SYNOPSYS®

DesignWare® DW_axi_gs

Databook

DW_axi_gs – 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 5

Preface 7
Organization 7
Related Documentation 7
Web Resources 8
Customer Support 8
Product Code 9

Chapter 1
Product Overview 11
1.1 DesignWare Synthesizable Components for AMBA System Overview 11
1.2 General Product Description 13
1.2.1 DW_axi_gs Block Diagram 13
1.3 Features 14
1.4 Standards Compliance 15
1.5 Verification Environment Overview 15
1.6 Where To Go From Here 15

Chapter 2
Functional Description 17
2.1 Overview 17
2.2 GIF Request Channel 17
2.3 GIF Response Channel 19
2.4 GIF Transaction Examples 20
2.5 Microarchitecture 27
2.6 Extended GIF Mode 28
2.7 GIF Lite Mode 28
2.8 Direct-Ready Feedthroughs 29
2.9 Outstanding Transaction Control 29
2.10 Synchronous Generic Clock 30
2.11 Low-Power Interface 30
2.11.1 cactive Signal De-assertion 32
2.11.2 cactive Signal Assertion 32
2.12 AXI4 Optional Signaling Interface 34
2.13 Exclusive Access Monitor 36
2.14 Performance 37

Chapter 3
Parameter Descriptions 39

3.1 Toplevel Parameters . . . 40
3.2 Performance Parameters . . . 42
3.3 Sideband/User Signals Parameters . . . 44
3.4 Low Power Interface Configuration Parameters . . . 46

Chapter 4
Signal Descriptions . . . 47
4.1 Clock and Resets Signals . . . 49
4.2 AXI Write Address Channel Signals . . . 50
4.3 AXI Write Data Channel Signals . . . 54
4.4 AXI Write Response Channel Signals . . . 57
4.5 AXI Read Address Channel Signals . . . 59
4.6 AXI Read Data Channel Signals . . . 63
4.7 GIF Request Channel Signals . . . 66
4.8 GIF Response Channel Signals . . . 70
4.9 AXI Low Power Interface Signals . . . 72

Chapter 5
Internal Parameter Descriptions . . . 73

Chapter 6
Verification . . . 75
6.1 Overview of Vera Tests . . . 75
6.1.1 test_random . . . 75
6.1.2 test_exclusive . . . 75
6.1.3 test_extd_gif . . . 76
6.2 Overview of DW_axi_gs Testbench . . . 76

Chapter 7
Integration Considerations . . . 79
7.1 Performance . . . 79
7.1.1 Power Consumption, Frequency, Area and DFT Coverage . . . 79

Appendix 8
Basic Core Module (BCM) Library . . . 81
8.1 BCM Library Components . . . 81

Appendix A
Glossary . . . 83

Index . . . 87

Revision History

This section tracks the significant documentation changes that occur from release-to-release and during a release from version 1.03b onward.

Date	Release	Description
2.04a	March 2020	■ Revision version change for 2020.03a release ■ Added Appendix 8, “Basic Core Module (BCM) Library” ■ Updated synthesis results in “Performance” on page 79 ■ “Signal Descriptions” on page 47 and “Parameter Descriptions” on page 39 “Internal Parameter Descriptions” on page 73, auto-extracted from the RTL
2.03a	February 2018	■ Revision version change for 2018.02a release ■ Updated synthesis results in “Performance” on page 79 ■ 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 47 and “Parameter Descriptions” on page 39 “Internal Parameter Descriptions” on page 73, 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 47 and “Parameter Descriptions” on page 39 auto-extracted from the RTL ■ Added the chapter “Internal Parameter Descriptions” on page 73 ■ Updated area and power numbers in “Performance” on page 79
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 AMBA 4 AXI specifications.
1.11a	May 2013	Added sideband signals and mid, sid, and slast signals. Added coreConsultant parameters for sideband signals. Added a section to describe the “Extended GIF Mode”. Corrected the width of the awlen, arlen, and mlen signals. Corrected the parameter names for AXI and GIF address width, data width, and ID width. Added a section on the Extended GIF Mode. Updated the template.

Date	Release	Description
1.10a	Oct 2012	Added the product code on the cover and in Table 1-1.
1.10a	Jun 2012	Version change for 2012.06b release.
1.09a	Oct 2011	Corrected gclken input delay.
1.07a	Jun 2011	Updated system diagram in Figure 1-1; enhanced “Related Documents” section in Preface.
1.07a	Jan 2011	Version change for 2010.12a release.
1.06a	Sep 2010	Corrected names of include files and vcs command used for simulation
1.05a	Dec 2009	Corrected first step of 3-beat read burst in “GIF Transaction Examples”; updated databook to new template for consistency with other IIP/VIP/PHY databooks.
1.05a	May 2009	Removed references to QuickStarts, as they are no longer supported.
1.05a	Jan 2009	Version change for 2009.01a release.
1.04a	Oct 2008	Version change for 2008.10a release.
1.03c	Jun 2008	Version change for 2008.06a release.
1.03b	Jul 2007	Updated parameter names that changed from GSX* to GS*.

Preface

This databook provides information that you need to interface the DW_axi_gs component to the Synopsys DesignWare Synthesizable Components for AMBA environment. 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, and signal and parameter descriptions, as well as information on how you can configure, create RTL for, simulate, and synthesize the component using coreConsultant. Also provided are 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_gs.
- Chapter 3, "Parameter Descriptions" identifies the configurable parameters supported by the DW_axi_gs.
- Chapter 4, "Signal Descriptions" provides a list and description of the DW_axi_gs 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_gs.
- Chapter 7, "Integration Considerations"includes information you need to integrate the configured DW_axi_gs 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, 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

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_gs is a configurable module between a generic interface (GIF) and the AMBA AXI bus. The DW_axi_gs component has both an AMBA 3 AXI/AMBA 4 AXI-compliant slave interface and a simplified GIF. It is part of the family of DesignWare Synthesizable Components for AMBA.

The DW_axi_gs AXI slave interface connects to an AXI bus — for example, the DW_axi Interconnect for AMBA AXI interfaces. The DW_axi_gs GIF connects to a third-party component that receives transactions from the DW_axi_gs.

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

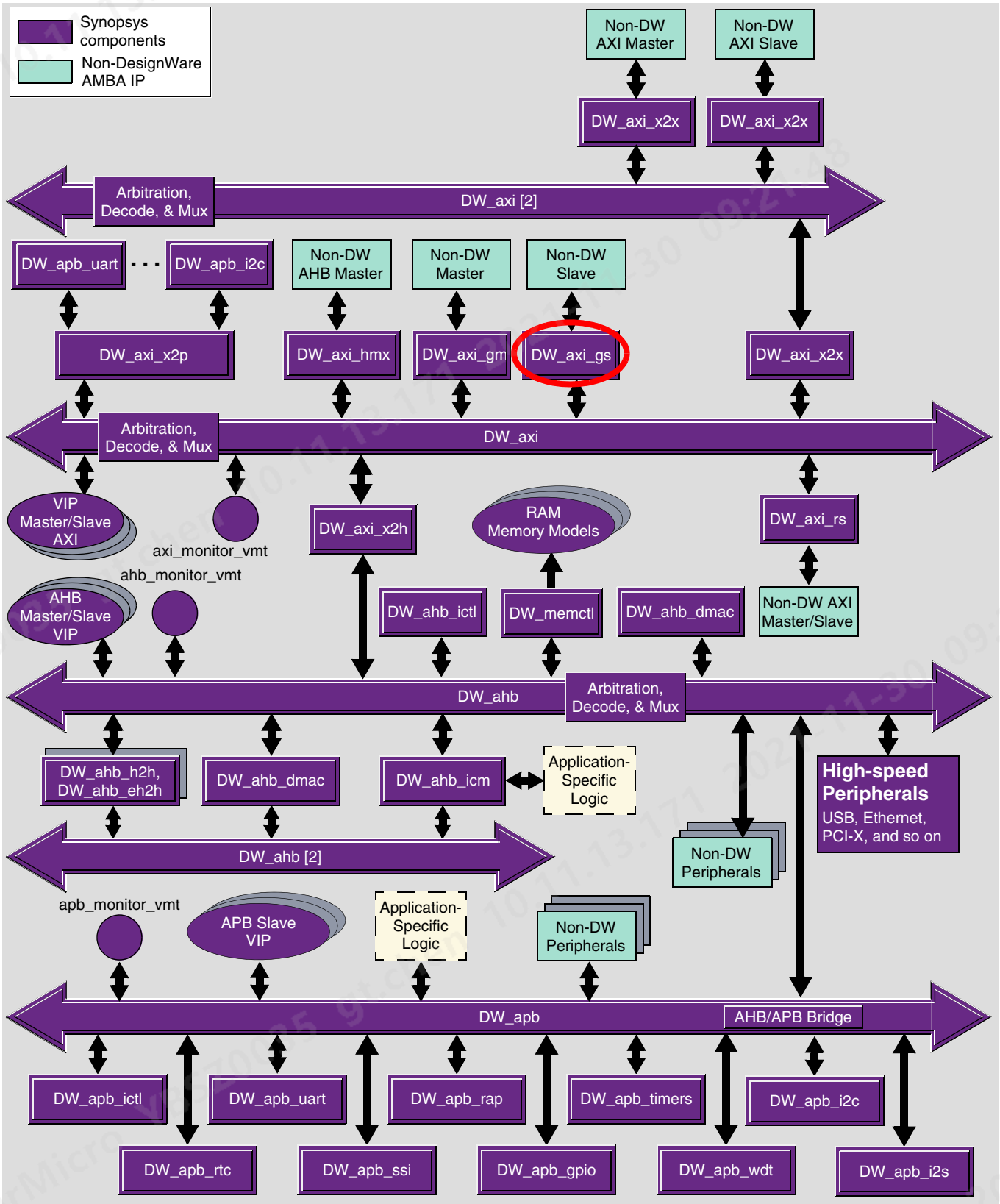
1.1 DesignWare Synthesizable Components for AMBA System Overview

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, and AMBA 3 AXI/AMBA 4 AXI (Advanced eXtensible Interface) 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_gs in a Complete System



You can connect, configure, synthesize, and verify the DW_axi_gs within a DesignWare subsystem using coreAssembler, documentation for which is available on the web in the *coreAssembler User Guide*.

If you want to configure, synthesize, and verify a single component such as the DW_axi_gs component, you might prefer to use coreConsultant, documentation for which is available in the *coreConsultant User Guide*.

1.2 General Product Description

The DesignWare DW_axi_gs AMBA AXI slave-to-generic-interface (GIF) provides a method to transfer AXI-generated transactions to a more basic GIF. It provides the subsystem designer with an easy way to reuse existing custom or third-party components in new designs that use the high-performance AMBA AXI bus. The DW_axi_gs uses a simplified interface, reducing the complexity of design required to port a custom or third-party component to the AXI bus.

Functionally, the DW_axi_gs AXI slave interface receives transactions from the AXI bus. Internal to the DW_axi_gs, these AXI transactions are translated into GIF transactions. The DW_axi_gs GIF initiates the translated AXI transactions to the external custom/third-party component.

The GIF consists of:

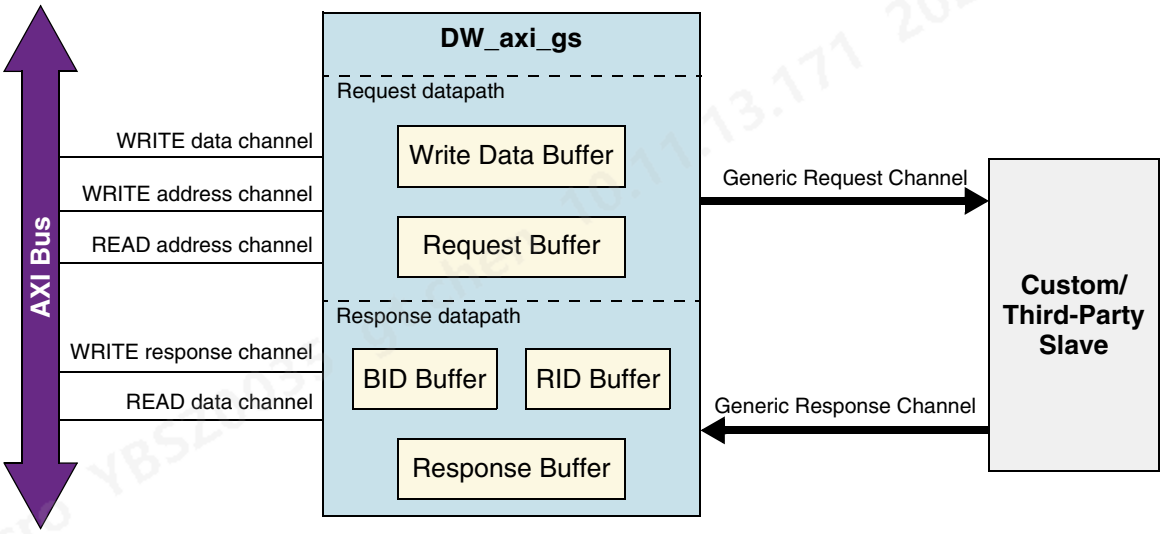
- A combined read/write generic request channel that initiates new transactions to the custom or third-party component
- A combined read/write generic response channel that receives read data and write responses

The DW_axi_gs is able to pipeline multiple AXI transactions. Internal to the DW_axi_gs are user-configurable buffers, which allow channel transaction queuing if the destination channel ever stalls transactions.

1.2.1 DW_axi_gs Block Diagram

Figure 1-2 shows a block diagram of a DW_axi_gs peripheral.

Figure 1-2 DW_axi_gs Block Diagram



As illustrated in Figure 1-2, the DW_axi_gs is partitioned into five buffers that communicate with the AXI bus through the five AXI channels and the GIF through the two generic channels.

- Write Data buffer – Buffers write data from the AXI bus.
- Request buffer – Shared for read and write requests from the AXI bus.
- BID buffer – Buffers AWIDs from the AXI bus to label write responses to the AXI.
- RID buffer – Buffers ARIDs from the AXI bus to label read responses to the AXI.
- Response buffer – Shared for read responses, read data, and write responses from the GIF.

1.3 Features

The DW_axi_gs supports the following features:

- Complies with the following specifications:
  - AMBA 3 AXI
  - AMBA 4 AXI
- Full support of AXI low-power interface
  - Transaction activity status to an external low-power controller (LPC) for enabling or disabling the clock.
  - Configurable maximum number of clock cycles the system must remain idle before entering low-power mode.
- Two-way flow control
- Synchronous point-to-point communication on generic interface (GIF)
- Independent request and response GIF channels
- Configurable number of outstanding transactions
- Extended GIF mode with transaction ID
- Sideband/user signals on all channels
- Synchronous clock support; including slower GIF clock
- Configurable microarchitecture
- Configurable I/O signals to support simple peripherals (such as, SRAM)

For details on these features, see "Functional Description" on page 17.

The DW_axi_gs has the following known limitations:

- No support for data reordering
- No support for write data interleaving
- No support for awcache/awprot or arcache/arprot
- Sideband/user signal width verified up to 64 bits only

When the Extended GIF mode is enabled, the DW_axi_gs has the following known limitations:

- No support for exclusive access.

- No support for out-of-order responses for writes on the GIF.
- No support for read data interleaving.
- No support for combinatorial GIF Ready (GS_DIRECT_GIF_READY) mode.
- No support for combinatorial AXI Ready (GS_DIRECT_AXI_READY) mode.
- Limited write transaction verification. Write transaction ID on the GIF is verified for write ID consistency between the AXI and the GIF without write interleaving or out-of-order write responses.

1.4 Standards Compliance

The DW_axi_gs conforms to the AMBA 3 AXI and AMBA 4 AXI specifications defined in the *AMBA AXI and ACE Protocol Specification* from ARM.

1.5 Verification Environment Overview

The DW_axi_gs 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" section describes the DW_axi_gs testbench environment.

1.6 Where To Go From Here

At this point, you may want to get started working with the DW_axi_gs 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_gs component, see the *"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_gs component within a DesignWare subsystem using coreAssembler, see the *"Overview of the coreAssembler Configuration and Integration Process"* section in *DesignWare Synthesizable Components for AMBA 2 User Guide*.

2 Functional Description

This chapter describes the DW_axi_gs, including the Generic Interface (GIF) and overall functionality.

2.1 Overview

The Generic Interface Module for the AMBA AXI slave—DW_axi_gs—simplifies the connection of custom or third-party slave components to the AMBA AXI bus using a Generic Interface (GIF). The GIF consists of two independent channels for requests and responses. Each channel is a point-to-point connection from a single source to a single sink. The channels offer two-way flow control similar to the valid/ready handshake on the AXI bus.

By default, when the Extended GIF mode is not enabled, the DW_axi_gs forces all transactions to complete in order with no data interleaving. Therefore, the GIF slave component must also send in-order responses.

In the Extended GIF mode, the DW_axi_gs supports out-of-order responses for read transactions and in-order write transaction response. Data interleaving is not supported. Further, the DW_axi_gs cannot be configured to posted read and write transactions.

