals (Continued)

Port Name	I/O	Description
awaddr_m[(RS_AW-1):0]	I	Primary Port Write address. The write address bus gives the address of the first transfer in a write burst transaction. The associated control signals are used to determine the addresses of the remaining transfers in the burst. **Exists:** Always **Synchronous To:** aclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
awlen_m[(RS_BLW-1):0]	I	Primary Port Write Address Burst length. The burst length gives the exact number of transfers in a burst. This information determines the number of data transfers associated with the address. **Exists:** Always **Synchronous To:** aclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
awsize_m[2:0]	I	Primary Port Burst size. This signal indicates the size of each transfer in the burst. Byte lane strobes indicate exactly which byte lanes to update. **Exists:** Always **Synchronous To:** aclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
awburst_m[1:0]	I	Primary Port Burst type for Write Address Channel. The burst type, coupled with the size information, details how the address for each transfer within the burst is calculated. **Exists:** Always **Synchronous To:** aclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
awlock_m[(RS_LTW-1):0]	I	Primary Port Lock type for Write Address Channel. This signal provides additional information about the atomic characteristics of the transfer. **Exists:** Always **Synchronous To:** aclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A

Table 4-3 Master Port Write Address Channel Signals (Continued)

Port Name	I/O	Description
awcache_m[3:0]	I	Primary Port Cache type for Write Address Channel. This signal indicates the bufferable, cacheable, write-through, write-back, and allocation attributes of the transaction. **Exists:** Always **Synchronous To:** aclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
awprot_m[2:0]	I	Primary Port Protection type for Write Address Channel. This signal indicates the normal, privileged, or secure protection level of the transaction and whether the transaction is a data access or an instruction access. **Exists:** Always **Synchronous To:** aclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
awvalid_m	I	Primary Port Write address valid. This signal indicates that a valid write address and control information are available. ■ 0: Address and control information unavailable ■ 1: Address and control information available **Exists:** Always **Synchronous To:** aclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** High
awqos_m[3:0]	I	Primary Port Quality of Service identifier, conveying priority information associated with each transaction at the write address channel of the master port. **Exists:** (RS_HAS_QOS == 1) **Synchronous To:** aclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
awregion_m[3:0]	I	Primary Port Write Address Region identifier. Permits a single physical interface on a slave to be used for multiple logical interfaces. **Exists:** (RS_HAS_REGION == 1) **Synchronous To:** aclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A

Table 4-3 Master Port Write Address Channel Signals (Continued)

Port Name	I/O	Description
awsnoop_m[2:0]	I	Primary Port ACELITE signal. This signal indicates the transaction type for shareable write transactions. **Exists:** (RS_AXI_INTERFACE_TYPE == 2) **Synchronous To:** aclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
awdomain_m[1:0]	I	Primary Port ACELITE signal. This signal indicates the shareability domain of a write transaction. **Exists:** (RS_AXI_INTERFACE_TYPE == 2) **Synchronous To:** aclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
awbar_m[1:0]	I	Primary Port ACELITE signal. This signal indicates a write barrier transaction. **Exists:** (RS_AXI_INTERFACE_TYPE == 2) **Synchronous To:** aclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
awsideband_m[(RS_AW_SBW-1):0]	I	Optional. Primary Port Sideband signal for write address channel. **Exists:** (RS_HAS_AWSB == 1&&RS_AXI_INTERFACE_TYPE==0) **Synchronous To:** aclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
awuser_m[(RS_AW_SBW-1):0]	I	Primary Port User signal for write address channel. **Exists:** (RS_HAS_AWSB == 1&&RS_AXI_INTERFACE_TYPE!=0) **Synchronous To:** aclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A

4.4 Master Port Write Data Channel Signals

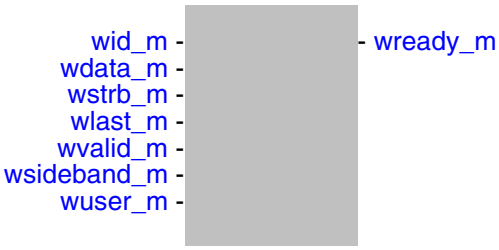


Table 4-4 Master Port Write Data Channel Signals

Port Name	I/O	Description
wready_m	O	Primary Port Write ready. This signal indicates that the master can accept the write data. ■ 0: Master not ready ■ 1: Master ready **Exists:** Always **Synchronous To:** aclk **Registered:** (RS_W_TMO==2 || RS_W_TMO==3) ? "Yes" : "No" **Power Domain:** SINGLE_DOMAIN **Active State:** High
wid_m[(RS_IDW-1):0]	I	Primary Port Write ID tag. This signal is the ID tag of the write data transfer. The WID value must match the AWID value of the write transaction. **Exists:** (RS_AXI_INTERFACE_TYPE ==0) **Synchronous To:** aclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
wdata_m[(RS_DW-1):0]	I	Primary Port Write data. **Exists:** Always **Synchronous To:** aclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A

Table 4-4 Master Port Write Data Channel Signals (Continued)

Port Name	I/O	Description
wstrb_m[((RS_DW/8)-1):0]	I	Primary Port Write strobes. This signal indicates which byte lanes to update in memory. There is one write strobe for each eight bits of the write data bus. Therefore, wstrb_m corresponds to wdata_m[(8 x i)+7:(8 x i)]. **Exists:** Always **Synchronous To:** aclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** High
wlast_m	I	Primary Port Write last. This signal indicates the last transfer in a write burst. **Exists:** Always **Synchronous To:** aclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** High
wvalid_m	I	Primary Port Write valid. This signal indicates that valid write data and strobes are available. ■ 0: Write data and strobes unavailable ■ 1: Write data and strobes available **Exists:** Always **Synchronous To:** aclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** High
wsideband_m[(RS_W_SBW-1):0]	I	Optional. Primary Port Sideband signal for write data channel. The signal name changes between the AXI3 and the AXI4/ACE-Lite configurations. **Exists:** (RS_HAS_WSB == 1&&RS_AXI_INTERFACE_TYPE==0) **Synchronous To:** aclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
wuser_m[(RS_W_SBW-1):0]	I	Optional. Primary Port User signal for write data channel. The signal name changes between the AXI3 and the AXI4/ACE-Lite configurations. **Exists:** (RS_HAS_WSB == 1&&RS_AXI_INTERFACE_TYPE!=0) **Synchronous To:** aclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A

