SYNOPSYS®

DesignWare® DW_ahb_icm

Databook

DW_ahb_icm – Product Code

1.18a
December 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 9
DatabookOrganization 9
Related Documentation 9
Web Resources 10
Customer Support 10
Product Code 11

Chapter 1
Product Overview 13
1.1 DesignWare System Overview 13
1.2 General Product Description 15
1.2.1 DW_ahb_icm Block Diagram 15
1.3 Features 15
1.4 Standards Compliance 16
1.5 Verification Environment Overview 16
1.6 Licenses 17
1.7 Where To Go From Here 17

Chapter 2
Functional Description 19
2.1 Input Stage 20
2.2 Arbitration Scheme and Layer Selection 21
2.2.1 Fixed-Priority Arbitration Scheme 21
2.2.2 Dynamic-Priority Arbitration Scheme 22
2.3 Slave to Layer Data and Response Return 22
2.3.1 Data Generation 22
2.3.2 Slave Response Generation 22
2.4 Slave Return from Split Generation 23
2.5 Non-Standard Master ID Sideband Signals 23
2.6 Timing 24
2.6.1 Layer Arbitration 24
2.6.2 RETRY Generation 25
2.6.3 Mask Priority or Arbitration 26
2.7 AMBA 5 AHB Features 30
2.7.1 Extended Memory Types 30
2.7.2 Secure Transfers 30
2.7.3 Exclusive Transfers 30
2.7.4 Multiple Slave Select 31

2.7.5 User Signals 31

Chapter 3
Parameter Descriptions 33
3.1 Top Level Parameters 34
3.2 Top Level Parameters / DW_ahb_icm Source Code Configuration Parameters 39
3.3 AMBA 5 AHB Configuration Parameters 40
3.4 User Signal Configuration Parameters 41

Chapter 4
Signal Descriptions 43
4.1 Clock and Reset Interface Signals 45
4.2 Arbitration Control Signals 46
4.3 Slave Layer Interface Signals 47
4.4 Master Layer x Interface Signals 52

Chapter 5
Verification 57
5.1 Overview of Vera Tests 57
5.1.1 Access 57
5.1.2 Priority Scheme 58
5.1.3 Locked Transfers 58
5.1.4 Responses 58
5.2 DW_ahb_icm Testbench 58

Chapter 6
Integration Considerations 61
6.1 Performance 62
6.1.1 Power Consumption, Frequency, and Area and DFT Coverage 62

Chapter A
Internal Parameter Descriptions 65

Appendix B
Basic Core Module (BCM) Library 67
B.1 BCM Library Components 67

Appendix C
Glossary 69

Revision History

This table shows the revision history for the databook from release to release. This is being tracked from version 1.10b onward.

Version	Date	Description
1.18a	December 2020	**Added:** ■ "AMBA 5 AHB Features" on page 30 ■ Appendix B, "Basic Core Module (BCM) Library" **Updated:** ■ Image version changed for 2020.12a release ■ "Features" on page 15 ■ "Performance" on page 62 ■ "Parameter Descriptions" on page 33, "Signal Descriptions" on page 43, and "Internal Parameter Descriptions" on page 65 are auto-extracted with change bars from the RTL **Removed:** ■ Index chapter
1.17a	July 2018	**Updated:** ■ Image version changed for 2018.07a release ■ "Performance" on page 62 ■ "Parameter Descriptions" on page 33, "Signal Descriptions" on page 43, and "Internal Parameter Descriptions" on page 65 are auto-extracted with change bars from the RTL **Removed:** ■ Chapter 2, "Building and Verifying a Component or Subsystem" and added the contents in the newly created user guide.