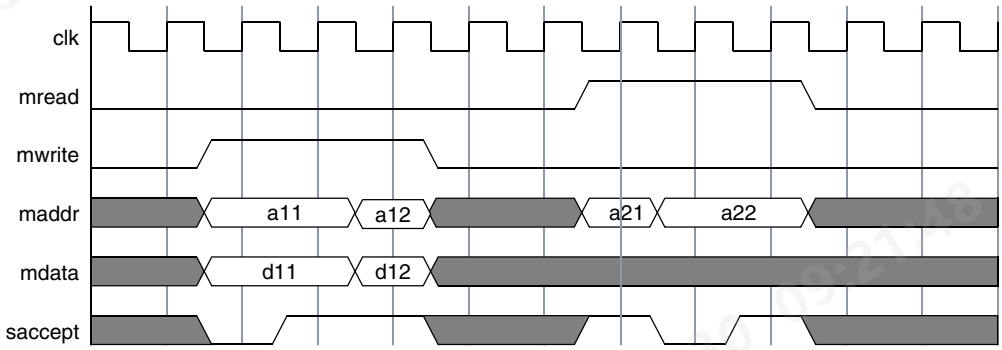
As the DW_axi_gs accepts new transactions from the AXI, it transfers them to the GIF request channel and can be configured to buffer AXI transactions if the GIF is unable to immediately accept a transaction. Similarly, read data and write responses from the GIF response channel can be configured to be buffered if the AXI is unable to immediately accept a response.

All channels on the DW_axi_gs support sideband/user signal propagation from the AXI to the GIF and vice versa.

2.2 GIF Request Channel

Figure 2-1 illustrates a timing diagram for a two-beat write and a two-beat read to the GIF request channel.

Figure 2-1 Write and Read Request



A write request from the DW_axi_gs on the GIF request channel starts when the mwrite signal is asserted high; similarly, a read request starts when the mread signal is asserted high. The mwrite and mread signals are not be driven high at the same time. Once a valid access starts, a high on the saccept signal indicates that the slave accepts the current request cycle. If a low occurs on the saccept signal, the DW_axi_gs must extend the current request cycle.

Single transactions have only one request cycle. Write and read bursts have multiple cycles, referred to as beats. Each beat must be accepted separately by the saccept signal, which must be sampled high *N* times for a burst that has *N* beats. The address in a burst changes from beat to beat according to the burst type. The most common scenario is an incrementing address. Like the INCR burst type in the AXI protocol, address increments depend on transaction size. The GIF burst types match the AXI specification, and the DW_axi_gs also supports WRAP and FIXED burst types.

As seen in Figure 2-1, the saccept signal can be low or high on the first beat. If the slave cannot latch the address phase, then it must drive saccept low to ensure that the same address is still driven by the DW_axi_gs. If the slave can latch at least one address phase, it can drive saccept high on the first cycle, and if necessary, drive saccept low on subsequent cycles.

Figure 2-1 does not show all the control signals. Additional control signals are mlast, msize, mlen, and mburst. All other control signals are identical to their counterparts on the AXI read and write request channels.

In the Extended GIF mode, the DW_axi_gs supports ID signal propagation from the AXI read and write interface to the GIF. The master ID (mid) signal on the GIF request channel passes the arid or awid information from the DW_axi_gs for every read or write transaction respectively. For a given ID, the ID responses must follow the order of requests. However, the DW_axi_gs supports out-of-order responses between different IDs.

The DW_axi_gs also supports sideband/user signals, with a maximum width of 256 bits on each channel. The masideband signal on the GIF request channel passes information from the AXI write address or AXI read address sideband/user signal and the mwsideband signal passes information from the AXI write data sideband/user signal.

Attention

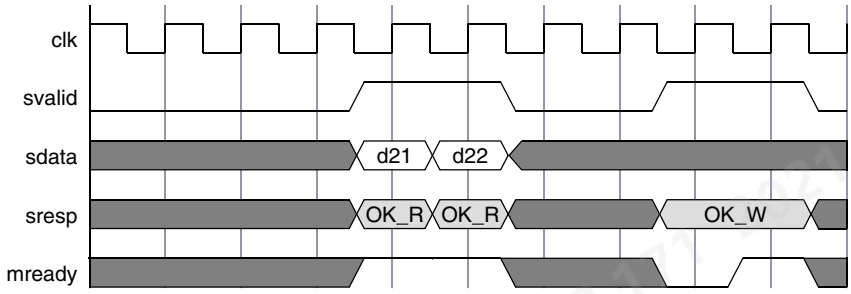
If you use both the DW_axi_gm and DW_axi_gs components, you should understand that the DW_axi_gm differs in the definition of the request channel:

- GIF reads are handled differently by the DW_axi_gs and DW_axi_gm. The DW_axi_gs initiates multi-beat GIF read bursts to the external GIF slave, generating addresses for each beat of the burst. In contrast, the DW_axi_gm only supports a single cycle GIF read burst request from the external GIF master; therefore only a single address is used. This simplifies address generation logic in peripherals connected to the DW_axi_gs and DW_axi_gm GIF.
- GIF writes are handled slightly differently by the DW_axi_gs and DW_axi_gm. The DW_axi_gs initiates multi-beat GIF write bursts to the external GIF slave, generating addresses for each beat of the burst. Similarly, the DW_axi_gm supports a multi-beat GIF write burst request from the external GIF master, but it only needs the address for the first beat of the burst; subsequent beat addresses are not necessary and are ignored. This simplifies address generation logic in peripherals connected to the DW_axi_gs and DW_axi_gm GIF.
- The mlock signal does not exist on the DW_axi_gs.
- The mlast signal does not exist on DW_axi_gm, which keeps track of the write beats internally.

2.3 GIF Response Channel

Figure 2-2 illustrates a timing diagram for a 2-beat read and write to the GIF response channel.

Figure 2-2 Read and Write Response



The svalid signal indicates a valid response, which can be comprised of data and the response information for a read, or of just the response information for a write. If the DW_axi_gs is not ready to accept the response, it can drive mready low, at which point the slave must extend the current cycle.

A read response contains the actual data that were read, along with the response information; both are asserted in the same cycle. In contrast, a write response contains only the response information. Additionally, there is always one response for every beat in a burst of a read transaction, whereas there is only one response for a complete burst in a write transaction.

All GIF read responses must be returned to the DW_axi_gs in the same order as the read requests. All GIF write responses must be returned in the same order as the write requests, as well. However, there are no restrictions on the response order between reads and writes, because the response signal has a separate encoding for read and write responses. This encoding is different from the encoding used on the AXI bus.

The srsideband signal on the GIF response channel passes information to the AXI read data sideband/user signal or the AXI write response sideband/user signal.

Attention If you use both the DW_axi_gm and DW_axi_gs components, you should know there is a difference in the encoding of the response signal. For the DW_axi_gm, the sresp output signal distinguishes between SLVERR and DECERR and is three bits wide, whereas the DW_axi_gs understands only SLVERR and has a 2-bit wide sresp input signal.For the sresp output signal encoding details, see "sresp[1:0]" on page 71.

2.4 GIF Transaction Examples

Figure 2-3 illustrates a timing diagram for a read transaction in a single beat, with the Extended GIF mode enabled. Note the following:

1. The arsideband/aruser signal is asserted along with the arvalid signal by the AXI master on the AXI interface. It is qualified by the arready signal from DW_axi_gs on the AXI interface. The arsideband/aruser signal is synchronous to the AXI clock (clk).
2. The mid and masideband signals are asserted along with the mread signal by DW_axi_gs on the GIF. Both signals are qualified by the saccept signal on the GIF. Similarly, the sid and srsideband signals are asserted along with the svalid signal on the GIF. Both signals are qualified by the mready signal from DW_axi_gs on the GIF.
3. When the last beat of read-data is transferred on the GIF, the slast signal is asserted along with the svalid signal. It is qualified by the mready signal on the AXI master. The slast signal is synchronous to the AXI clock (clk).
4. The rsideband/ruser signal is asserted along with the rvalid signal by DW_axi_gs on the AXI interface. It is qualified by the rready signal on the AXI master. The rsideband/ruser signal is synchronous to the AXI clock (clk).
5. The value of the arsideband/aruser signal is transferred to the masideband signal. On the GIF, the masideband signal is asserted along with the mread signal. The sideband/user signals are routed internally in the same manner as the other control signals. On the AXI interface, the sideband/user signals are asserted along with the arvalid signal. Similarly, the rsideband/ruser signal on the AXI interface is transferred from the srsideband signal on the GIF along with the assertion of the svalid signal.

Figure 2-3 Read Transaction of a Single Beat in the Extended GIF Mode

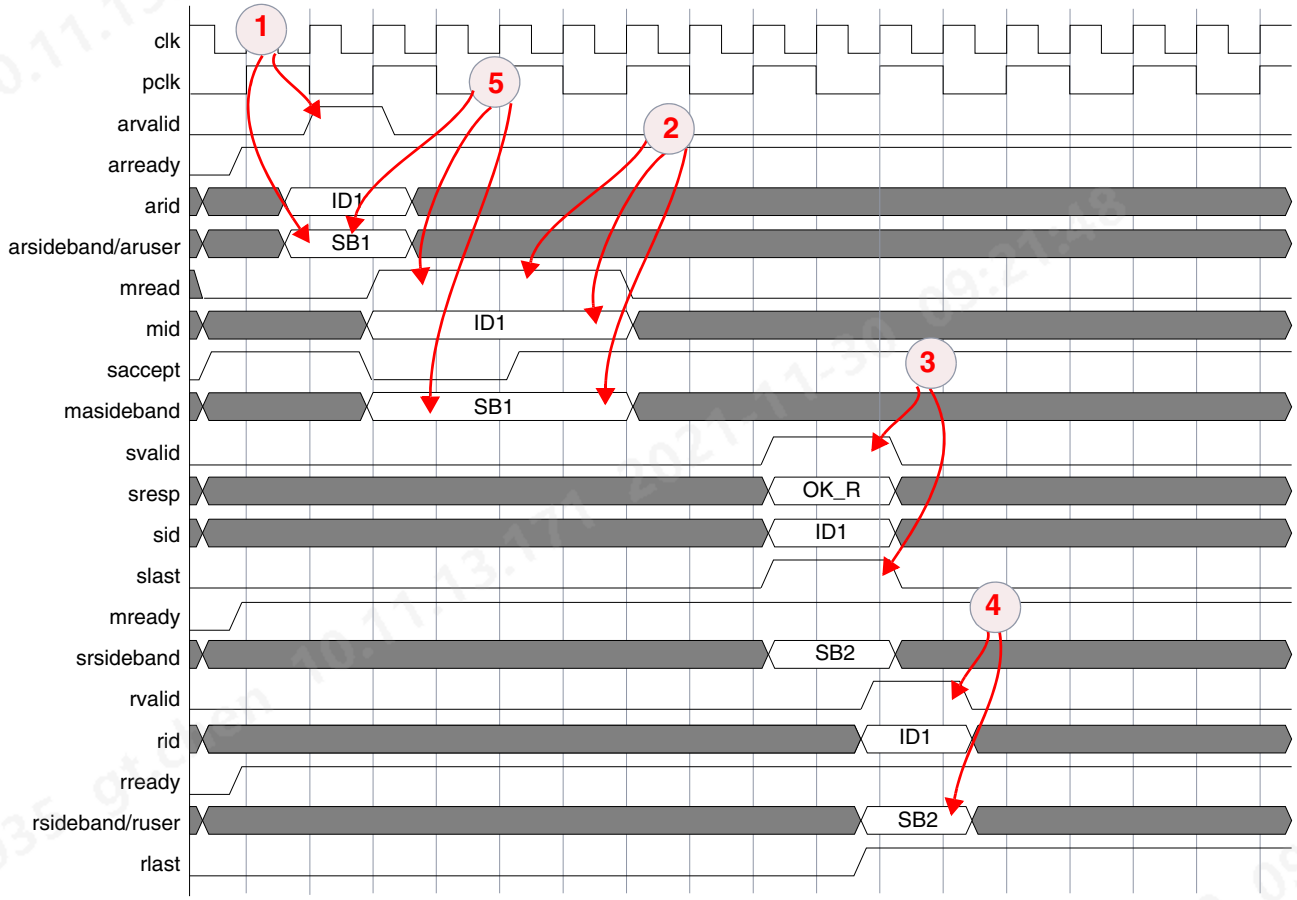
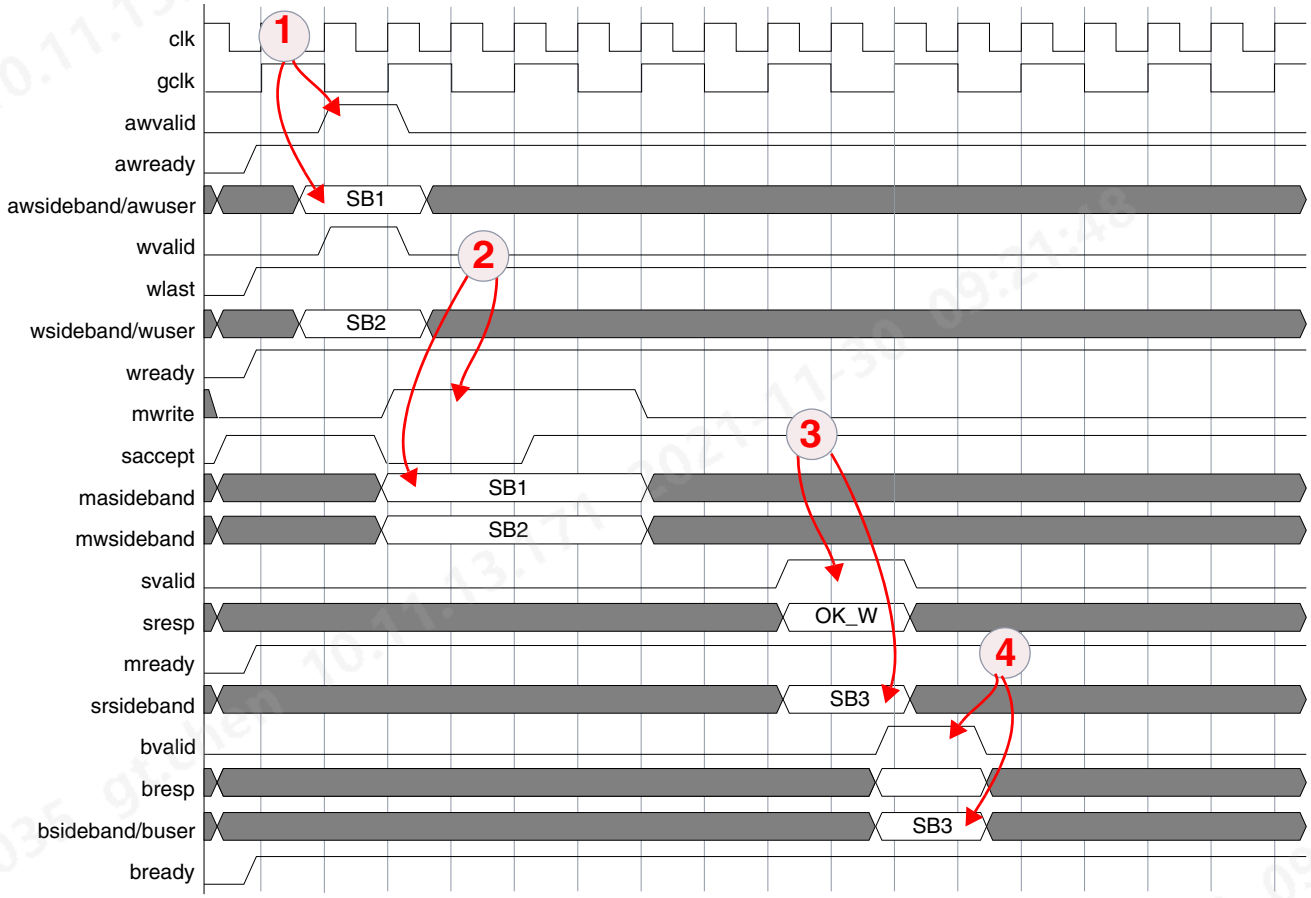


Figure 2-4 illustrates a timing diagram for a write transaction in a single beat, with the Extended GIF mode enabled.

Figure 2-4 Write Transaction of a Single Beat in the Extended GIF Mode



Note the following in Figure 2-4:

1. The awsideband/awuser signal is asserted along with the awvalid signal from the AXI master on the AXI interface. It is qualified by the awready signal from DW_axi_gs on the AXI interface. The awsideband/awuser signal is synchronous to the AXI clock (clk).
2. The masideband signal is asserted along with the mwrite signal by DW_axi_gs on the GIF. It is qualified by the saccept signal on the GIF.
3. The srsideband signal is asserted along with the svalid signal on the GIF. This signal is qualified by the mready signal from DW_axi_gs on the GIF.
4. The bsideband/buser signal is asserted along with the bvalid signal by DW_axi_gs on the AXI interface. It is qualified by bready from the AXI master. The bsideband/buser signal is synchronous to the AXI clock (clk).
5. The mid and sid signals are not verified for write transactions.