4.5 Master Port Read Data Channel Signals

Table 4-5 Master Port Read Data Channel Signals

Port Name	I/O	Description
rid_m[(RS_IDW-1):0]	O	Primary Port Read ID tag. This signal is the ID tag of the read data group of signals. The RID value is generated by the slave and must match the ARID value of the read transaction to which it is responding. **Exists:** Always **Synchronous To:** aclk **Registered:** RS_R_TMO==1 ? "Yes" : "No" **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
rdata_m[(RS_DW-1):0]	O	Primary Port Read data. **Exists:** Always **Synchronous To:** aclk **Registered:** RS_R_TMO==1 ? "Yes" : "No" **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
rresp_m[1:0]	O	Primary Port Read response. This signal indicates the status of the read transfer. The supported responses are OKAY, EXOKAY, SLVERR, and DECERR. **Exists:** Always **Synchronous To:** aclk **Registered:** RS_R_TMO==1 ? "Yes" : "No" **Power Domain:** SINGLE_DOMAIN **Active State:** N/A

Table 4-5 Master Port Read Data Channel Signals (Continued)

Port Name	I/O	Description
rlast_m	O	Primary Port Read last. This signal indicates the last transfer in a read burst. **Exists:** Always **Synchronous To:** aclk **Registered:** RS_R_TMO==1 ? "Yes" : "No" **Power Domain:** SINGLE_DOMAIN **Active State:** High
rvalid_m	O	Primary Port Read valid. This signal indicates that the required read data is available and the read transfer can complete. ■ 0: Read data not available ■ 1: Read data available **Exists:** Always **Synchronous To:** aclk **Registered:** (RS_R_TMO==1 || RS_R_TMO==2) ? "Yes" : "No" **Power Domain:** SINGLE_DOMAIN **Active State:** High
rsideband_m[(RS_R_SBW-1):0]	O	Primary Port Sideband signal for read data channel. **Exists:** (RS_HAS_RSB == 1&&RS_AXI_INTERFACE_TYPE==0) **Synchronous To:** aclk **Registered:** RS_R_TMO==1 ? "Yes" : "No" **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
ruser_m[(RS_R_SBW-1):0]	O	Primary Port User signal for read data channel. **Exists:** (RS_HAS_RSB == 1&&RS_AXI_INTERFACE_TYPE!=0) **Synchronous To:** aclk **Registered:** RS_R_TMO==1 ? "Yes" : "No" **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
rready_m	I	Primary Port Read ready. This signal indicates that the master can accept the read data and response information. ■ 0: Master not ready ■ 1: Master ready **Exists:** Always **Synchronous To:** aclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** High

4.6 Master Port Write Response Channel Signals

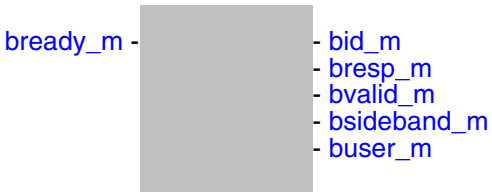


Table 4-6 Master Port Write Response Channel Signals

Port Name	I/O	Description
bid_m[(RS_IDW-1):0]	O	Primary Port Response ID tag. This signal is the ID tag of the write response. The BID value must match the AWID value of the write transaction to which the slave is responding. **Exists:** Always **Synchronous To:** aclk **Registered:** RS_B_TMO==1 ? "Yes" : "No" **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
bresp_m[1:0]	O	Primary Port Write response. This signal indicates the status of the write transaction. The supported responses are OKAY, EXOKAY, SLVERR, and DECERR. **Exists:** Always **Synchronous To:** aclk **Registered:** RS_B_TMO==1 ? "Yes" : "No" **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
bvalid_m	O	Primary Port Write response valid. This signal indicates that the valid write response data is available. ■ 0: Write response unavailable ■ 1: Write response available **Exists:** Always **Synchronous To:** aclk **Registered:** (RS_B_TMO==1 || RS_B_TMO==2) ? "Yes" : "No" **Power Domain:** SINGLE_DOMAIN **Active State:** High

Table 4-6 Master Port Write Response Channel Signals (Continued)

Port Name	I/O	Description
bsideband_m[(RS_B_SBW-1):0]	O	Primary Port Sideband signal for write response. The signal name changes between the AXI3 and AXI4/ACE-Lite configurations. ■ When the AXI3 interface is enabled, this signal is named bsideband_m. ■ When the AXI4/ACE-Lite interface is enabled, this signal is named buser_m. **Exists:** (RS_HAS_BSB == 1&&RS_AXI_INTERFACE_TYPE==0) **Synchronous To:** aclk **Registered:** RS_B_TMO==1 ? "Yes" : "No" **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
buser_m[(RS_B_SBW-1):0]	O	Primary Port User signal for write response. The signal name changes between the AXI3 and AXI4/ACE-Lite configurations. ■ When the AXI3 interface is enabled, this signal is named bsideband_m. ■ When the AXI4/ACE-Lite interface is enabled, this signal is named buser_m. **Exists:** (RS_HAS_BSB == 1&&RS_AXI_INTERFACE_TYPE!=0) **Synchronous To:** aclk **Registered:** RS_B_TMO==1 ? "Yes" : "No" **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
bready_m	I	Primary Port Response ready. This signal indicates that the master can accept the response information. ■ 0: Master not ready ■ 1: Master ready **Exists:** Always **Synchronous To:** aclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** High

4.7 Slave Port Read Address Channel Signals

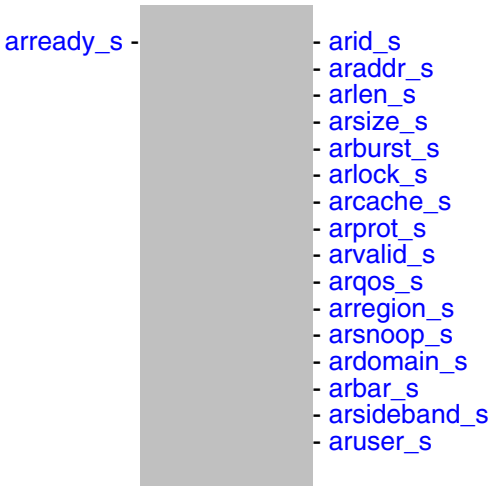


Table 4-7 Slave Port Read Address Channel Signals

Port Name	I/O	Description
arid_s[(RS_IDW-1):0]	O	Secondary Port Read address ID. This signal is the identification tag for the read address group of signals. **Exists:** Always **Synchronous To:** aclk **Registered:** RS_AR_TMO==1 ? "Yes" : "No" **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
araddr_s[(RS_AW-1):0]	O	Secondary Port Read address. The read address bus gives the address of the first transfer in a read burst transaction. The associated control signals are used to determine the addresses of the remaining transfers in the burst. **Exists:** Always **Synchronous To:** aclk **Registered:** RS_AR_TMO==1 ? "Yes" : "No" **Power Domain:** SINGLE_DOMAIN **Active State:** N/A

Table 4-7 Slave Port Read Address Channel Signals (Continued)

Port Name	I/O	Description
arlen_s[(RS_BLW-1):0]	O	Secondary Port Read Address Burst length. The burst length gives the exact number of transfers in a burst. This information determines the number of data transfers associated with the address. **Exists:** Always **Synchronous To:** aclk **Registered:** RS_AR_TMO==1 ? "Yes" : "No" **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
arsize_s[2:0]	O	Secondary Port Burst size. This signal indicates the size of each transfer in the burst. **Exists:** Always **Synchronous To:** aclk **Registered:** RS_AR_TMO==1 ? "Yes" : "No" **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
arburst_s[1:0]	O	Secondary Port Burst type for Read Channel. The burst type, coupled with the size information, details how the address for each transfer within the burst is calculated. **Exists:** Always **Synchronous To:** aclk **Registered:** RS_AR_TMO==1 ? "Yes" : "No" **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
arlock_s[(RS_LTW-1):0]	O	Secondary Port Lock type for Read Channel. This signal provides additional information about the atomic characteristics of the transfer. **Exists:** Always **Synchronous To:** aclk **Registered:** RS_AR_TMO==1 ? "Yes" : "No" **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
arcache_s[3:0]	O	Secondary Port Cache type for Read Channel. This signal indicates the bufferable, cacheable, write-through, write-back, and allocation attributes of the transaction. **Exists:** Always **Synchronous To:** aclk **Registered:** RS_AR_TMO==1 ? "Yes" : "No" **Power Domain:** SINGLE_DOMAIN **Active State:** N/A

Table 4-7 Slave Port Read Address Channel Signals (Continued)

Port Name	I/O	Description
arprot_s[2:0]	O	Secondary Port Protection type for Read Channel. This signal indicates the normal, privileged, or secure protection level of the transaction and whether the transaction is a data access or an instruction access. **Exists:** Always **Synchronous To:** aclk **Registered:** RS_AR_TMO==1 ? "Yes" : "No" **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
arvalid_s	O	Secondary Port Read address valid. This signal indicates that a valid read address and control information are available. ■ 0: Address and control information unavailable ■ 1: Address and control information available **Exists:** Always **Synchronous To:** aclk **Registered:** (RS_AR_TMO==1 || RS_AR_TMO==2) ? "Yes" : "No" **Power Domain:** SINGLE_DOMAIN **Active State:** High
arqos_s[3:0]	O	Secondary Port Quality of Service identifier, conveys priority information associated with each transaction at Read Address channel of Master port. **Exists:** (RS_HAS_QOS == 1) **Synchronous To:** aclk **Registered:** RS_AR_TMO==1 ? "Yes" : "No" **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
arregion_s[3:0]	O	Secondary Port Read Address Region identifier. Permits a single physical interface on a slave to be used for multiple logical interfaces. **Exists:** (RS_HAS_REGION == 1) **Synchronous To:** aclk **Registered:** RS_AR_TMO==1 ? "Yes" : "No" **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
arsnoop_s[3:0]	O	Secondary Port ACELITE signal. This signal indicates the transaction type for shareable read transactions. **Exists:** (RS_AXI_INTERFACE_TYPE == 2) **Synchronous To:** aclk **Registered:** RS_AR_TMO==1 ? "Yes" : "No" **Power Domain:** SINGLE_DOMAIN **Active State:** N/A

Table 4-7 Slave Port Read Address Channel Signals (Continued)

Port Name	I/O	Description
ardomain_s[1:0]	O	Secondary Port ACELITE signal. This signal indicates the shareability domain of a read transaction. **Exists:** (RS_AXI_INTERFACE_TYPE == 2) **Synchronous To:** aclk **Registered:** RS_AR_TMO==1 ? "Yes" : "No" **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
arbar_s[1:0]	O	Secondary Port ACELITE signal. This signal indicates a read barrier transaction. **Exists:** (RS_AXI_INTERFACE_TYPE == 2) **Synchronous To:** aclk **Registered:** RS_AR_TMO==1 ? "Yes" : "No" **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
arsideband_s[(RS_AR_SBW-1):0]	O	Secondary Port Sideband signal for read address channel. **Exists:** (RS_HAS_ARSB == 1&&RS_AXI_INTERFACE_TYPE==0) **Synchronous To:** aclk **Registered:** RS_AR_TMO==1 ? "Yes" : "No" **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
aruser_s[(RS_AR_SBW-1):0]	O	Secondary Port User signal for read address channel. **Exists:** (RS_HAS_ARSB == 1&&RS_AXI_INTERFACE_TYPE!=0) **Synchronous To:** aclk **Registered:** RS_AR_TMO==1 ? "Yes" : "No" **Power Domain:** SINGLE_DOMAIN
arready_s	I	Secondary Port Read address ready. This signal indicates that the slave is ready to accept an address and associated control signals. ■ 0: Slave not ready ■ 1: Slave ready **Exists:** Always **Synchronous To:** aclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** High