Figure 2-5 shows a sequence of read and write transactions for the GIF, as initiated by the DW_axi_gs. The sequence consists of a 3-beat read, 2-beat read, and a back-to-back sequence of 2-beat write, 2-beat write and 2-beat read. Note the following behaviors in Figure 2-5:

1. Start a 3-beat read burst; for beat 2 in this example, the GIF slave connected to DW_axi_gs is not ready and drives saccept low for one cycle

2. Second write response arrives after read responses, which is okay

Figure 2-5 DW_axi_gs Read and Write Bursts

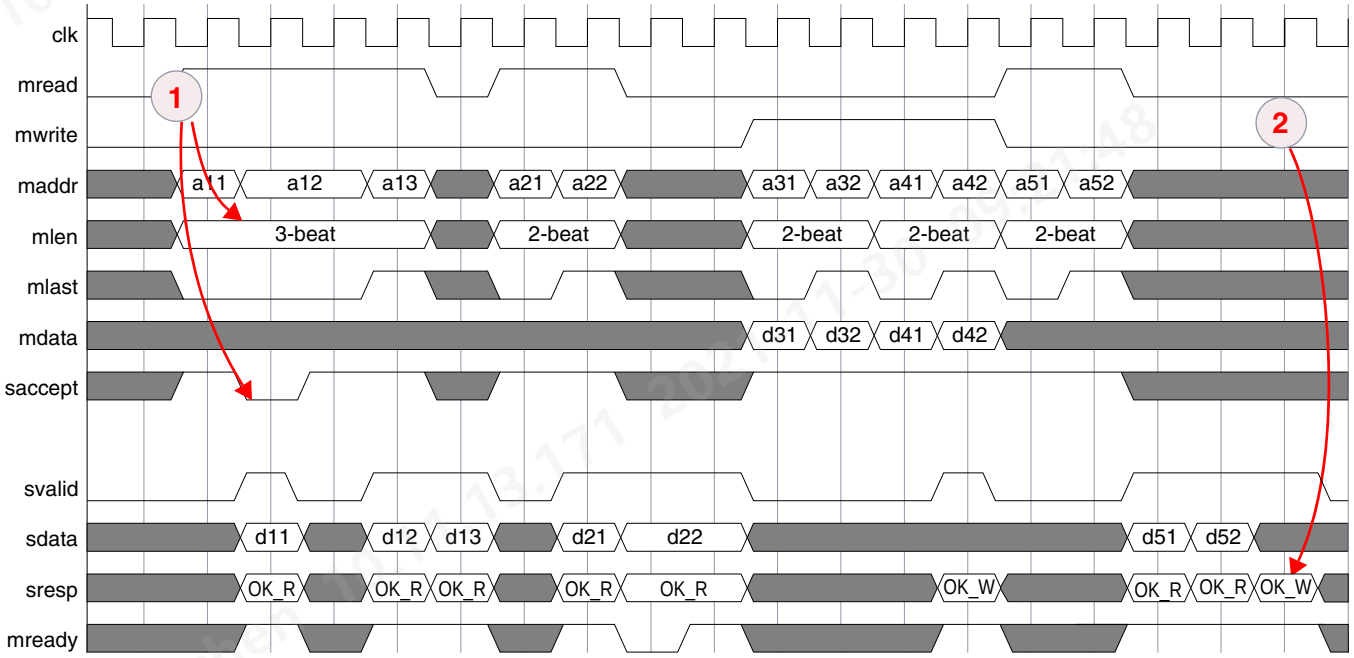


Figure 2-6 illustrates a one-beat write request. Note the following:

1. wready asserted from DW_axi_gs; GIF write data and strobe
2. awready asserted from DW_axi_gs; GIF write indicator and address
3. GIF command accept for slave
4. svalid asserted to DW_axi_gs; AXI write response

Figure 2-6 1-Beat Write GIF Request

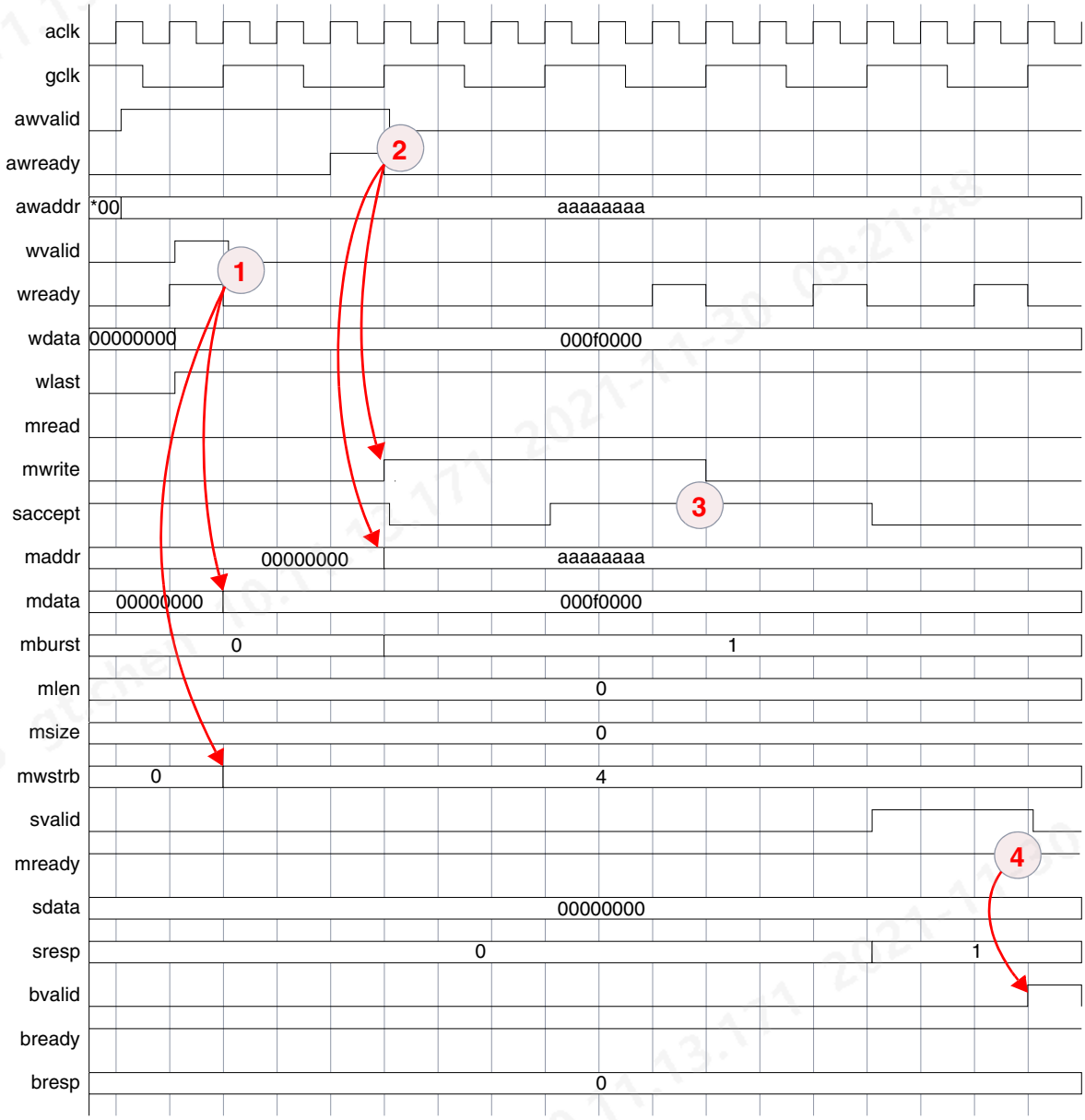


Figure 2-7 illustrates a one-beat read request. Note the following:

1. aready asserted from DW_axi_gs; GIF read indicator and address
2. mread de-asserted from DW_axi_gs
3. svalid de-asserted to DW_axi_gs; AXI read response

Figure 2-7 1-Beat Read GIF Request

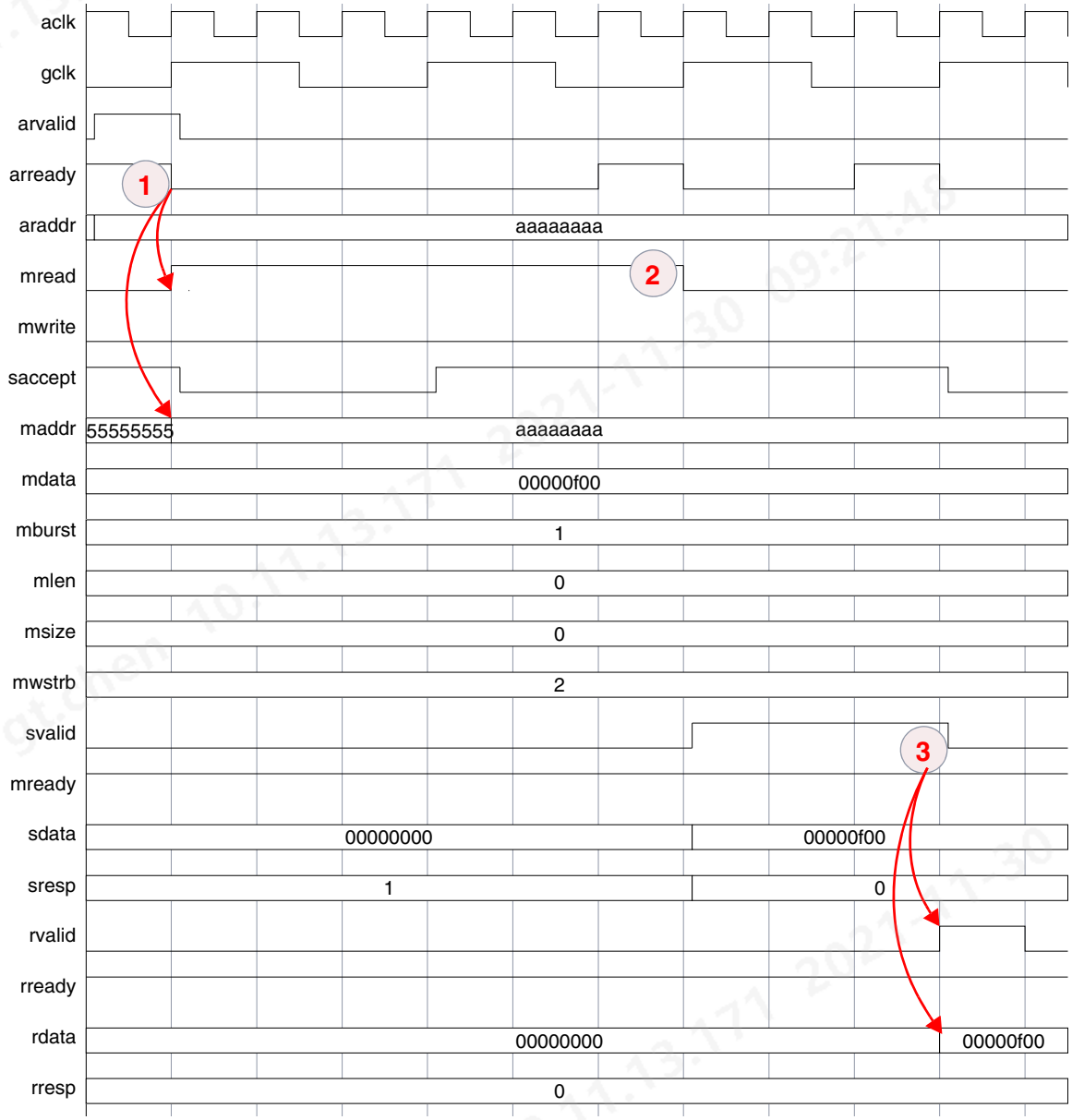
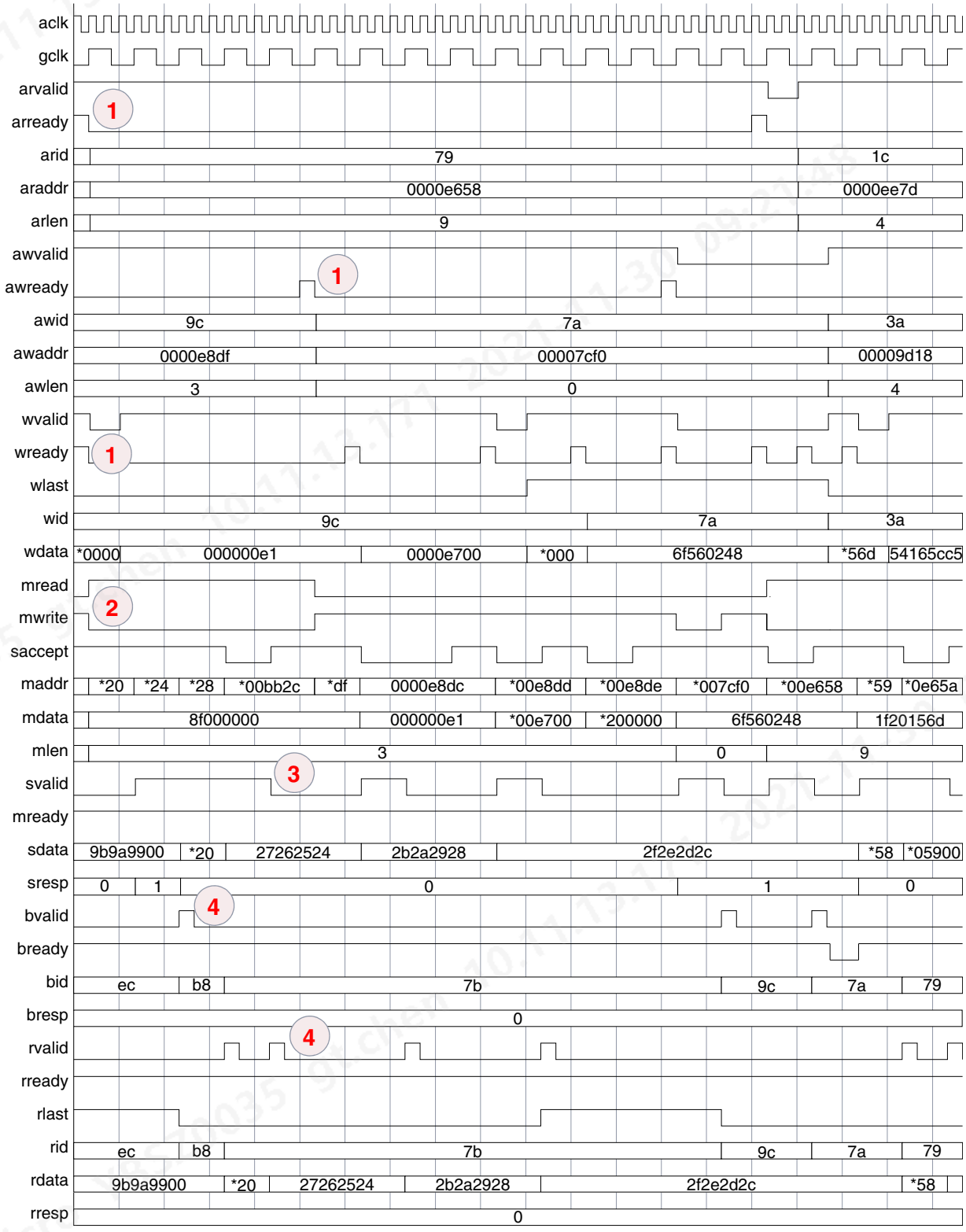


Figure 2-8 illustrates a multiple-beat read and multiple-beat write GIF request. Note the following:

1. arready/awready/wready assertions from DW_axi_gs
2. mread/mwrite assertions on the GIF read and write indicators
3. svalid assertions on the GIF response channel
4. bvalid/rvalid assertions on the AXI response channels