4.8 Slave Port Write Address Channel Signals

Table 4-8 Slave Port Write Address Channel Signals

Port Name	I/O	Description
awid_s[(RS_IDW-1):0]	O	Secondary Port Write address ID. This signal is the identification tag for the write address group of signals. **Exists:** Always **Synchronous To:** aclk **Registered:** RS_AW_TMO==1 ? "Yes" : "No" **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
awaddr_s[(RS_AW-1):0]	O	Secondary Port Write address. The write address bus gives the address of the first transfer in a write burst transaction. The associated control signals are used to determine the addresses of the remaining transfers in the burst. **Exists:** Always **Synchronous To:** aclk **Registered:** RS_AW_TMO==1 ? "Yes" : "No" **Power Domain:** SINGLE_DOMAIN **Active State:** N/A

Table 4-8 Slave Port Write Address Channel Signals (Continued)

Port Name	I/O	Description
awlen_s[(RS_BLW-1):0]	O	Secondary Port Write address Burst length. The burst length gives the exact number of transfers in a burst. This information determines the number of data transfers associated with the address. **Exists:** Always **Synchronous To:** aclk **Registered:** RS_AW_TMO==1 ? "Yes" : "No" **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
awsize_s[2:0]	O	Secondary Port Burst size. This signal indicates the size of each transfer in the burst. Byte lane strobes indicate exactly which byte lanes to update. **Exists:** Always **Synchronous To:** aclk **Registered:** RS_AW_TMO==1 ? "Yes" : "No" **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
awburst_s[1:0]	O	Secondary Port Burst type for Write Channel. The burst type, coupled with the size information, details how the address for each transfer within the burst is calculated. **Exists:** Always **Synchronous To:** aclk **Registered:** RS_AW_TMO==1 ? "Yes" : "No" **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
awlock_s[(RS_LTW-1):0]	O	Secondary Port Lock type for Write Channel. This signal provides additional information about the atomic characteristics of the transfer. **Exists:** Always **Synchronous To:** aclk **Registered:** RS_AW_TMO==1 ? "Yes" : "No" **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
awcache_s[3:0]	O	Secondary Port Cache type for Write Channel. This signal indicates the bufferable, cacheable, write-through, write-back, and allocation attributes of the transaction. **Exists:** Always **Synchronous To:** aclk **Registered:** RS_AW_TMO==1 ? "Yes" : "No" **Power Domain:** SINGLE_DOMAIN **Active State:** N/A

Table 4-8 Slave Port Write Address Channel Signals (Continued)

Port Name	I/O	Description
awprot_s[2:0]	O	Secondary Port Protection type for Write Channel. This signal indicates the normal, privileged, or secure protection level of the transaction and whether the transaction is a data access or an instruction access. **Exists:** Always **Synchronous To:** aclk **Registered:** RS_AW_TMO==1 ? "Yes" : "No" **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
awvalid_s	O	Secondary Port Write address valid. This signal indicates that a valid write address and control information are available. ■ 0: Address and control information unavailable ■ 1: Address and control information available **Exists:** Always **Synchronous To:** aclk **Registered:** (RS_AW_TMO==1 || RS_AW_TMO==2)? "Yes" : "No" **Power Domain:** SINGLE_DOMAIN **Active State:** High
awqos_s[3:0]	O	Secondary Port Quality of Service identifier, conveying priority information associated with each transaction at the write address channel of the slave port. **Exists:** (RS_HAS_QOS == 1) **Synchronous To:** aclk **Registered:** RS_AW_TMO==1 ? "Yes" : "No" **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
awregion_s[3:0]	O	Secondary Port Write address Region identifier. Permits a single physical interface on a slave to be used for multiple logical interfaces. **Exists:** (RS_HAS_REGION == 1) **Synchronous To:** aclk **Registered:** RS_AW_TMO==1 ? "Yes" : "No" **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
awsnoop_s[2:0]	O	Secondary Port ACELITE signal. This signal indicates the transaction type for shareable write transactions. **Exists:** (RS_AXI_INTERFACE_TYPE == 2) **Synchronous To:** aclk **Registered:** RS_AW_TMO==1 ? "Yes" : "No" **Power Domain:** SINGLE_DOMAIN **Active State:** N/A

Table 4-8 Slave Port Write Address Channel Signals (Continued)

Port Name	I/O	Description
awdomain_s[1:0]	O	Secondary Port ACELITE signal. This signal indicates the shareability domain of a write transaction. **Exists:** (RS_AXI_INTERFACE_TYPE == 2) **Synchronous To:** aclk **Registered:** RS_AW_TMO==1 ? "Yes" : "No" **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
awbar_s[1:0]	O	Secondary Port ACELITE signal. This signal indicates a write barrier transaction. **Exists:** (RS_AXI_INTERFACE_TYPE == 2) **Synchronous To:** aclk **Registered:** RS_AW_TMO==1 ? "Yes" : "No" **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
awsideband_s[(RS_AW_SBW-1):0]	O	Secondary Port Sideband signal for write address channel. **Exists:** (RS_HAS_AWSB == 1&&RS_AXI_INTERFACE_TYPE==0) **Synchronous To:** aclk **Registered:** RS_AW_TMO==1 ? "Yes" : "No" **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
awuser_s[(RS_AW_SBW-1):0]	O	Secondary Port User signal for write address channel. **Exists:** (RS_HAS_AWSB == 1&&RS_AXI_INTERFACE_TYPE!=0) **Synchronous To:** aclk **Registered:** RS_AW_TMO==1 ? "Yes" : "No" **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
awready_s	I	Secondary Port Write address ready. This signal indicates that the slave is ready to accept an address and associated control signals. ■ 0: Slave not ready ■ 1: Slave ready **Exists:** Always **Synchronous To:** aclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** High