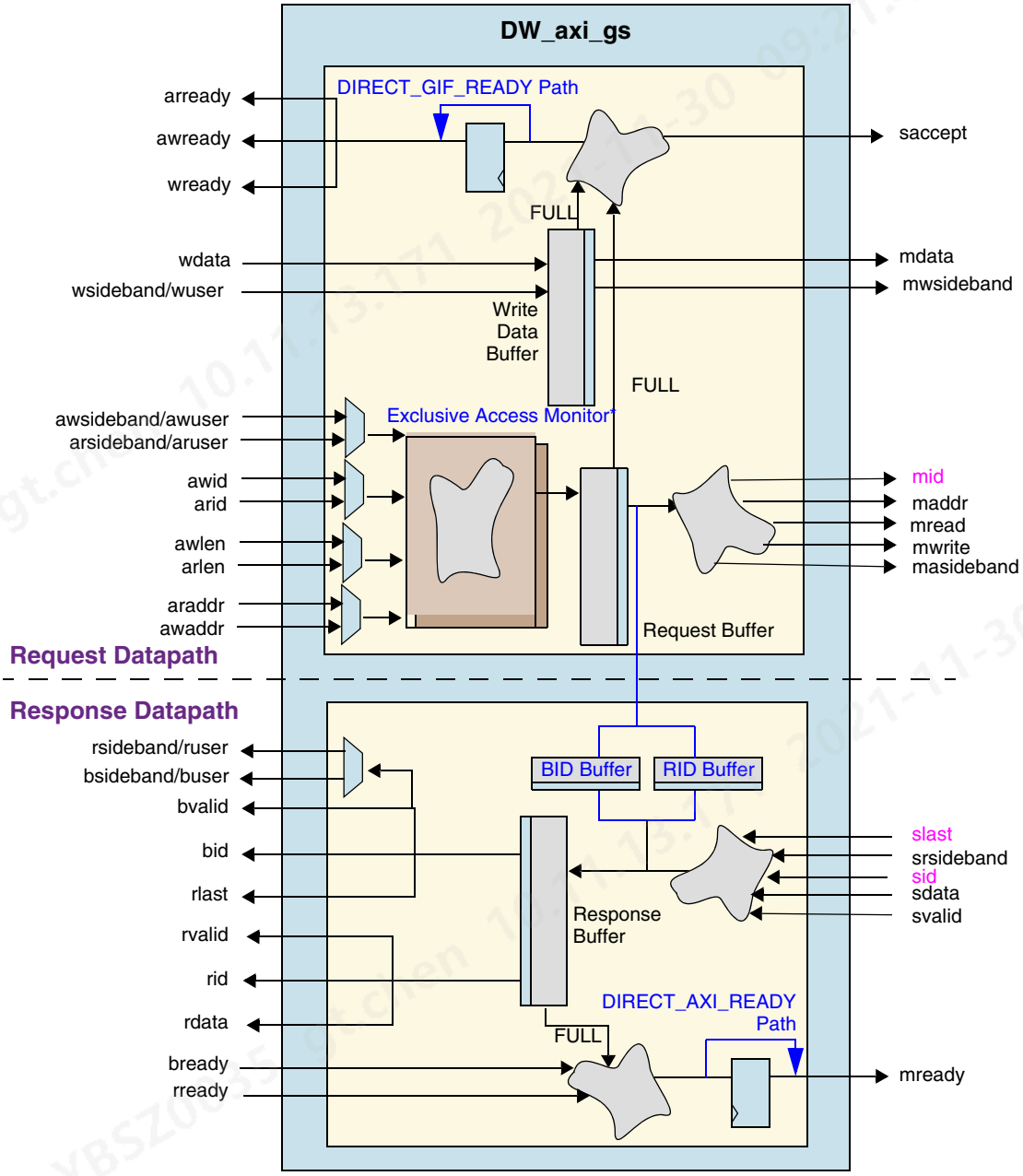
Figure 2-8 Multiple-Beat Read and Write Series



2.5 Microarchitecture

Figure 2-9 illustrates the microarchitecture of the DW_axi_gs. The diagram shows only handshake signals in order to simplify the relationships. DIRECT_GIF_READY, DIRECT_AXI_READY, and Buffer depths are all configurable. See "Parameter Descriptions" on page 39 for configuration parameter details, and the remainder of this chapter for functionality details.

Figure 2-9 DW_axi_gs Microarchitecture



Items in blue are not applicable when the Extended GIF mode is enabled.
Items in pink are applicable *only* when the Extended GIF mode is enabled.

The Request buffer stores address and control signals for read and write requests that come in from the AXI bus. The Write Data buffer stores write data and associated write strobes. The Response buffer stores read data and all responses from read and write transactions on the GIF.

The saccept-to-mread/mwrite handshake does not have a direct combinational path from the saccept signal to the mread/mwrite signals in order to facilitate timing closure. The same holds true for the valid/ready handshake on each AXI bus channel.

2.6 Extended GIF Mode

Extended GIF mode allows high performance GIF slaves to connect to the DW_axi_gs. In the Extended GIF mode, the DW_axi_gs supports ID signal propagation from the AXI read or write interface to the GIF. The sid signal on the GIF response channel collects the generic slave response ID and associates the received response with the rid or bid signal of the AXI read or AXI write response transaction respectively.

To use the Extended GIF mode, the external GIF slave must be able to do the following:

- Sample the mid signal on the request channel for the transaction and drive the same value for the corresponding transaction response on the sid signal.
- Drive the slast signal high for the last beat of the read transaction.

In extended GIF mode, the external GIF slave supports the following features:

- Reordering of read transaction response without interleaving

  For a given ID, the ID responses must follow the order of requests. However, the DW_axi_gs supports out-of-order responses between different IDs.
- Higher number of outstanding transactions
  - When the low-power interface is not enabled (GS_LOWPWR_HS_IF = 0), there is no limitation on the number of outstanding transactions from the DW_axi_gs.
  - When the low-power interface is enabled (GS_LOWPWR_HS_IF = 1), the DW_axi_gs supports a maximum of 128 outstanding transactions.

Figure 2-9 on page 27 illustrates the microarchitecture of DW_axi_gs in Extended GIF mode with handshake.

For transaction examples in the Extended GIF mode, see Figure 2-3 on page 21 and Figure 2-4 on page 22.

2.7 GIF Lite Mode

The GIF Lite mode allows for simple GIF slaves to be connected to DW_axi_gs. To use the GIF Lite mode, the attached external GIF slave must be designed such that it always returns read data on the GIF response channel the cycle after a read request is accepted on the GIF request channel. In the GIF Lite mode, the GIF response channel sresp and svalid input signals are ignored and the DW_axi_gs auto-generates an AXI response of OKAY or EXOKAY. The GIF Lite mode is enabled by the GS_GIF_LITE parameter described on "Parameter Descriptions" on page 39.

If the attached external GIF slave does not have, or does not need, the ability to hold off new GIF request channel transactions, then flow control from the GIF slave can be disabled. To do this, tie the DW_axi_gs saccept input to high.

A GIF slave may choose to ignore the GIF request channel signals mburst, mlen, msize, mlast, and mwstrb. In this case, software is responsible for not issuing transactions with sizes less than the full data bus width for reads or writes to this type of GIF slave.

As an example, if a synchronous SRAM is hooked up to the DW_axi_gs and the GIF Lite mode is enabled, the SRAM is designed to never generate an error response, and the DW_axi_gs always sends an auto-generated OKAY or EXOKAY response on the AXI bus. The response data from the SRAM is designed to arrive, and be sampled, in the next cycle after the GIF request channel saccept/mread signals are both high. Two-way flow control is disabled accordingly, the saccept input to DW_axi_gs is driven high, and DW_axi_gs continuously drives the mready output high.

Note The GIF Lite mode is not supported when the Extended GIF mode is enabled.

2.8 Direct-Ready Feedthroughs

In order to facilitate timing closure, you can configure a direct-ready feed-through in order to have either registered or unregistered ready signals from the GIF side to AXI and vice versa. If GS_DIRECT_AXI_READY is set to true, the AXI ready signals rready and bready are combinationally connected to GIF mready. If GS_DIRECT_GIF_READY is set to true, the GIF saccept signal is combinationally connected to arready, awready and wready.

Note Direct-ready feedthrough is not supported when the Extended GIF mode is enabled.

When DIRECT_AXI_READY and GIF Lite mode are both enabled, the rready and bready signals are combinationally connected to mread and mwrite. When DIRECT_AXI_READY and GIF Lite mode are both enabled and if bready and rready go low, DW_axi_gs can still accept, issue new read, or write requests if the response buffer is not full. If the GIF clock is a lower frequency than the AXI clock, mread and mwrite can change during one AXI clock cycle. The combinational fanin of the mread and mwrite signals comes from registers clocked at the AXI clock frequency, as well as the synchronous, but slower, GIF clock frequency; see "Synchronous Generic Clock" on page 30. Therefore, timing closure on the GIF side can be more difficult to achieve, since a lower-frequency GIF clock does not eliminate the dependency on the AXI clock.

Note There are performance degradation issues when ready signals are registered and the associated buffer depth is too shallow. For more information, see "Performance" on page 37.

2.9 Outstanding Transaction Control

The DW_axi_gs can be configured to support multiple outstanding transactions. The default setting is to allow one outstanding read and one outstanding write. A transaction is considered complete only when the last transaction response has been successfully delivered to the AXI response channel.

On the GIF request channel, the request buffer and the write data buffer can be configured to queue up to four AXI transactions and four AXI write beats, if the GIF request channel holds off new transaction requests or write data transfers.

> **Note** In the Extended GIF mode, there is no limitation on the number of outstanding read and write transactions, unless the low-power interface is enabled. When the Extended GIF mode and the low-power interface are both enabled, a maximum of 128 outstanding read and write transactions are allowed.

After GIF request channel transactions have been accepted by the external GIF slave, the posted write transaction (BID) buffer and the posted read transaction (RID) buffer determine the total number of outstanding write and read GIF transactions allowed. The BID and RID buffers can be configured to allow up to 16 outstanding transactions each.

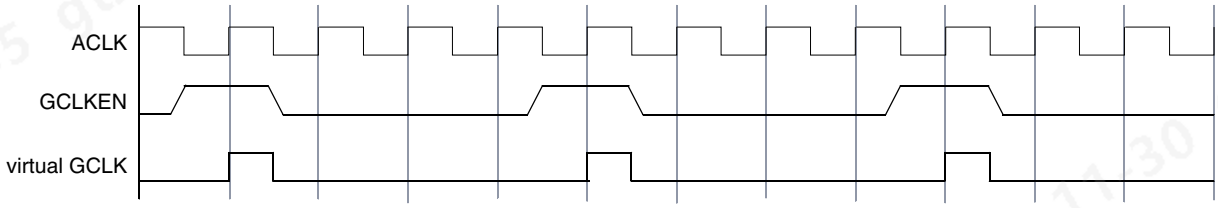
Finally, on the GIF response channel, the response buffer can be configured to queue up to four GIF responses if the AXI response channels holds off transaction responses.

2.10 Synchronous Generic Clock

The AXI and GIF can operate using the same clock, or the GIF can run at a slower, synchronous frequency. To run at a slower frequency, the AXI clock frequency must be an integer multiple of the GIF clock frequency. To run on a slower clock, you must generate an enable signal that enables the AXI clock only on every *N*th clock; this signal must be connected to the gclken port of the DW_axi_gs.

Figure 2-10 illustrates a generic clock that is four times slower that the AXI clock.

Figure 2-10 GCLK = ACLK / 4



2.11 Low-Power Interface

You can configure the DW_axi_gs to include an AXI low-power handshaking interface. This interface allows the following:

- DW_axi_gs informs the system low-power controller (LPC) when it has no outstanding transactions
- LPC requests the DW_axi_gs to enter into a low-power state

You can include this low-power interface in your design by setting the GS_LOWPWR_HS_IF parameter to 1.

The low-power handshaking interface includes the following signals:

- csysreq input – de-asserted by the LPC to initiate a low-power state; asserted by LPC to initiate an exit from a low-power state
- csysack output – de-asserted by DW_axi_gs to acknowledge a request to enter a low-power state; asserted by DW_axi_gs to acknowledge a request to exit a low-power state
- cactive output – de-asserted by DW_axi_gs when the clock can be removed after the next positive edge; csysack must also be deasserted

The sequence of events for entering a low-power state is as follows:

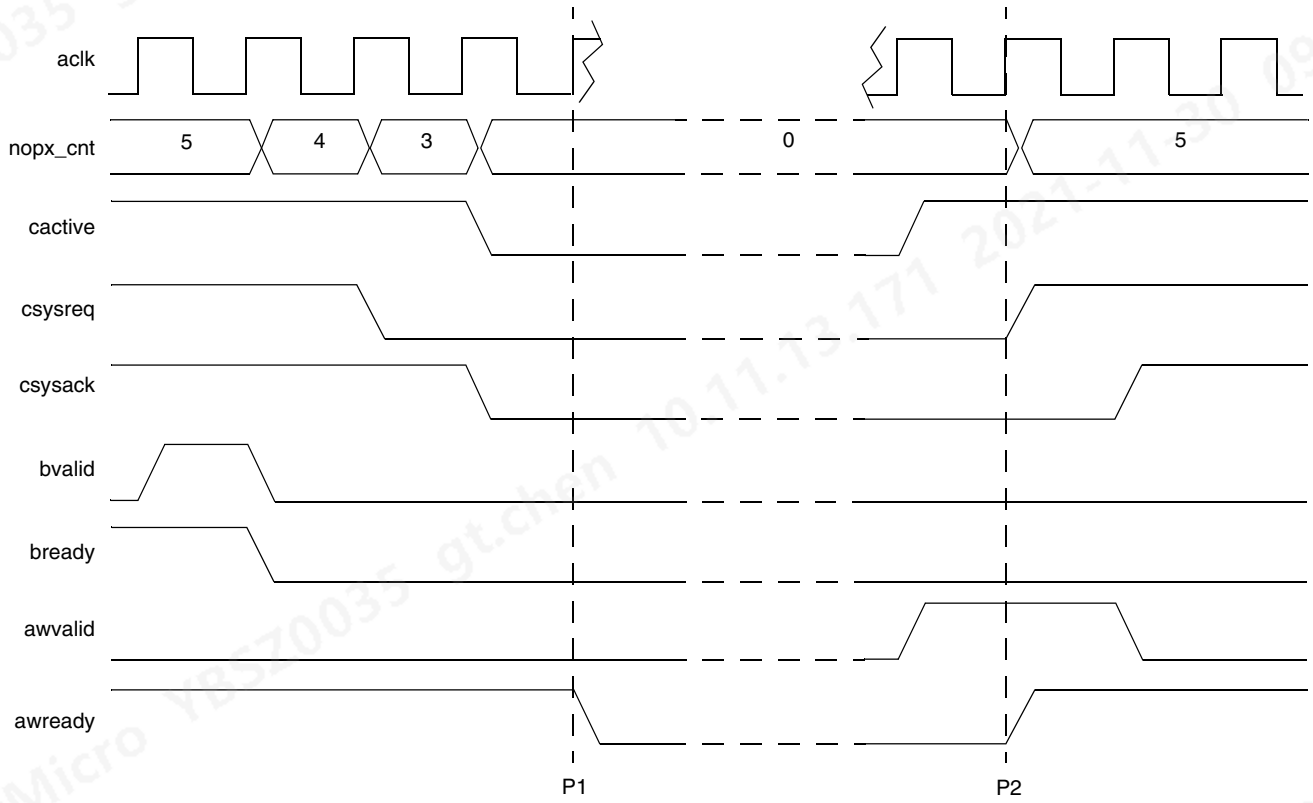
1. The LPC requests the DW_axi_gs to enter a low-power state by de-asserting the csysreq signal.
2. Since the DW_axi_gs does not have a power-up or power-down sequence, it always acknowledges a csysreq signal on the next cycle by asserting or de-asserting the csysack signal.

   When requested by the LPC, the low-power state depends on the value of the cactive signal at the time that the csysack signal is sampled by the LPC after the csysreq signal has been de-asserted.
3. The cactive signal de-asserts when DW_axi_gs has no outstanding transactions. If you set the GS_LOWPWR_NOPX_CNT parameter to X in coreConsultant, the cactive signal de-asserts after *X* cycles, during which no outstanding transactions are elapsed.
4. DW_axi_gs enters a low-power state when *all* of the cactive, csysreq, and csysack signals are low (0) when sampled at the positive edge of aclk; this is illustrated at P1 in Figure 2-11.

Note Entry to a low-power state happens at any cycle in which the above condition is satisfied.

While in a low-power state, the awready, wready, and arready signals are held low, which prevents any new transaction from starting and thus allows the clock to be safely disabled; this is illustrated between points P1 and P2 of Figure 2-11.

Figure 2-11 Low-Power State Entry and Exit (GS_LOWPWR_NOPX_CNT = 5)



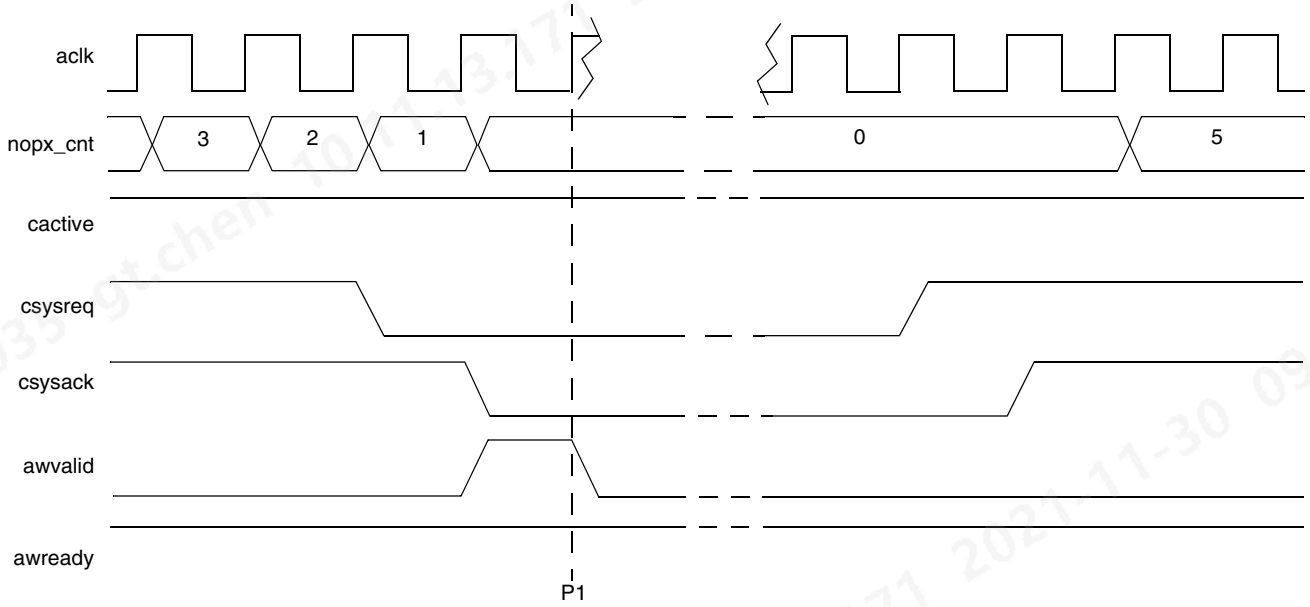
The DW_axi_gs exits a low-power state when the cactive signal goes high and is sampled at the positive edge of aclk; illustrated at P2 of Figure 2-11.

An exit from low-power can be initiated by:

- LPC asserting the csysreq signal, which causes the DW_axi_gs to assert the cactive signal, illustrated in Figure 2-13.
- A master asserting the awvalid or arvalid signals of the DW_axi_gs, which causes the DW_axi_gs to assert the cactive signal, illustrated in Figure 2-11.

Figure 2-12 shows the DW_axi_gs rejecting a request to enter a low-power state. At P1, the no-pending-transaction (nopx_cnt) counter has reached 0, but on this same cycle the awvalid signal asserts, which keeps the cactive signal asserted. The LPC samples at P1 that the DW_axi_gs has rejected the entry to low-power mode, and therefore must not remove the clock.

Figure 2-12 DW_axi_gs Low-Power Interface Rejects Low-Power Entry



2.11.1 cactive Signal De-assertion

The cactive signal de-asserts the GS_LOWPWR_NOPX_CNT parameter *aclk* cycles after the last pending transaction completes. If the csysreq signal de-asserts while the component is counting down to 0 from *GS_LOWPWR_NOPX_CNT*, the cactive signal de-asserts on the next cycle.

After the positive edge of the aclk signal where the cactive signal and the csysack signal are sampled low, the low-power controller (LPC) may remove the clocks from the DW_axi_gs (P1 in Figure 2-11). At this point the bready and awready signals are also be driven low.

2.11.2 cactive Signal Assertion

The cactive signal asserts combinatorially when the awvalid or arvalid signals are asserted, at which point it remains asserted until all outstanding transactions have completed and *GS_LOWPWR_NOPX_CNT* cycles have elapsed.

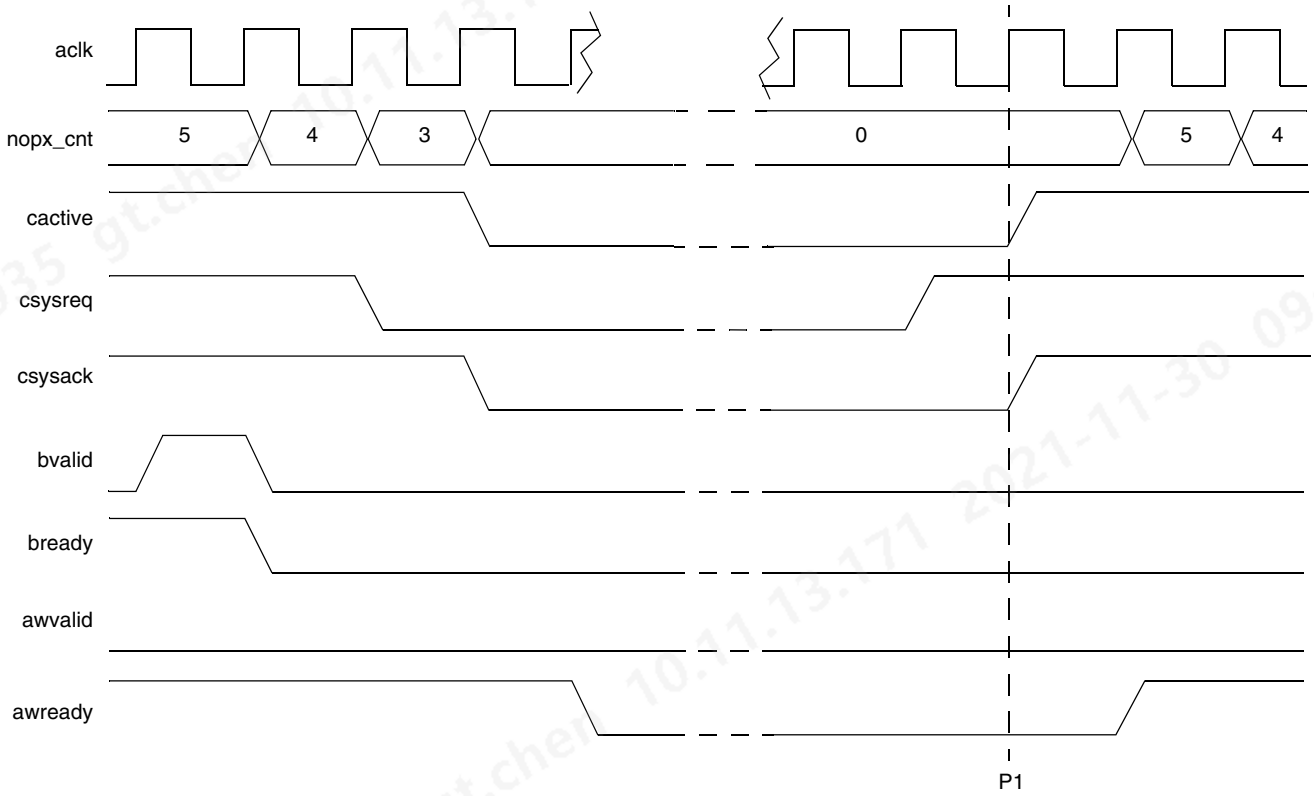
Note Early write data does not cause the cactive signal to assert or to remain asserted.

If the cactive signal is low and the csysreq signal is asserted, the DW_axi_gs asserts the cactive signal at the same time as the csysack signal in order to complete the low-power exit handshake. After the clock edge where the cactive and csysack signals are sampled high on exit from low-power mode, the DW_axi_gs keeps the cactive signal asserted for *GS_LOWPWR_NOPX_CNT* cycles, after which it will either:

- De-assert until there are active transactions again.
- De-assert until another low-power exit handshake is completed.

This can be seen at P1 in Figure 2-13 where the cactive signal is asserted 1 clock cycle after the csysreq signal asserts, and the no-pending-transaction counter (nopx_cnt) starts to decrement again from the next cycle.

Figure 2-13 LPC Initiates Low-Power Exit



Note

When coming out of reset, if the awvalid and arvalid signals equal 0:

- cactive signal equals 0, if GS_LOWPWR_NOPX_CNT parameter equals 0
- cactive signal equals 1, if GS_LOWPWR_NOPX_CNT parameter is greater than 0 and it de-asserts when *GS_LOWPWR_NOPX_CNT* clock cycles have elapsed

2.12 AXI4 Optional Signaling Interface

When the AXI4 interface is enabled, the following optional signals can be enabled in the DW_axi_gs:

- QoS signals (awqos and arqos), for priority signaling.
- Region signals (awregion and arregion), for multi-region signaling.
- User signals (awuser, aruser, wuser, ruser, and buser) for user data signaling.

The QoS and region signals are not processed by the DW_axi_gs; the DW_axi_gs just transfers the information in these signals from the AXI4 interface to the GIF.

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

Figure 2-14 shows the optional signals for an AXI4 write transaction. Note the following:

1. The awuser signal on the AXI4 write address channel is qualified by the awvalid signal and transferred to the masideband signal on the GIF request channel, which is qualified by the mwrite signal.
2. The wuser signal on the AXI4 write data channel is qualified by the wvalid signal and transferred to the mwsideband signal on the GIF request channel, which is qualified by the mwrite signal.
3. The srsideband signal on the GIF response channel is qualified by the svalid signal and transferred to the buser signal on the AXI4 write response channel, which is qualified by the bvalid signal.
4. The awqos signal on the AXI4 write address channel is qualified by the awvalid signal and transferred to the mqos on the GIF request channel, which is qualified by the mwrite signal.
5. The awregion signal on the AXI4 write address channel is qualified by the awvalid signal and transferred to the mregion signal on the GIF request channel, which is qualified by the mwrite signal.

Figure 2-14 Optional Signals in an AXI4 Write Transaction

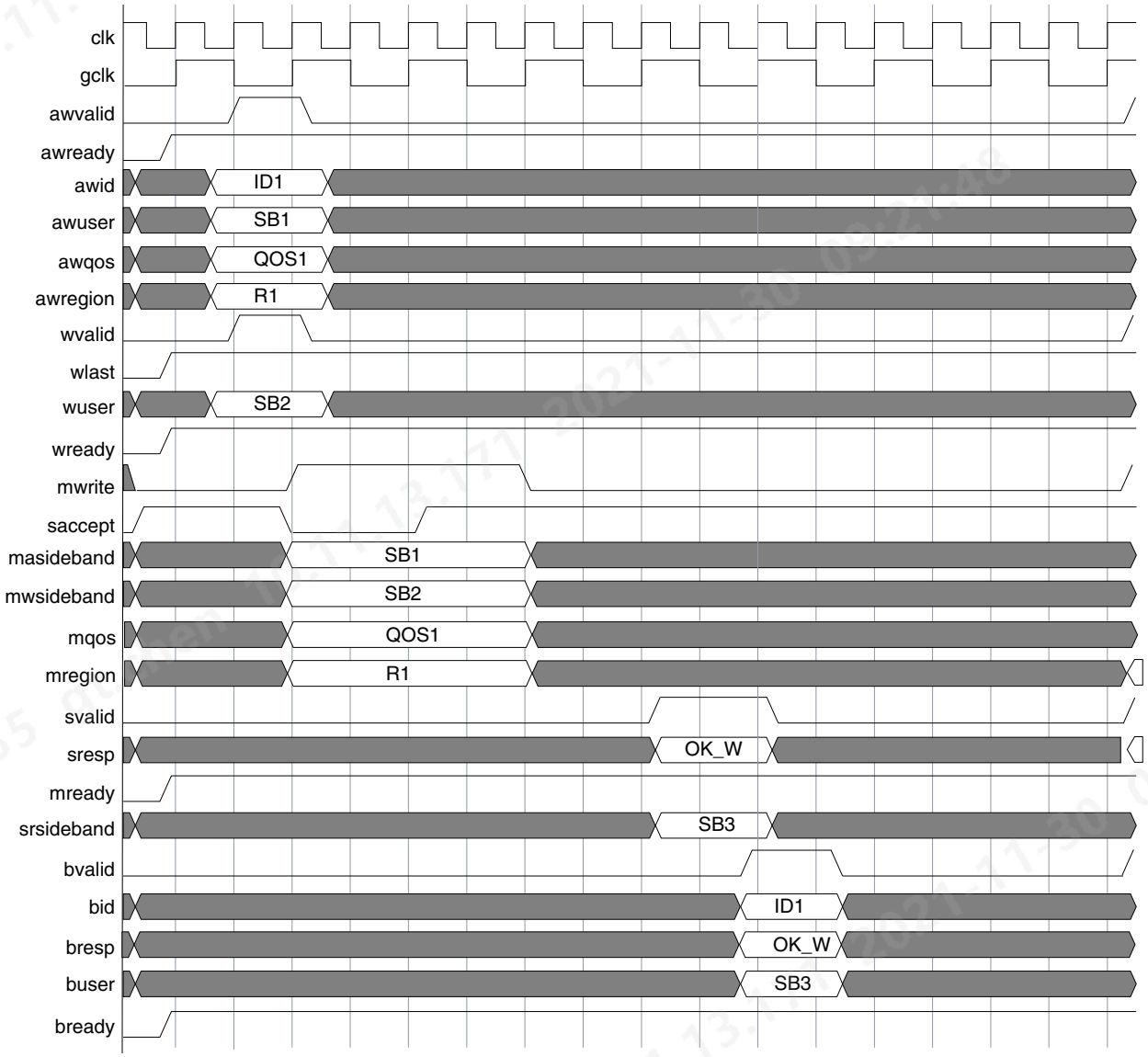
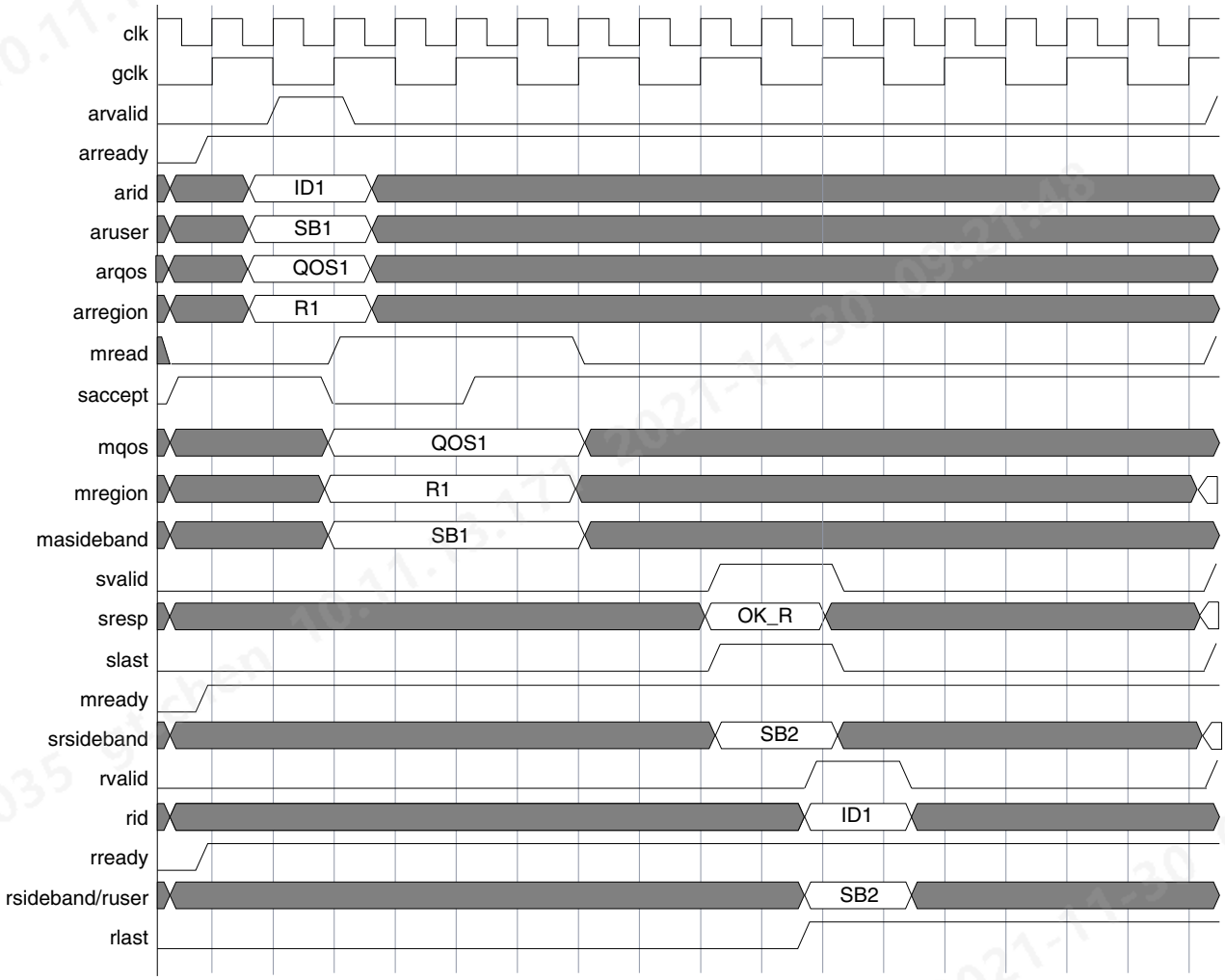


Figure 2-15 on page 36 shows the optional signals for an AXI4 write transaction. Note the following:

1. The aruser signal on the AXI4 read address channel is qualified by the arvalid signal and transferred to the masideband signal on the GIF request channel, which is qualified by the mread signal.
2. The srsideband signal on the GIF response channel is qualified by the svalid signal and transferred to the ruser signal on the AXI4 read data channel, which is qualified by the rvalid signal.
3. The arqos signal on the AXI4 read address channel is qualified by the arvalid signal and transferred to the mqos signal on the GIF request channel, which is qualified by the mread signal.
4. The arregion signal on the AXI4 read address channel is qualified by the arvalid signal and transferred to the mregion signal on the GIF request channel, which is qualified by the mread signal.