4.9 Slave Port Write Data Channel Signals

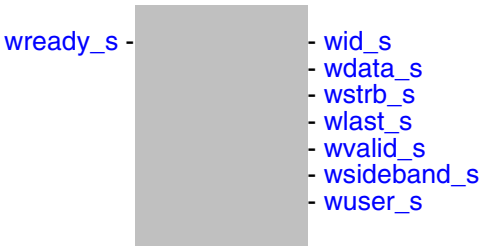


Table 4-9 Slave Port Write Data Channel Signals

Port Name	I/O	Description
wid_s[(RS_IDW-1):0]	O	Secondary Port Write ID tag. This signal is the ID tag of the write data transfer. The WID value must match the AWID value of the write transaction. **Exists:** (RS_AXI_INTERFACE_TYPE ==0) **Synchronous To:** aclk **Registered:** RS_W_TMO==1 ? "Yes" : "No" **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
wdata_s[(RS_DW-1):0]	O	Secondary Port Write data. **Exists:** Always **Synchronous To:** aclk **Registered:** RS_W_TMO==1 ? "Yes" : "No" **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
wstrb_s[((RS_DW/8)-1):0]	O	Secondary Port Write strobes. This signal indicates which byte lanes to update in memory. There is one write strobe for each eight bits of the write data bus. Therefore, wstrb_s corresponds to wdata_s[(8 x i)+7:(8 x i)]. **Exists:** Always **Synchronous To:** aclk **Registered:** RS_W_TMO==1 ? "Yes" : "No" **Power Domain:** SINGLE_DOMAIN **Active State:** High

Table 4-9 Slave Port Write Data Channel Signals (Continued)

Port Name	I/O	Description
wlast_s	O	Secondary Port Write last. This signal indicates the last transfer in a write burst. **Exists:** Always **Synchronous To:** aclk **Registered:** RS_W_TMO==1 ? "Yes" : "No" **Power Domain:** SINGLE_DOMAIN **Active State:** High
wvalid_s	O	Secondary Port Write valid. This signal indicates that valid write data and strobes are available. ■ 0: Write data and strobes unavailable ■ 1: Write data and strobes available **Exists:** Always **Synchronous To:** aclk **Registered:** (RS_W_TMO==1 || RS_W_TMO==2) ? "Yes" : "No" **Power Domain:** SINGLE_DOMAIN **Active State:** High
wsideband_s[(RS_W_SBW-1):0]	O	Optional. Secondary Port Sideband signal for write data channel. The signal name changes between the AXI3 and the AXI4 or ACE-Lite configurations. **Exists:** (RS_HAS_WSB == 1&&RS_AXI_INTERFACE_TYPE==0) **Synchronous To:** aclk **Registered:** RS_W_TMO==1 ? "Yes" : "No" **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
wuser_s[(RS_W_SBW-1):0]	O	Optional. Secondary Port User signal for write data channel. The signal name changes between the AXI3 and the AXI4 or ACE-Lite configurations. **Exists:** (RS_HAS_WSB == 1&&RS_AXI_INTERFACE_TYPE!=0) **Synchronous To:** aclk **Registered:** RS_W_TMO==1 ? "Yes" : "No" **Power Domain:** SINGLE_DOMAIN **Active State:** N/A

Table 4-9 Slave Port Write Data Channel Signals (Continued)

Port Name	I/O	Description
wready_s	I	Secondary Port Write ready. This signal indicates that the slave can accept the write data. ■ 0: Slave not ready ■ 1: Slave ready **Exists:** Always **Synchronous To:** aclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** High

4.10 Slave Port Read Data Channel Signals

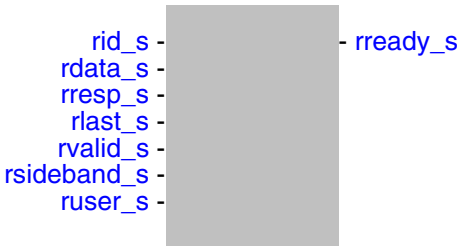


Table 4-10 Slave Port Read Data Channel Signals

Port Name	I/O	Description
rready_s	O	Secondary Port Read ready. This signal indicates that the slave can accept the read data and response information. ■ 0: Slave not ready ■ 1: Slave ready **Exists:** Always **Synchronous To:** aclk **Registered:** (RS_R_TMO==2 || RS_R_TMO==3) ? "Yes" : "No" **Power Domain:** SINGLE_DOMAIN **Active State:** High
rid_s[(RS_IDW-1):0]	I	Secondary Port Read ID tag. This signal is the ID tag of the read data group of signals. The RID value is generated by the slave and must match the ARID value of the read transaction to which it is responding. **Exists:** Always **Synchronous To:** aclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
rdata_s[(RS_DW-1):0]	I	Secondary Port Read data. **Exists:** Always **Synchronous To:** aclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A

Table 4-10 Slave Port Read Data Channel Signals (Continued)

Port Name	I/O	Description
rresp_s[1:0]	I	Secondary Port Read response. This signal indicates the status of the read transfer. The supported responses are OKAY, EXOKAY, SLVERR, and DECERR. **Exists:** Always **Synchronous To:** aclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
rlast_s	I	Secondary Port Read last. This signal indicates the last transfer in a read burst. **Exists:** Always **Synchronous To:** aclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** High
rvalid_s	I	Secondary Port Read valid. This signal indicates that the required read data is available and the read transfer can complete. ■ 0: Read data not available ■ 1: Read data available **Exists:** Always **Synchronous To:** aclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** High
rsideband_s[(RS_R_SBW-1):0]	I	Secondary Port Sideband signal for read data channel. **Exists:** (RS_HAS_RSB == 1&&RS_AXI_INTERFACE_TYPE==0) **Synchronous To:** aclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
ruser_s[(RS_R_SBW-1):0]	I	Secondary Port User signal for read data channel. **Exists:** (RS_HAS_RSB == 1&&RS_AXI_INTERFACE_TYPE!=0) **Synchronous To:** aclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A