Figure 2-15 Optional Signals in an AXI4 Read Transaction



2.13 Exclusive Access Monitor

The DW_axi_gs supports exclusive accesses from the AXI bus. You can configure exclusive accesses, which are monitored in parallel up to the configured maximum, set by the GS_AXI_EX_ACCESS parameter. The number of outstanding exclusive accesses are determined by the number of exclusive reads that have not been completed yet with a corresponding exclusive write. When the configured maximum number of outstanding exclusive accesses is reached, any additional exclusive reads fail with a response of OKAY or SLVERR, but not EXOKAY. The AXI master can retry the failed exclusive read, which is accepted once one of the outstanding exclusive accesses has completed.

Exclusive access addresses are monitored for the maximum allowable size of 128 bytes for every access, regardless of the actual size and length of the exclusive read.

An exclusive write fails if any byte of the 128 byte address range is written to since the exclusive read. If an exclusive write fails, a GIF write transaction still occurs, but all the write strobes are turned off, so no data is actually written to the GIF slave.

If the DW_axi_gs GS_AXI_EX_ACCESS parameter is configured to support no (0) exclusive accesses, the DW_axi_gs behaves like an AXI slave that does not support exclusive accesses at all, at which point all exclusive accesses fail.

Note Exclusive Access is not supported when the Extended GIF mode is enabled.

2.14 Performance

The DW_axi_gs matches the performance of the AXI bus with the smallest possible footprint when used in the default minimum configuration. Performance can be measured with respect to throughput, latency, and clock speed.

Using the default configuration for the Direct-Ready parameters, latency occurs over one clock cycle such that an access started on the AXI side is asserted on the GIF side with the delay of one clock cycle. However, a clock cycle is measured on the GIF. If the GIF clock is slowed down by a factor of *N*, compared to the AXI clock, the access on the AXI interface is delayed by up to *N* AXI clock cycles. For more information, see "Synchronous Generic Clock" on page 30.

Even with a slower GIF clock, throughput has 100% efficiency; that is, data streams through the DW_axi_gs without wait cycles when the receiving side is ready to accept the data. This is true for request and response data path.

There are some configurations that reduce throughput. Table 2-1 shows throughput as it depends on the configured values of the Direct-Ready and buffer depth parameters; this applies to all buffers when GIF Lite mode is disabled. The buffers in the request path (Request, Write Data, and Posted Read/Write Transaction Buffers) are affected only by DIRECT_GIF_READY. The buffer in the response path (Response Buffer) is affected only by DIRECT_AXI_READY. This enables you to independently configure the throughput for the Request and Response paths.

Table 2-1 Throughput Dependency on Configuration (GIF Lite Mode Disabled)

	Buffer depth ==1	Buffer depth >= 2
Direct-Ready False	1/2	full
Direct-Ready True	full (default)	full

Table 2-2 shows the throughput dependency when GIF Lite is enabled. However, it applies only to the Response datapath; that is, the Response Buffer depth and DIRECT_AXI_READY. The Request datapath throughput dependency is unaffected by GIF Lite mode, and is still described by Table 2-1.

Table 2-2 Throughput Dependency for Response Datapath Only (GIF Lite Mode Enabled)

	Buffer depth == 1	Buffer depth == 2	Buffer depth >= 3
Direct-Ready False	1/3	1/2	full
Direct-Ready True	1/2	full	full

For the throughput performance described, it is assumed that the limit of outstanding transactions is not reached. If the outstanding transaction limit required the DW_axi_gs to hold off new transactions, the throughput is affected, since subsequent transactions does not pass the DW_axi_gs until the responses of preceding transactions have completed.

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.

- Toplevel Parameters on page 40
- Performance Parameters on page 42
- Sideband/User Signals on page 44
- Low Power Interface Configuration on page 46

3.1 Toplevel Parameters

Table 3-1 Toplevel Parameters

Label	Description
Toplevel Parameters	
Select Interface AXI3/AXI4.	Select AXI Interface Type as AXI3 or AXI4. By default, DW_axi_gs supports the AXI3 interface. **Values:** ■ AXI3 (0) ■ AXI4 (1) **Default Value:** AXI3 **Enabled:** DWC-AMBA-Fabric-Source license required. **Parameter Name:** GS_AXI_INTERFACE_TYPE
AXI & GIF Address Width	Address width on AXI and GIF interfaces. **Values:** 32, ..., 64 **Default Value:** 32 **Enabled:** Always **Parameter Name:** GS_AW
AXI & GIF Data Width	Data width on AXI and GIF interfaces. No distinction is made between read and write channels. **Values:** 8, 16, 32, 64, 128, 256, 512 **Default Value:** 32 **Enabled:** Always **Parameter Name:** GS_DW
AXI & GIF ID Width	Width of transaction ID field on the AXI (awid, arid, wid, rid, bid) and GIF (mid, sid) interfaces. **Values:** 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16 **Default Value:** 12 **Enabled:** Always **Parameter Name:** GS_ID
AXI & GIF Burst Length Width	Width of burst length field on the AXI and GIF interfaces. **Values:** 4, 5, 6, 7, 8 **Default Value:** 4 **Enabled:** Always **Parameter Name:** GS_BW

Table 3-1 Toplevel Parameters (Continued)

Label	Description
AXI Number of Exclusive Accesses	Maximum number of exclusive accesses supported in parallel. A value of 0 means exclusive accesses are not supported. **Values:** 0, ..., 16 **Default Value:** (EXTD_GIF == 0) ? 1 : 0 **Enabled:** EXTD_GIF == 0 **Parameter Name:** GS_AXI_EX_ACCESS
GIF Lite	Lite version of GIF. Supports devices with one-cycle data response and no flow control. **Values:** ■ false (0) ■ true (1) **Default Value:** false **Enabled:** EXTD_GIF == 0 **Parameter Name:** GS_GIF_LITE
Support for Extended GIF Mode?	Support for Extended GIF Mode. It enables the mid, sid, and slast signals on the GIF Interface. **Values:** ■ false (0) ■ true (1) **Default Value:** false **Enabled:** DWC-AMBA-Fabric-Source license required **Parameter Name:** EXTD_GIF
Include QoS Signals ?	If set to True, the QoS is enabled in DW_axi_gs, and the write address and read address channels on the GIF and the AXI Interfaces have QoS signals on I/Os. **Values:** ■ false (0) ■ true (1) **Default Value:** false **Enabled:** GS_AXI_INTERFACE_TYPE > 0 **Parameter Name:** GS_AXI_HAS_QOS
Include Region Signals?	If set to True, write address channel and read address channels on GIF and AXI Interface have region signals on the I/Os. **Values:** ■ false (0) ■ true (1) **Default Value:** false **Enabled:** GS_AXI_INTERFACE_TYPE > 0 **Parameter Name:** GS_AXI_HAS_REGION

3.2 Performance Parameters

Table 3-2 Performance Parameters

Label	Description
Request Path Buffer Configuration	
Combinational GIF Ready	If true, GIF saccept input is combinationally connected to AXI awready, wready, and arready outputs. If false, GIF saccept input is registered, inserting one cycle of latency. **Values:** ■ false (0) ■ true (1) **Default Value:** true **Enabled:** EXTD_GIF == 0 **Parameter Name:** GS_DIRECT_GIF_READY
Request Buffer Depth	Depth of combined read and write AXI request buffer. Higher values allow AXI requests to be buffered, rather than stalled, if GIF request channel stalls DW_axi_gs transactions. **Values:** 1, 2, 3, 4 **Default Value:** 1 **Enabled:** Always **Parameter Name:** GS_REQ_BUFFER
Write Data Buffer Depth	Depth of AXI write data buffer. Higher values allow AXI write data to be buffered, rather than being stalled, if GIF request channel stalls DW_axi_gs transactions data. **Values:** 1, 2, 3, 4 **Default Value:** 1 **Enabled:** Always **Parameter Name:** GS_WDATA_BUFFER
Response Path Buffer Configuration	
Combinational AXI Ready	If true, AXI rready and bready inputs are combinationally connected to GIF mready outputs. If false, AXI rready and bready inputs are registered, inserting one cycle of latency. **Values:** ■ false (0) ■ true (1) **Default Value:** true **Enabled:** EXTD_GIF == 0 **Parameter Name:** GS_DIRECT_AXI_READY

Table 3-2 Performance Parameters (Continued)

Label	Description
Response Buffer Depth	Depth of GIF response buffer. Higher values allow GIF responses to be buffered, rather than being stalled, if AXI read data or write response channel stall DW_axi_gs responses. **Values:** 1, 2, 3, 4 **Default Value:** 1 **Enabled:** Always **Parameter Name:** GS_RESP_BUFFER
ID Buffer Configuration	
Posted write transactions limit	Depth of BID buffer for outstanding GIF write requests. If set to 1, equivalent to blocking GIF write; current GIF write requests must complete (write response received from GIF Response Channel) before next AXI write request is accepted by DW_axi_gs. If greater than 1, AXI write IDs are allowed to be queued in BID buffer, while write request is transferred to GIF Request Channel before previous GIF writes responses are completed (up to the configured limit). **Values:** 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16 **Default Value:** 1 **Enabled:** EXTD_GIF == 0 **Parameter Name:** GS_BID_BUFFER
Posted read transactions limit	Depth of RID buffer for outstanding RID requests. If set to 1 (which is equivalent to blocking GIF read) current GIF read requests must complete (read response received from GIF response channel) before next AXI read request is accepted by DW_axi_gs. If greater than 1, AXI read IDs are allowed to be queued in RID buffer, while read request is transferred to GIF Request Channel before previous GIF read responses are completed (up to the configured limit). **Values:** 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16 **Default Value:** 1 **Enabled:** EXTD_GIF == 0 **Parameter Name:** GS_RID_BUFFER

3.3 Sideband/User Signals Parameters

Table 3-3 Sideband/User Signals Parameters

Label	Description
HasSidebandOrUser	
Include Sideband/User Bus for Write Address Channels?	If set to True, then all AXI write address and GIF address channels have an associated sideband/user bus in AXI3/AXI4 mode, respectively. The write address channel sideband/user bus is routed in the same way as the other write address channel control signals. **Values:** ■ false (0) ■ true (1) **Default Value:** false **Enabled:** DWC-AMBA-Fabric-Source license required **Parameter Name:** GS_HAS_AWSB
Include Sideband Bus for Write Data Channels?	If set to True, then both AXI and GIF write data channels have an associated sideband/user bus in AXI3/AXI4 mode, respectively. The write data channel sideband/user bus is routed in the same way as the other write data channel control signals. **Values:** ■ false (0) ■ true (1) **Default Value:** false **Enabled:** DWC-AMBA-Fabric-Source license required **Parameter Name:** GS_HAS_WSB
Include Sideband Bus for Write Response Channels?	If set to True, then both AXI write response and GIF response channels have an associated sideband/user bus in AXI3/AXI4 mode, respectively. The write response channel sideband/user bus is routed in the same way as the other write response channel control signals. **Values:** ■ false (0) ■ true (1) **Default Value:** false **Enabled:** DWC-AMBA-Fabric-Source license required **Parameter Name:** GS_HAS_BSB

Table 3-3 Sideband/User Signals Parameters (Continued)

Label	Description
Include Sideband Bus for Read Address Channels?	If set to True, then both AXI read address and GIF address channels have an associated sideband/user bus in AXI3/AXI4 mode, respectively. The read address channel sideband/user bus is routed in the same way as the other read address channel control signals. **Values:** ■ false (0) ■ true (1) **Default Value:** false **Enabled:** DWC-AMBA-Fabric-Source license required **Parameter Name:** GS_HAS_ARSB
Include Sideband Bus for Read Data Channels?	If set to True, then both AXI read data and GIF response channels have an associated sideband/user bus in AXI3/AXI4 mode, respectively. The read data channel sideband/user bus is routed in the same way as the other read data channel control signals. **Values:** ■ false (0) ■ true (1) **Default Value:** false **Enabled:** DWC-AMBA-Fabric-Source license required **Parameter Name:** GS_HAS_RSB
Width	
Width of the Write Address Channel Sideband bus	When the GS_HAS_AWSB or GS_HAS_ARSB parameter is set to True, you can set the address channel sideband/user bus width. **Values:** 1, ..., GS_MAX_SBW **Default Value:** 1 **Enabled:** GS_HAS_AWSB == 1 || GS_HAS_ARSB == 1 **Parameter Name:** GS_A_SBW
Width of the Write Data Channel Sideband bus	When the GS_HAS_WSB parameter is set to True, you can set the write address channel sideband/user bus width. **Values:** 1, ..., GS_MAX_SBW **Default Value:** 1 **Enabled:** GS_HAS_WSB == 1 **Parameter Name:** GS_W_SBW
Width of the Write Address Channel Sideband bus	When the GS_HAS_RSB or GS_HAS_BSB parameter is set to True, you can set the read-data/write-response channel sideband/user bus width. **Values:** 1, ..., GS_MAX_SBW **Default Value:** 1 **Enabled:** GS_HAS_BSB == 1 || GS_HAS_RSB == 1 **Parameter Name:** GS_R_SBW

3.4 Low Power Interface Configuration Parameters

Table 3-4 Low Power Interface Configuration Parameters

Label	Description
Low Power Interface Configuration	
Low Power Interface Enable	If true, the low-power handshaking interface (csysreq, csysack, and cactive signals) and associated control logic is implemented. If false, no support for low-power handshaking interface is provided. **Values:** ■ false (0) ■ true (1) **Default Value:** false **Enabled:** Always **Parameter Name:** GS_LOWPWR_HS_IF
Inactive Clock cycles before power down	Number of AXI clock cycles to wait before cactive signal de-asserts, when there are no pending transactions. Note that if csysreq de-asserts while waiting this number of cycles, cactive will immediately de-assert. If a new transaction is initiated during the wait period, the counting will be halted, cactive will not deassert, and the counting will be re-initiated when there are no pending transactions. This parameter is available only if GS_LOWPWR_HS_IF is true. **Values:** 0, ..., 4294967295 **Default Value:** 0 **Enabled:** GS_LOWPWR_HS_IF==1 **Parameter Name:** GS_LOWPWR_NOPX_CNT

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 49
- AXI Write Address Channel on page 50
- AXI Write Data Channel on page 54
- AXI Write Response Channel on page 57
- AXI Read Address Channel on page 59
- AXI Read Data Channel on page 63
- GIF Request Channel on page 66
- GIF Response Channel on page 70
- AXI Low Power Interface on page 72

4.1 Clock and Resets Signals

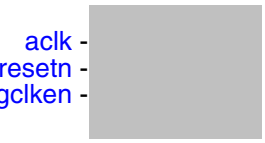


Table 4-1 Clock and Resets Signals

Port Name	I/O	Description
aclk	I	AXI clock. **Exists:** Always **Synchronous To:** None **Registered:** N/A **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
aresetn	I	AXI reset. Asynchronous assertion, synchronous de-assertion. The reset must be synchronously de-asserted after rising edge of aclk. DW_axi_gs 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
gclken	I	Clock enable signal. Allows operation on clock slower than aclk. **Exists:** Always **Synchronous To:** aclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** High

4.2 AXI Write Address Channel Signals

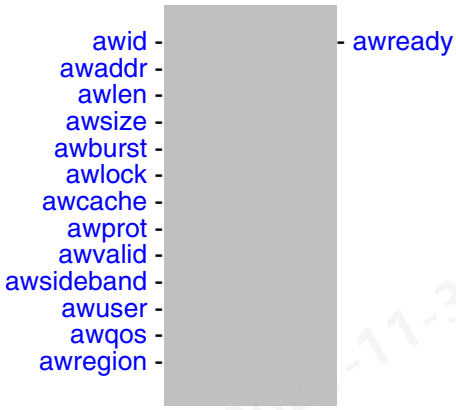


Table 4-2 AXI Write Address Channel Signals

Port Name	I/O	Description
awready	O	AXI write address ready. Indicates that slave is ready to accept 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
awid[(GS_ID-1):0]	I	AXI write address identification. Provides identification tag for write address signals. **Exists:** Always **Synchronous To:** aclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
awaddr[(GS_AW-1):0]	I	AXI write address. Specifies address of first transfer in write burst transaction. Associated control signals used to determine addresses of remaining transfers in burst. **Exists:** Always **Synchronous To:** aclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A

Table 4-2 AXI Write Address Channel Signals (Continued)

Port Name	I/O	Description
awlen[(GS_BW-1):0]	I	AXI Burst length. Specifies exact number of transfers in the burst and determines number of data transfers associated with the address. **Exists:** Always **Synchronous To:** aclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
awsize[2:0]	I	AXI write burst size. Indicates size of each transfer in the burst. Byte lane strobes indicate exact byte lanes to update. **Exists:** Always **Synchronous To:** aclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
awburst[1:0]	I	AXI Write Burst. Combined with size information, this signal shows how address for each transfer with burst is calculated. **Exists:** Always **Synchronous To:** aclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
awlock[(((GS_AXI_INTERFACE_TYPE==0)?2:1)-1):0]	I	AXI write lock. Provides additional information about characteristics of a transfer. **Exists:** Always **Synchronous To:** aclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
awcache[3:0]	I	AXI write cache. Indicates bufferable, cacheable, write-through, and write-back attributes of transaction. This signal is unused. It is included for interface consistency only. **Exists:** Always **Synchronous To:** None **Registered:** N/A **Power Domain:** SINGLE_DOMAIN **Active State:** N/A

Table 4-2 AXI Write Address Channel Signals (Continued)

Port Name	I/O	Description
awprot[2:0]	I	AXI write protection. Indicates normal, privileged, or secure protection level of transaction and whether transaction is data access or instruction access. This signal is unused. It is included for interface consistency only. **Exists:** Always **Synchronous To:** None **Registered:** N/A **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
awvalid	I	AXI write address valid. Indicates valid write address and control information are available. Address and control information remain stable until awready signal is high. ■ 0: Address and control information not available. ■ 1: Address and control information available. **Exists:** Always **Synchronous To:** aclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** High
awsideband[(GS_A_SBW_INT-1):0]	I	AXI Sideband signal for write address channel. The signal name changes between the AXI3 and the AXI4 configurations. ■ When the AXI3 interface is enabled, this signal is named awsideband. ■ When the AXI4 interface is enabled, this signal is named awuser. This signal is transferred to the GIF masideband for write transactions. **Exists:** (GS_HAS_AWSB == 1) && (GS_AXI_INTERFACE_TYPE == 0) **Synchronous To:** aclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A

Table 4-2 AXI Write Address Channel Signals (Continued)

Port Name	I/O	Description
awuser[(GS_A_SBW_INT-1):0]	I	AXI User signal for write address channel. The signal name changes between the AXI3 and the AXI4 configurations. ■ When the AXI3 interface is enabled, this signal is named awsideband. ■ When the AXI4 interface is enabled, this signal is named awuser. This signal is transferred to the GIF masideband for write transactions. **Exists:** (GS_HAS_AWSB == 1) && (GS_AXI_INTERFACE_TYPE > 0) **Synchronous To:** aclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
awqos[3:0]	I	Optional. Quality of Service identifier, conveying priority information associated with each transaction on Write Address channel of the AXI Interface. **Exists:** (GS_AXI_HAS_QOS == 1) && (GS_AXI_INTERFACE_TYPE > 0) **Synchronous To:** aclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
awregion[3:0]	I	Write address channel region identifier. Permits a single physical interface on a slave to be used for multiple logical interfaces implemented only in AXI4. **Exists:** (GS_AXI_HAS_REGION == 1) && (GS_AXI_INTERFACE_TYPE > 0) **Synchronous To:** aclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A

4.3 AXI Write Data Channel Signals

Table 4-3 AXI Write Data Channel Signals

Port Name	I/O	Description
wready	O	AXI write ready. 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
wid[(GS_ID-1):0]	I	AMBA AXI3 write identification tag of write data transfer. The value must match the awid value of write transaction. This signal is unused. It is included for interface consistency only. **Exists:** (GS_AXI_INTERFACE_TYPE == 0) **Synchronous To:** None **Registered:** N/A **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
wdata[(GS_DW-1):0]	I	AXI write data. **Exists:** Always **Synchronous To:** aclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A

Table 4-3 AXI Write Data Channel Signals (Continued)

Port Name	I/O	Description
wstrb[((GS_DW/8)-1):0]	I	AXI write strobes. Indicates byte lanes to update in memory. There is one write strobe for each eight bits of the write data bus. WSTRB[n] is associated with the following byte of WDATA: WDATA[8n+7: 8n] **Exists:** Always **Synchronous To:** aclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** High
wlast	I	AXI write last. Indicates the last transfer in a write burst. This signal is unused. It is included for interface consistency only. **Exists:** Always **Synchronous To:** None **Registered:** N/A **Power Domain:** SINGLE_DOMAIN **Active State:** High
wvalid	I	AXI write valid. This signal indicates that valid write data and strobes are available. ■ 0: Write data and strobes not available ■ 1: Write data and strobes available **Exists:** Always **Synchronous To:** aclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** High
wsideband[(GS_W_SBW_INT-1):0]	I	AXI Sideband signal for write data channel. The signal name changes between the AXI3 and the AXI4 configurations. ■ When the AXI3 interface is enabled, this signal is named wsideband. ■ When the AXI4 interface is enabled, this signal is named wuser. This signal is transferred to the GIF mwsideband for write transactions. **Exists:** (GS_HAS_WSB == 1) && (GS_AXI_INTERFACE_TYPE == 0) **Synchronous To:** aclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A

Table 4-3 AXI Write Data Channel Signals (Continued)

Port Name	I/O	Description
wuser[(GS_W_SBW_INT-1):0]	I	AXI User signal for write data channel. The signal name changes between the AXI3 and the AXI4 configurations. ■ When the AXI3 interface is enabled, this signal is named wsideband. ■ When the AXI4 interface is enabled, this signal is named wuser. This signal is transferred to the GIF mwsideband for write transactions. **Exists:** (GS_HAS_WSB == 1) && (GS_AXI_INTERFACE_TYPE > 0) **Synchronous To:** aclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A

4.4 AXI Write Response Channel Signals

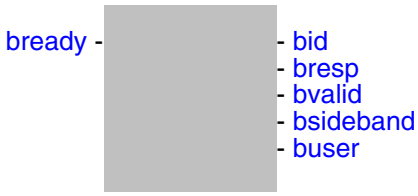


Table 4-4 AXI Write Response Channel Signals

Port Name	I/O	Description
bid[(GS_ID-1):0]	O	AXI write response identification tag. **Exists:** Always **Synchronous To:** aclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
bresp[1:0]	O	AXI write response. Indicates status of write transaction. **Exists:** Always **Synchronous To:** aclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
bvalid	O	AXI write response valid. Indicates that valid write response is available. ■ 0: Write response not available. ■ 1: Write response available. **Exists:** Always **Synchronous To:** aclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** High

Table 4-4 AXI Write Response Channel Signals (Continued)

Port Name	I/O	Description
bsideband[(GS_R_SBW_INT-1):0]	O	AXI Sideband signal for write response channel. The signal name changes between the AXI3 and the AXI4 configurations. ■ When the AXI3 interface is enabled, this signal is named bsideband. ■ When the AXI4 interface is enabled, this signal is named buser. This signal is transferred to the GIF srsideband for write transactions. **Exists:** (GS_HAS_BSB == 1) && (GS_AXI_INTERFACE_TYPE == 0) **Synchronous To:** aclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
buser[(GS_R_SBW_INT-1):0]	O	AXI User signal for write response channel. The signal name changes between the AXI3 and the AXI4 configurations. ■ When the AXI3 interface is enabled, this signal is named bsideband. ■ When the AXI4 interface is enabled, this signal is named buser. This signal is transferred to the GIF srsideband for write transactions.. **Exists:** (GS_HAS_BSB == 1) && (GS_AXI_INTERFACE_TYPE > 0) **Synchronous To:** aclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
bready	I	AXI response ready. Indicates that a master can accept response information. ■ 0: Master not ready. ■ 1: Master ready. **Exists:** Always **Synchronous To:** aclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** High

4.5 AXI Read Address Channel Signals

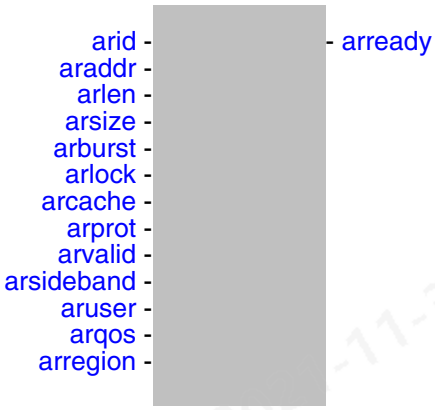


Table 4-5 AXI Read Address Channel Signals

Port Name	I/O	Description
aready	O	AXI read address ready. Indicates that slave is ready to accept 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
arid[(GS_ID-1):0]	I	AXI read address identification. Provides identification tag for read address signals. **Exists:** Always **Synchronous To:** aclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
araddr[(GS_AW-1):0]	I	AXI read address. Provides initial address of read burst transaction. Only start address of the burst is provided, and control signals issued alongside address show how the address is calculated for remaining transfers in the burst. **Exists:** Always **Synchronous To:** aclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A

Table 4-5 AXI Read Address Channel Signals (Continued)

Port Name	I/O	Description
arlen[(GS_BW-1):0]	I	Burst length. The burst length provides the exact number of transfers in a burst and 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[2:0]	I	AXI read burst size. Indicates size of each transfer in burst. **Exists:** Always **Synchronous To:** aclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
arburst[1:0]	I	AXI read Burst. Combined with size information and shows how the address for each transfer with burst is calculated. **Exists:** Always **Synchronous To:** aclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
arlock[(((GS_AXI_INTERFACE_TYPE==0)?2:1)-1):0]	I	AXI read lock. Encoded value indicates whether the transfer is normal, exclusive, or locked access. **Exists:** Always **Synchronous To:** aclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
arcache[3:0]	I	AXI read cache. Indicates bufferable, cacheable, read-through, and read-back attributes of a transaction. This signal is unused. It is included for interface consistency only. **Exists:** Always **Synchronous To:** None **Registered:** N/A **Power Domain:** SINGLE_DOMAIN **Active State:** N/A

Table 4-5 AXI Read Address Channel Signals (Continued)

Port Name	I/O	Description
arprot[2:0]	I	AXI read protection. Indicates normal, privileged, or secure protection level of transaction and whether transaction is data access or instruction access. This signal is unused. It is included for interface consistency only. **Exists:** Always **Synchronous To:** None **Registered:** N/A **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
arvalid	I	AXI read address valid. Indicates that valid write address and control information are available. ■ 0: Address and control information not valid ■ 1: Address and control information valid **Exists:** Always **Synchronous To:** aclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** High
arsideband[(GS_A_SBW_INT-1):0]	I	AXI Sideband signal for read address channel. The signal name changes between the AXI3 and the AXI4 configurations. ■ When the AXI3 interface is enabled, this signal is named arsideband. ■ When the AXI4 interface is enabled, this signal is named aruser. This signal is transferred to the GIF masideband for read transactions. **Exists:** (GS_HAS_ARSB == 1) && (GS_AXI_INTERFACE_TYPE == 0) **Synchronous To:** aclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A

Table 4-5 AXI Read Address Channel Signals (Continued)

Port Name	I/O	Description
aruser[(GS_A_SBW_INT-1):0]	I	AXI User signal for read address channel. The signal name changes between the AXI3 and the AXI4 configurations. ■ When the AXI3 interface is enabled, this signal is named arsideband. ■ When the AXI4 interface is enabled, this signal is named aruser. This signal is transferred to the GIF masideband for read transactions. **Exists:** (GS_HAS_ARSB == 1) && (GS_AXI_INTERFACE_TYPE > 0) **Synchronous To:** aclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
arqos[3:0]	I	Optional. Quality of Service identifier. This signal conveys priority information associated with each transaction on Read Address channel of AXI Interface. **Exists:** (GS_AXI_HAS_QOS == 1) && (GS_AXI_INTERFACE_TYPE > 0) **Synchronous To:** aclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
arregion[3:0]	I	Read address channel region identifier. Permits a single physical interface on a slave to be used for multiple logical interfaces. **Exists:** (GS_AXI_HAS_REGION == 1) && (GS_AXI_INTERFACE_TYPE > 0) **Synchronous To:** aclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A

4.6 AXI Read Data Channel Signals

Table 4-6 AXI Read Data Channel Signals

Port Name	I/O	Description
rid[(GS_ID-1):0]	O	AXI Read identification tag. **Exists:** Always **Synchronous To:** aclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
rdata[(GS_DW-1):0]	O	AXI read data. **Exists:** Always **Synchronous To:** aclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
rresp[1:0]	O	AXI read response. Indicates status of read transaction. **Exists:** Always **Synchronous To:** aclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
rlast	O	AXI read last. Indicates last transfer in read burst. **Exists:** Always **Synchronous To:** aclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** High

Table 4-6 AXI Read Data Channel Signals (Continued)

Port Name	I/O	Description
rvalid	O	AXI read valid. Indicates that required read data is available and 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[(GS_R_SBW_INT-1):0]	O	AXI Sideband signal for read data channel. The signal name changes between the AXI3 and the AXI4 configurations. ■ When the AXI3 interface is enabled, this signal is named rsideband. ■ When the AXI4 interface is enabled, this signal is named ruser. This signal is transferred to the GIF srsideband for write transactions. **Exists:** (GS_HAS_RSB == 1) && (GS_AXI_INTERFACE_TYPE == 0) **Synchronous To:** aclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
ruser[(GS_R_SBW_INT-1):0]	O	AXI User signal for read data channel. The signal name changes between the AXI3 and the AXI4 configurations. ■ When the AXI3 interface is enabled, this signal is named rsideband. ■ When the AXI4 interface is enabled, this signal is named ruser. This signal is transferred to the GIF srsideband for write transactions. **Exists:** (GS_HAS_RSB == 1) && (GS_AXI_INTERFACE_TYPE > 0) **Synchronous To:** aclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A

Table 4-6 AXI Read Data Channel Signals (Continued)

Port Name	I/O	Description
rready	I	AXI read ready. Indicates master can accept 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

4.7 GIF Request Channel Signals

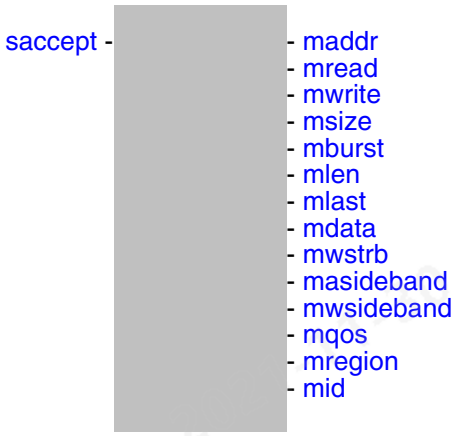


Table 4-7 GIF Request Channel Signals

Port Name	I/O	Description
maddr[(GS_AW-1):0]	O	GIF address. This signal is transferred from the AXI channel signals awaddr and araddr. **Exists:** Always **Synchronous To:** aclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
mread	O	GIF read indicator. Indicates valid read request. **Exists:** Always **Synchronous To:** aclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** High
mwrite	O	GIF write indicator. Indicates valid write request. **Exists:** Always **Synchronous To:** aclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** High

Table 4-7 GIF Request Channel Signals (Continued)

Port Name	I/O	Description
msize[2:0]	O	Size of transfer. This signal is transferred from the AXI channel signals awsize and arsize. **Exists:** Always **Synchronous To:** aclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
mburst[1:0]	O	GIF burst type. This signal is transferred from the AXI channel signals awburst and arburst. **Exists:** Always **Synchronous To:** aclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
mlen[(GS_BW-1):0]	O	GIF burst length. This signal is transferred from the AXI channel signals awlen and arlen. **Exists:** Always **Synchronous To:** aclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
mlast	O	GIF last transfer indicator. This signal is transferred from the AXI write data channel signal wlast. **Exists:** Always **Synchronous To:** aclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** High
mdata[(GS_DW-1):0]	O	GIF write data. This signal is transferred from the AXI write data channel signal wdata. **Exists:** Always **Synchronous To:** aclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A

Table 4-7 GIF Request Channel Signals (Continued)

Port Name	I/O	Description
mwstrb[((GS_DW/8)-1):0]	O	Write byte lane write strobes indicates which byte lanes to update in memory. One write strobe for each eight bits of write data bus. This signal is transferred from the AXI write data channel signal (wstrb). This signal is associated with the following byte of MDATA: MDATA[8n+7: 8n] **Exists:** Always **Synchronous To:** aclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** High
masideband[(GS_A_SBW_INT-1):0]	O	GIF Sideband signal for address. This signal is transferred from the AXI awsideband/arsideband signal. **Exists:** (GS_HAS_AWSB == 1) ||(GS_HAS_ARSB == 1) **Synchronous To:** aclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
mwsideband[(GS_W_SBW_INT-1):0]	O	GIF Sideband signal for write data. This signal is transferred from the AXI wsideband signal. **Exists:** (GS_HAS_WSB == 1) **Synchronous To:** aclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
mqos[3:0]	O	Optional. Quality of Service identifier. This signal conveys priority information associated with each transaction at Address channel of GIF Interface. **Exists:** (GS_AXI_HAS_QOS == 1) && (GS_AXI_INTERFACE_TYPE > 0) **Synchronous To:** aclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
mregion[3:0]	O	Optional. Address Channel Region Identifier on GIF. Permits a single physical interface on a slave to be used for multiple logical interfaces. **Exists:** (GS_AXI_HAS_REGION == 1) && (GS_AXI_INTERFACE_TYPE > 0) **Synchronous To:** aclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A