4.11 Slave Port Write Response Channel Signals

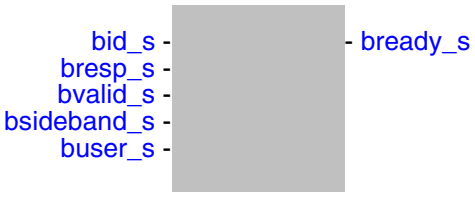


Table 4-11 Slave Port Write Response Channel Signals

Port Name	I/O	Description
bready_s	O	Secondary Port Response ready. This signal indicates that the slave can accept the response information. ■ 0: Slave not ready ■ 1: Slave ready **Exists:** Always **Synchronous To:** aclk **Registered:** (RS_B_TMO==2 || RS_B_TMO==3) ? "Yes" : "No" **Power Domain:** SINGLE_DOMAIN **Active State:** High
bid_s[(RS_IDW-1):0]	I	Secondary Port Response ID tag. This signal is the ID tag of the write response. The BID value must match the AWID value of the write transaction to which the slave is responding. **Exists:** Always **Synchronous To:** aclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
bresp_s[1:0]	I	Secondary Port Write response. This signal indicates the status of the write transaction. The supported responses are OKAY, EXOKAY, SLVERR, and DECERR. **Exists:** Always **Synchronous To:** aclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A

Table 4-11 Slave Port Write Response Channel Signals (Continued)

Port Name	I/O	Description
bvalid_s	I	Secondary Port Write response valid. This signal indicates that the valid write response data is available. ■ 0: Write response unavailable ■ 1: Write response available **Exists:** Always **Synchronous To:** aclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** High
bsideband_s[(RS_B_SBW-1):0]	I	Optional. Secondary Port Sideband signal for write response. The signal name changes between the AXI3 and AXI4/ACE-Lite configurations. ■ When the AXI3 interface is enabled, this signal is named bsideband_s. ■ When the AXI4/ACE-Lite interface is enabled, this signal is named buser_s. **Exists:** (RS_HAS_BSB == 1&&RS_AXI_INTERFACE_TYPE==0) **Synchronous To:** aclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
buser_s[(RS_B_SBW-1):0]	I	Optional. Secondary Port User signal for write response.The signal name changes between the AXI3 and AXI4/ACE-Lite configurations. ■ When the AXI3 interface is enabled, this signal is named bsideband_s. ■ When the AXI4/ACE-Lite interface is enabled, this signal is named buser_s. **Exists:** (RS_HAS_BSB == 1&&RS_AXI_INTERFACE_TYPE!=0) **Synchronous To:** aclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A

5 Internal Parameter Descriptions

Provides a description of the internal parameters that might be indirectly referenced in expressions in the Signals, Parameters, or Registers chapters. These parameters are not visible in the coreConsultant GUI and most of them are derived automatically from visible parameters. **You must not set any of these parameters directly.**

Some expressions might refer to TCL functions or procedures (sometimes identified as **function_of**) that coreConsultant uses to make calculations. The exact formula used by these TCL functions is not provided in this chapter. However, when you configure the core in coreConsultant, all TCL functions and parameters are evaluated completely; and the resulting values are displayed where appropriate in the coreConsultant GUI reports.

Table 5-1 Internal Parameters

Parameter Name	Equals To
RS_LTW	=(RS_AXI_INTERFACE_TYPE ! = 0 ? 1 : 2)

6 Verification

This chapter provides an overview of the testbench available for the DW_axi_rs verification. After DW_axi_rs has been configured and the verification setup, simulations can be run. For information on running simulations for DW_axi_rs in coreAssembler or coreConsultant, see the "Running the Simulation" section in the user guide.

Note DW_axi_rs verification testbench is built using Synopsys SVT Verification IP (VIP). Ensure that you have the supported version of the VIP components for this release to avoid tool compatibility problems. For more information about supported tools in this release, see the "Supported Versions of Tools and Libraries" section in the installation guide.

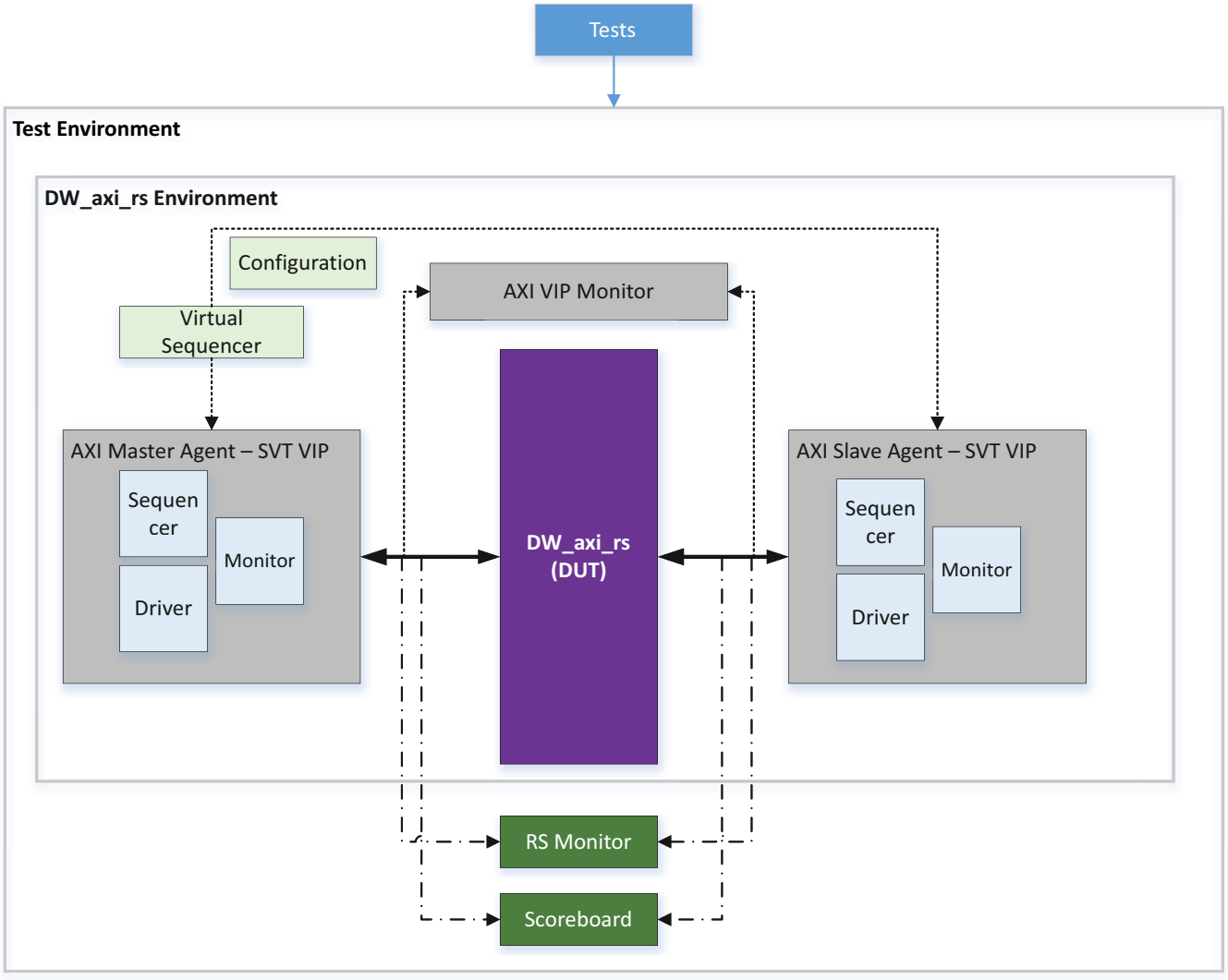
This chapter discusses the following sections:

6.1 Verification Environment

DW_axi_rs is verified using a UVM-methodology-based constrained random verification environment. The environment can generate random scenarios and the test case has hooks to control the scenarios to be generated.

Figure 6-1 shows the verification environment of the DW_axi_rs testbench:

Figure 6-1 DW_axi_rs UVM Verification Environment



The testbench consists of the following elements:

- Testbench uses the standard SVT VIP for the protocol interfaces:
  - AMBA SVT VIP
    - AXI VIP for the interface with AXI Master Data transfer interface
    - AXI Master VIP model is used to generate random sequences
    - AXI Slave VIP model is used to generate response for the incoming AXI transfers

6.2 Testbench Directories and Files

The DW_axi_rs verification environment contains the following directories and associated files.

Table 6-1 shows the various directories and associated files:

Table 6-1 DW_axi_rs Testbench Directory Structure

Directory	Description
<configured workspace>/sim/testbench	Top level testbench module (test_top.sv) and the DUT to the testbench wrapper (dut_sv_wrapper.sv) exist in this folder
<configured workspace>/sim/testbench/env	Contains testbench files. For example, scoreboard, sequences, VIP, environment, sequencers, and agents.
<configured workspace>/sim/	Primarily contains the supporting files to compile and run the simulation. After the completion of the simulation, the log files are present here
<configured workspace>/sim/test_*	Contains individual test cases. After the completion of the simulation, the test specific log files and if applicable the waveform files are stored here

6.3 Packaged Testcases

The simulation environment that comes as a package file includes some demonstrative tests. Some or all of the packaged demonstrative tests, depending upon their applicability to the chosen configuration are displayed in **Setup and Run Simulations > Testcases** in the coreConsultant GUI.

The associated shipped test cases and their description is explained in Table 6-2:

Table 6-2 DW_axi_rs Test Description

Test Name	Test Description
test_random	This test is aimed at generating random traffic, which is AXI compliant. The traffic is generated randomly based on DW_axi_rs configuration and VIP is auto-constrained to generate the traffic within the protocol limits. The traffic is generated in an outstanding fashion and sent towards DW_axi_rs. AXI Master Sequence from the VIP generates various possible AXI transfers for both Write and Read requests in parallel. The AXI transfer control attributes are randomized within the protocol and as per the DW_axi_rs requirement. The Primary port is monitored for the correctness of the AXI protocol across different IP configuration. AXI Slave Sequence from the VIP generates various possible AXI responses for the requests from secondary port of the DW_axi_rs. The response, delay, and other attributes are generated randomly. The secondary port is monitored for the correctness of the AXI protocol across different IP configuration.

7 Integration Considerations

After you have configured, tested, and synthesized your component with the coreTools flow, you can integrate the component into your own design environment.

7.1 Performance

This section discusses performance and the hardware configuration parameters that affect the performance of the DW_axi_rs.

7.1.1 Power Consumption, Frequency, Area and DFT Coverage

Table 7-1 provides information about the synthesis results (power consumption, frequency and area) and DFT coverage of the DW_axi_rs using the industry standard 16nm technology library.

Table 7-1 Synthesis Results for DW_axi_rs

Configuration	Operating Frequency	Gate Count	Power Consumption		Tetramax Coverage (%)		SpyGlass StuckAtCov (%)
			Dynamic	Static	StuckAt Test	Transition	
Default Configuration	aclk=400MHz	2004	0.120 mW	2 nW	100	100	100

Table 7-1 Synthesis Results for DW_axi_rs

Configuration	Operating Frequency	Gate Count	Power Consumption		Tetramax Coverage (%)		SpyGlass StuckAtCov (%)
			Dynamic	Static	StuckAt Test	Transition	
Typical Configuration 1 RS_AR_SBW = 64 RS_AR_TMO = 2 RS_AW = 64 RS_AW_SBW = 64 RS_AW_TMO = 2 RS_AXI_INTERFACE_TYPE = 2 RS_BLW = 8 RS_B_TMO = 2 RS_DW = 512 RS_HAS_ARSB = 1 RS_HAS_AWSB = 1 RS_HAS_BSB = 1 RS_HAS_QOS = 1 RS_HAS_REGION = 1 RS_HAS_RSB = 1 RS_HAS_WSB = 1 RS_IDW = 16 RS_R_SBW = 64 RS_R_TMO = 2 RS_W_SBW = 64 RS_W_TMO = 2	aclk=400MHz	36835	1.980 mW	30 nW	100	100	100
Typical Configuration 2 RS_AR_TMO = 3 RS_AW_TMO = 3 RS_B_TMO = 3 RS_R_TMO =3 RS_W_TMO = 3	aclk=400MHz	2576	0.133 mW	2 nW	100	99.85	100