Table 4-7 GIF Request Channel Signals (Continued)

Port Name	I/O	Description
mid[(GS_ID-1):0]	O	GIF ID. This signal is transferred from the AXI awid or arid signal. **Exists:** (EXTD_GIF == 1) **Synchronous To:** aclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
saccept	I	GIF command accept. Slave accepts current mread or mwrite request by driving saccept high. On saccept high, address and write data are updated by master. Read bursts are single-cycle requests, whereas all other bursts are multi-cycle, requiring saccept to be sampled high for every beat. **Exists:** Always **Synchronous To:** aclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** High

4.8 GIF Response Channel Signals

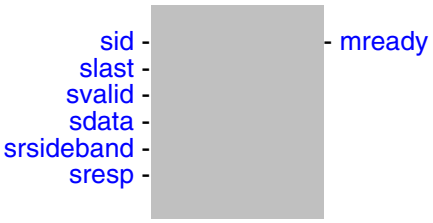


Table 4-8 GIF Response Channel Signals

Port Name	I/O	Description
mready	O	GIF response accept. When master is not ready to accept response, it drives mready low. **Exists:** Always **Synchronous To:** ((EXTD_GIF == 0) && (GS_GIF_LITE == 1)) ? "None" : "aclk" **Registered:** ((GS_DIRECT_AXI_READY == 0) && (GS_GIF_LITE == 0)) ? "Yes" : "No" **Power Domain:** SINGLE_DOMAIN **Active State:** High
sid[(GS_ID-1):0]	I	GIF response ID. The identification tag of the read and write responses. This signal is transferred from the AXI channel signals rid and bid. **Exists:** (EXTD_GIF == 1) **Synchronous To:** aclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
slast	I	GIF last. This signal indicates the last transfer in a read burst from Generic Slave. **Exists:** (EXTD_GIF == 1) **Synchronous To:** aclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** High

Table 4-8 GIF Response Channel Signals (Continued)

Port Name	I/O	Description
svalid	I	GIF valid response indicator. **Exists:** Always **Synchronous To:** ((EXTD_GIF == 0) && (GS_GIF_LITE == 1)) ? "None" : "aclk" **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** High
sdata[(GS_DW-1):0]	I	GIF read data. This signal is transferred to the AXI read data channel signal rdata. **Exists:** Always **Synchronous To:** aclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
srsideband[(GS_R_SBW_INT-1):0]	I	GIF Sideband signal for response. This signal is transferred to the AXI bsideband/rsideband signal. **Exists:** (GS_HAS_RSB == 1) ||(GS_HAS_BSB == 1) **Synchronous To:** aclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
sresp[1:0]	I	GIF response signal. Encoding is as follows: ■ h'0 OK_R: Read transaction finished successfully. ■ h'1 OK_W: Write transaction finished successfully. ■ h'2 SLVERR_R: Addressed slave generated error for read access. ■ h'3 SLVERR_W: Addressed slave generated an error for write access. **Exists:** Always **Synchronous To:** ((EXTD_GIF == 0) && (GS_GIF_LITE == 1)) ? "None" : "aclk" **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A

4.9 AXI Low Power Interface Signals

Table 4-9 AXI Low Power Interface Signals

Port Name	I/O	Description
csysack	O	Low power request acknowledgement. ■ De-asserted by DW_axi_gs to acknowledge request to enter low-power state. ■ Asserted by DW_axi_gs to acknowledge request to exit low-power state. **Exists:** (GS_LOWPWR_HS_IF == 1) **Synchronous To:** aclk **Registered:** Yes **Power Domain:** SINGLE_DOMAIN **Active State:** High
cactive	O	Clock active request. De-asserted by DW_axi_gs to inform the system low-power controller (LPC) that the clock can be removed. ■ 0: Peripheral clock not required. ■ 1: Peripheral clock required. **Note:** Clock should be removed on positive edge after cactive and csysack signals are sampled low (0). **Exists:** (GS_LOWPWR_HS_IF == 1) **Synchronous To:** aclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** High
csysreq	I	System low-power request from system clock controller. ■ De-asserted by system low power controller (LPC) to initiate entry into a low power state. ■ Asserted to initiate exit from a low power state. **Exists:** (GS_LOWPWR_HS_IF == 1) **Synchronous To:** aclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** High

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
GS_A_SBW_INT	=((GS_HAS_ARSB || GS_HAS_AWSB == 1) ? GS_A_SBW : 0)
GS_MAX_SBW	256
GS_R_SBW_INT	=((GS_HAS_RSB || GS_HAS_BSB == 1) ? GS_R_SBW : 0)
GS_W_SBW_INT	=((GS_HAS_WSB == 1) ? GS_W_SBW : 0)

6 Verification

This chapter provides an overview of the testbench available for DW_axi_gs verification. Once you have configured the DW_axi_gs in coreConsultant and have set up the verification environment, you can automatically run simulations.

Attention	These tests are provided only to verify the your specific configuration and to provide reference waveforms so that you can see how sample transactions affect input/output signal values. The tests cannot be modified, and the testbench cannot be reused as a starting point for a custom testbench.

6.1 Overview of Vera Tests

The DW_axi_gs verification testbench can perform the following tests:

- test_random
- test_exclusive
- test_extd_gif

For each test executed, the run script writes all simulation output files to the respective *workspace*/sim/test_*name* directory. The runtest.log file in *workspace*/sim includes all simulation results. All of these files are visible from the Reports tab of the "Setup and Run Simulations" activity in the GUI, or they can be viewed directly from the text files in the *workspace*/sim directory.

6.1.1 test_random

This test verifies all system features, except exclusive access transactions, using randomly generated transactions. This test generates 10,000 AXI transactions and verifies all responses. Although the test uses random transactions, the random seed is statically determined from the current testbench and DUT setup. If neither testbench nor DUT are changed, the test runs exactly the same in consecutive simulations.

6.1.2 test_exclusive

This test verifies exclusive transactions using randomly generated instructions. This test generates 10,000 mixed AXI exclusive and normal access transactions and verifies all responses. Although the test uses random transactions, the random seed is statically determined from the current testbench and DUT setup. If neither testbench nor DUT are changed, the test runs exactly the same in consecutive simulations.

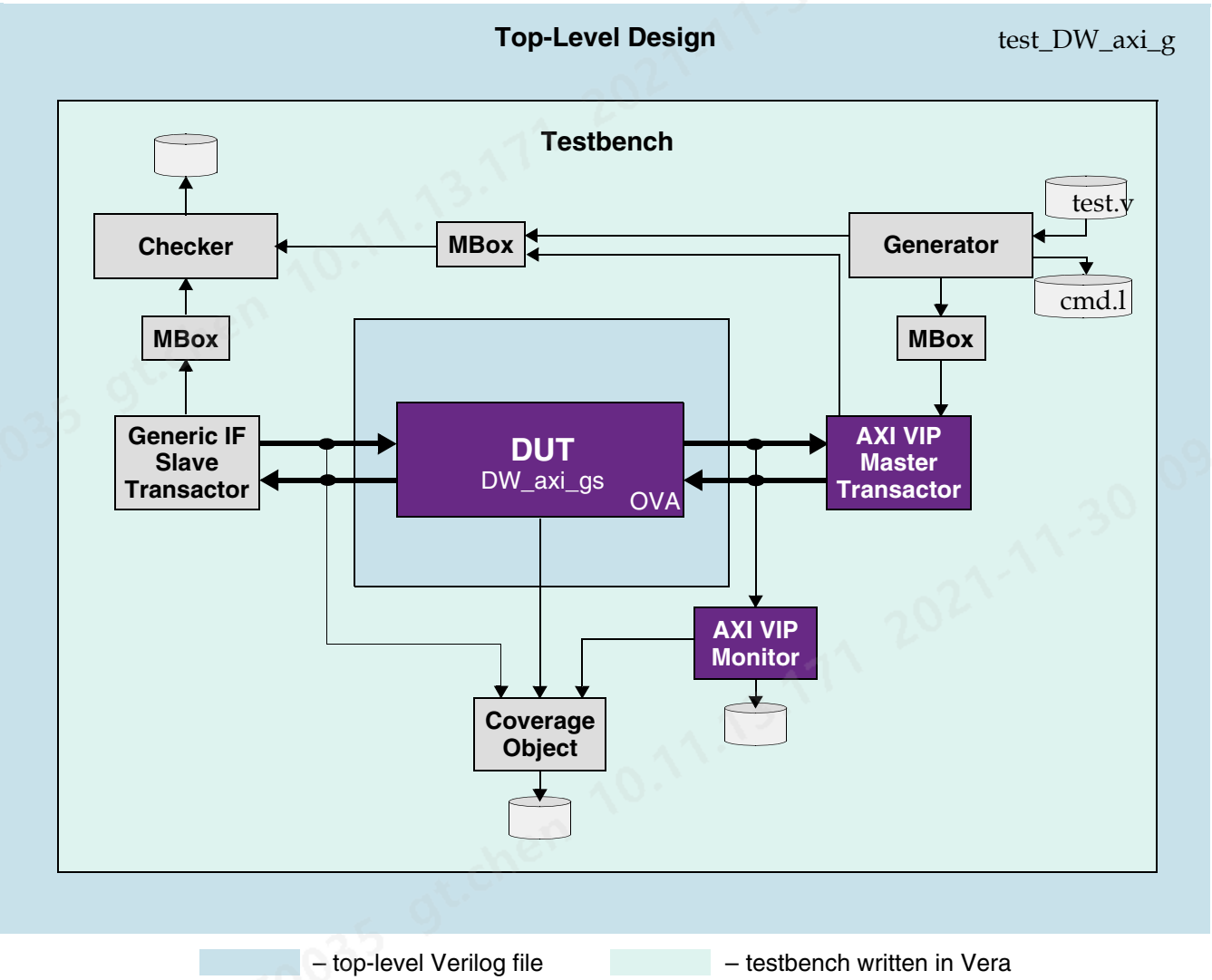
6.1.3 test_extd_gif

This test verifies ID behavior using randomly generated transactions in Extended GIF interface mode. The traffic is randomly generated with different/same ID and the DUT behavior is verified against the same on the generic interface.

6.2 Overview of DW_axi_gs Testbench

The following diagram illustrates the DW_axi_gs testbench, which applies random constraint testing and uses assertions in the DUT.

Figure 6-1 DW_axi_gs Testbench



The following define the testbench components:

- Generator – Produces stimulus to the design. Creates constrained random combinations of burst and single instructions, as well as specified instructions for directed tests.
- Transactors – Comprised of two transactors: Generic Interface (GIF) transactor and AXI VIP slave.

The GIF transactor is a directed transactor that converts stimulus from the generator into binary vectors and drives it into the DUT. The GIF transactor also implements temporal aspects, such as rate of transactions.

The AXI VIP slave is an automatic transactor that reacts to AXI transactions. The slave is programmable and keeps record of the incoming executions in a local buffer, which can be accessed by the checker.

- Checker – Reads the AXI VIP buffer and compares the results with the expected queue; also known as the scoreboard.
- Mail Box – Used to pass information on the fly.
- Coverage Object – Records all coverage points that are hit by the testcases. It can extract coverage information directly from the AXI or from the OVA statements in the source code. The log file allows determination of the coverage grade of the DUT.

Most transaction testing is done with the DUT acting as a black box, while some tests determine specific implementation details. These tests use assertions, which can also be used to collect functional coverage information.

Results are checked using a predictive approach; that is, the reference is derived from the generated stimulus. No reference models for the DUT are used.

Each transaction generates two complete logs. The GIF master transactor logs the entire transaction, including response, and sends it to the checker once the transaction completes. The AXI VIP slave has watchpoints enabled, which are triggered once a transaction completes. The watchpoints also log the complete transaction, including response, and send it to the checker. The checker can then compare the originating and resulting portions of each transaction.

For example, the request phase for a burst originates from the GIF master, and the resulting request is logged on the AXI interface. For read data the situation is reverse. The read data originates from the AXI slave, and the resulting data is logged on the GIF.

If DW_axi_gs is configured for AXI4 mode, the testbench is automatically capable of supporting AXI4 traffic using AMBA VMT VIPs.

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_gs.

7.1.1 Power Consumption, Frequency, Area and DFT Coverage

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

Table 7-1 Synthesis Results for DW_axi_gs

Configuration	Operating Frequency	Gate Count	Power Consumption		Tetramax (%)		SpyGlass StuckAtCov (%)
			Static Power	Dynamic Power	StuckAt Test	Transition	
Default Configuration	400 MHz	3363	3nW	0.136 mW	99.7	98.86	99.5

Table 7-1 Synthesis Results for DW_axi_gs

Configuration	Operating Frequency	Gate Count	Power Consumption		Tetramax (%)		SpyGlass StuckAtCov (%)
			Static Power	Dynamic Power	StuckAt Test	Transition	
Typical Configuration 1 GS_AW = 64 GS_AXI_EX_ACCESS = 16 GS_AXI_INTERFACE_TYPE= 0 GS_BID_BUFFER = 4 GS_BW = 8 GS_DIRECT_AXI_READY = 1 GS_DIRECT_GIF_READY = 1 GS_DW = 8 GS_GIF_LITE = 0 GS_ID = 13 GS_LOWPWR_HS_IF = 0 GS_LOWPWR_LEGACY_IF = 0 GS_REQ_BUFFER = 4 GS_RESP_BUFFER = 4 GS_RID_BUFFER = 4 GS_WDATA_BUFFER = 4	400 MHz	22229	20 nW	0.867 mW	99.95	99.91	100
Typical Configuration 2 GM_AW = 64 GM_AXI_HAS_QOS = 1 GM_AXI_INTERFACE_TYPE = 1 GM_BLOCK_READ = 1 GM_BLOCK_WRITE = 0 GM_DIRECT_AXI_READY = 1 GM_DIRECT_GIF_READY = 1 GM_DW = 512 GM_LOWPWR_HS_IF = 1 GM_REQ_BUFFER = 4 GM_RESP_BUFFER = 4 GM_WDATA_BUFFER = 4	400 MHz	22245	20 nW	0.871 mW	99.95	99.92	100

8 Basic Core Module (BCM) Library

The Basic Core Module (BCM) Library is a library of commonly used blocks for the Synopsys DesignWare IP development. These BCMs are configurable on an instance-by-instance basis and, for the majority of BCM designs, there is an equivalent (or nearly equivalent) DesignWare Building Block (DWBB) component.

This chapter provides more information about the BCMs used in DW_axi_gs.

8.1 BCM Library Components

Table 8-1 describes the list of BCM library components used in DW_axi_gs.

Table 8-1 List of BCM Library Components used in the Design

BCM Module Name	BCM Description	DWBB Equivalent
DW_axi_gs_bcm06	Synchronous (Single Clock) FIFO Controller with Dynamic Flags	DW_fifoctl_s1_df
DW_axi_gs_bcm57	Synchronous Write-Port, Asynchronous. Read-Port RAM (Flip-Flop-Based)	DW_ram_r_w_s_dff
DW_axi_gs_bcm65	Synchronous (Single Clock) FIFO with Static Flags	DW_fifo_s1_sf

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

A

active command queue
definition 83
application design
definition 83

B

BFM
definition 83
big-endian
definition 83
blocked command stream
definition 83
blocking command
definition 83

C

command channel
definition 83
command stream
definition 83
component
definition 83
configuration
definition 83
configuration intent
definition 83
Customer Support 8
cycle command
definition 83

D

decoder
definition 84
design context
definition 84
design creation
definition 84
DesignWare Library
definition 84
dual role device
definition 84

E

endian
definition 84

F

Full-Functional Mode
definition 84

G

GPIO
definition 84
GTECH
definition 84

H

hard IP
definition 84
HDL
definition 84

I

IIP
definition 84
implementation view
definition 84
instantiate
definition 84
interface
definition 84
IP
definition 84

L

little-endian
definition 84

M

master
definition 84
model
definition 84
monitor
definition 84

N

non-blocking command
definition 84

P

peripheral
definition 85

R

RTL
definition 85

S

SDRAM
definition 85
SDRAM controller
definition 85
slave
definition 85
SoC
definition 85
SoC Platform
AHB contained in 11
APB, contained in 11
AXI contained in 7
defined 7, 11
soft IP
definition 85
sparse data
definition 85
static controller
definition 85
synthesis intent
definition 85
synthesizable IP
definition 85

T

technology-independent
definition 85
Testsuite Regression Environment (TRE)
definition 85
TRE
definition 85

V

VIP
definition 85

W

wrap
definition 85
wrapper
definition 85

Z

zero-cycle command
definition 85