Table 7-1 Synthesis Results for DW_axi_rs

Configuration	Operating Frequency	Gate Count	Power Consumption		Tetramax Coverage (%)		SpyGlass StuckAtCov (%)
			Dynamic	Static	StuckAt Test	Transition	
Typical Configuration 3 RS_AR_SBW = 64 RS_AR_TMO = 2 RS_AW = 64 RS_AW_SBW = 64 RS_AW_TMO = 2 RS_AXI_INTERFACE_TYPE = 1 RS_BLW = 8 RS_B_TMO = 2 RS_DW = 512 RS_HAS_ARSB = 1 RS_HAS_AWSB = 1 RS_HAS_BSB = 1 RS_HAS_QOS = 1 RS_HAS_REGION = 1 RS_HAS_RSB = 1 RS_HAS_WSB = 1 RS_IDW = 16 RS_R_SBW = 64 RS_R_TMO = 2 RS_W_SBW = 64 RS_W_TMO = 2	aclk=400MHz	36497	1.95 mW	30 nW	100	100	100

A
Glossary

active command queue	Command queue from which a model is currently taking commands; see also command queue.
application design	Overall chip-level design into which a subsystem or subsystems are integrated.
BFM	Bus-Functional Model — A simulation model used for early hardware debug. A BFM simulates the bus cycles of a device and models device pins, as well as certain on-chip functions. See also Full-Functional Model.
big-endian	Data format in which most significant byte comes first; normal order of bytes in a word.
blocked command stream	A command stream that is blocked due to a blocking command issued to that stream; see also command stream, blocking command, and non-blocking command.
blocking command	A command that prevents a testbench from advancing to next testbench statement until this command executes in model. Blocking commands typically return data to the testbench from the model.
command channel	Manages command streams. Models with multiple command channels execute command streams independently of each other to provide full-duplex mode function.
command stream	The communication channel between the testbench and the model.
component	A generic term that can refer to any synthesizable IP or verification IP in the DesignWare Library. In the context of synthesizable IP, this is a configurable block that can be instantiated as a single entity (VHDL) or module (Verilog) in a design.
configuration	The act of specifying parameters for a core prior to synthesis; can also be used in the context of VIP.
configuration intent	Range of values allowed for each parameter associated with a reusable core.
cycle command	A command that executes and causes HDL simulation time to advance.

decoder	Software or hardware subsystem that translates from and “encoded” format back to standard format.
design context	Aspects of a component or subsystem target environment that affect the synthesis of the component or subsystem.
design creation	The process of capturing a design as parameterized RTL.
DesignWare Library	A collection of synthesizable IP and verification IP components that is authorized by a single DesignWare license. Products include SmartModels, VMT model suites, DesignWare Memory Models, Building Block IP, and the DesignWareSynthesizable Components for AMBA.
dual role device	Device having the capabilities of function and host (limited).
endian	Ordering of bytes in a multi-byte word; see also little-endian and big-endian.
Full-Functional Mode	A simulation model that describes the complete range of device behavior, including code execution. See also BFM.
GPIO	General Purpose Input Output.
GTECH	A generic technology view used for RTL simulation of encrypted source code by non-Synopsys simulators.
hard IP	Non-synthesizable implementation IP.
HDL	Hardware Description Language – examples include Verilog and VHDL.
IIP	Implementation Intellectual Property — A generic term for synthesizable HDL and non-synthesizable “hard” IP in all of its forms (coreKit, component, core, MacroCell, and so on).
implementation view	The RTL for a core. You can simulate, synthesize, and implement this view of a core in a real chip.
instantiate	The act of placing a core or model into a design.
interface	Set of ports and parameters that defines a connection point to a component.
IP	Intellectual property — A term that encompasses simulation models and synthesizable blocks of HDL code.
little-endian	Data format in which the least-significant byte comes first.
master	Device or model that initiates and controls another device or peripheral.
model	A Verification IP component or a Design View of a core.
monitor	A device or model that gathers performance statistics of a system.
non-blocking command	A testbench command that advances to the next testbench statement without waiting for the command to complete.

peripheral	Generally refers to a small core that has a bus connection, specifically an APB interface.
RTL	Register Transfer Level. A higher level of abstraction that implies a certain gate-level structure. Synthesis of RTL code yields a gate-level design.
SDRAM	Synchronous Dynamic Random Access Memory; high-speed DRAM adds a separate clock signal to control signals.
SDRAM controller	A memory controller with specific connections for SDRAMs.
slave	Device or model that is controlled by and responds to a master.
SoC	System on a chip.
soft IP	Any implementation IP that is configurable. Generally referred to as synthesizable IP.
sparse data	Data bus data which is individually byte-enabled. The AXI bus allows sparse data transfers using the WSTRB signal, each byte lane individually enabled. The AHB bus does not allow sparse data transfers.
static controller	Memory controller with specific connections for Static memories such as asynchronous SRAMs, Flash memory, and ROMs.
synthesis intent	Attributes that a core developer applies to a top-level design, ports, and core.
synthesizable IP	A type of Implementation IP that can be mapped to a target technology through synthesis. Sometimes referred to as Soft IP.
technology-independent	Design that allows the technology (that is, the library that implements the gate and via widths for gates) to be specified later during synthesis.
Testsuite Regression Environment (TRE)	A collection of files for stand-alone verification of the configured component. The files, tests, and functionality vary from component to component.
VIP	Verification Intellectual Property — A generic term for a simulation model in any form, including a Design View.
wrap, wrapper	Code, usually VHDL or Verilog, that surrounds a design or model, allowing easier interfacing. Usually requires an extra, sometimes automated, step to create the wrapper.
zero-cycle command	A command that executes without HDL simulation time advancing.

Index