Version	Date	Description
1.16a	October 2016	■ Version changed for 2016.10a release ■ "Parameter Descriptions" on page 33 auto-extracted from the RTL ■ Moved "Internal Parameter Descriptions" to Appendix. ■ Removed "Running Leda on Generated Code with coreConsultant" and reference to Leda directory in Table 2-1 ■ Removed "Running Leda on Generated Code with coreAssembler" section, and reference to Leda directory in Table 2-4 ■ Updated "Features" on page 15 ■ Updated "Parameter Descriptions" and "Signals" chapters ■ Added "mid_lx" and "mid" signals in "Signal Descriptions" on page 43 ■ Added "MID_WIDTH" parameter in "Parameter Descriptions" on page 33. ■ Updated maximum configuration area and power numbers, and added an entry for "Maximum Configuration (with 16 AHB Layers) for area and power, in "Integration Considerations" on page 61 ■ Added a note in "Arbitration Scheme and Layer Selection" on page 21 ■ Added "Non-Standard Master ID Sideband Signals" on page 23 ■ Added an entry for the xprop directory in Table 2-1 and Table 2-4. ■ Added "Running VCS XPROP Analyzer"
1.15a	June 2015	■ Added "Mask Priority or Arbitration" on page 26 ■ Added "Running SpyGlass® Lint and SpyGlass® CDC" ■ Added "Running SpyGlass on Generated Code with coreAssembler" ■ "Signal Descriptions" on page 43 auto-extracted from the RTL ■ Added Chapter A, "Internal Parameter Descriptions"
1.14a	June 2014	■ Version change for 2014.06a release ■ Added "Integration Considerations" chapter ■ Corrected External Input/Output Delays in Signals chapter
1.13d	May 2013	■ Version change for 2013.05a release ■ Updated the template.
1.13c	October 2012	Added the product code on the cover and in Table 1-1.
1.13c	March 2012	■ Updated description of icm_priority signal ■ Added a dynamic-priority arbitration subsection
1.13b	October 2011	Version change for 2011.10a release

Version	Date	Description
1.13a	June 2011	■ Updated system diagram in Figure 1-1 ■ Enhanced “Related Documents” section in Preface
1.13a	September 2010	Corrected names of include files and vcs command used for simulation.
1.12a	December 2009	Updated databook to new template for consistency with other IIP/VIP/PHY databooks
1.12a	May 2009	Removed references to QuickStarts, as they are no longer supported
1.12a	April 2009	Corrected dependency value of AHB_LITE to 1 in Parameter Descriptions
1.12a	October 2008	Version change for 2008.10a release
1.10c	June 2008	Version change for 2008.06a release
1.10b	March 2008	Changed from 1-4 layers to 1-8 layers for fixed-priority arbitration scheme
1.10b	June 2007	Version change for 2007.06a release

Preface

This databook provides information about the DesignWare® AHB Multi-layer Interconnection Matrix (ICM) peripheral, referred to as the DW_ahb_icm throughout the remainder of this databook. The DW_ahb_icm conforms to the *AMBA Specification, Revision 2.0* from Arm®.

The information in this databook includes a functional description, signal and parameter descriptions, programming interface, and detailed information about the operation of the DW_ahb_icm. Also provided are an overview of the component testbench, a description of the tests that are run to verify the DW_ahb_icm, and synthesis information for the component.

DatabookOrganization

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_ahb_icm.
- Chapter 3, "Parameter Descriptions" identifies the configurable parameters supported by the DW_ahb_icm.
- Chapter 4, "Signal Descriptions" provides a list and description of the DW_ahb_icm signals.
- Chapter 5, "Verification" provides information on verifying the configured DW_ahb_icm.
- Chapter 6, "Integration Considerations" provides performance hardware configuration parameters that affect the performance of the DW_ahb_icm.
- Appendix A, "Internal Parameter Descriptions" provides a list of internal parameter descriptions that might be indirectly referenced in expressions in the Signals chapter.
- Appendix B, "Basic Core Module (BCM) Library" provides information about BCM used in DW_ahb_icm.
- Appendix C, "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 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 2, see the *Guide to Documentation for DesignWare Synthesizable Components for AMBA 2 and AMBA 3 AXI (Documentation Overview)*.

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:** AMBA

  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
DW_axi_a2x	Configurable bridge between AXI and AHB components or AXI and AXI components.
DW_axi_gm	Simplify the connection of third party/custom master controllers to any AMBA 3 AXI fabric
DW_axi_gs	Simplify the connection of third party/custom slave controllers to any AMBA 3 AXI fabric
DW_axi_hmx	Configurable high performance interface from and AHB master to an AXI slave
DW_axi_rs	Configurable standalone pipelining stage for AMBA 3 AXI subsystems
DW_axi_x2h	Bridge from AMBA 3 AXI to AMBA 2.0 AHB, enabling easy integration of legacy AHB designs with newer AXI systems

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

Component Name	Description
DW_axi_x2p	High performance, low latency interconnect fabric and bridge for AMBA 2 & 3 APB for direct connect to AMBA 3 AXI fabric
DW_axi_x2x	Flexible bridge between multiple AMBA 3 AXI components or buses

1 Product Overview

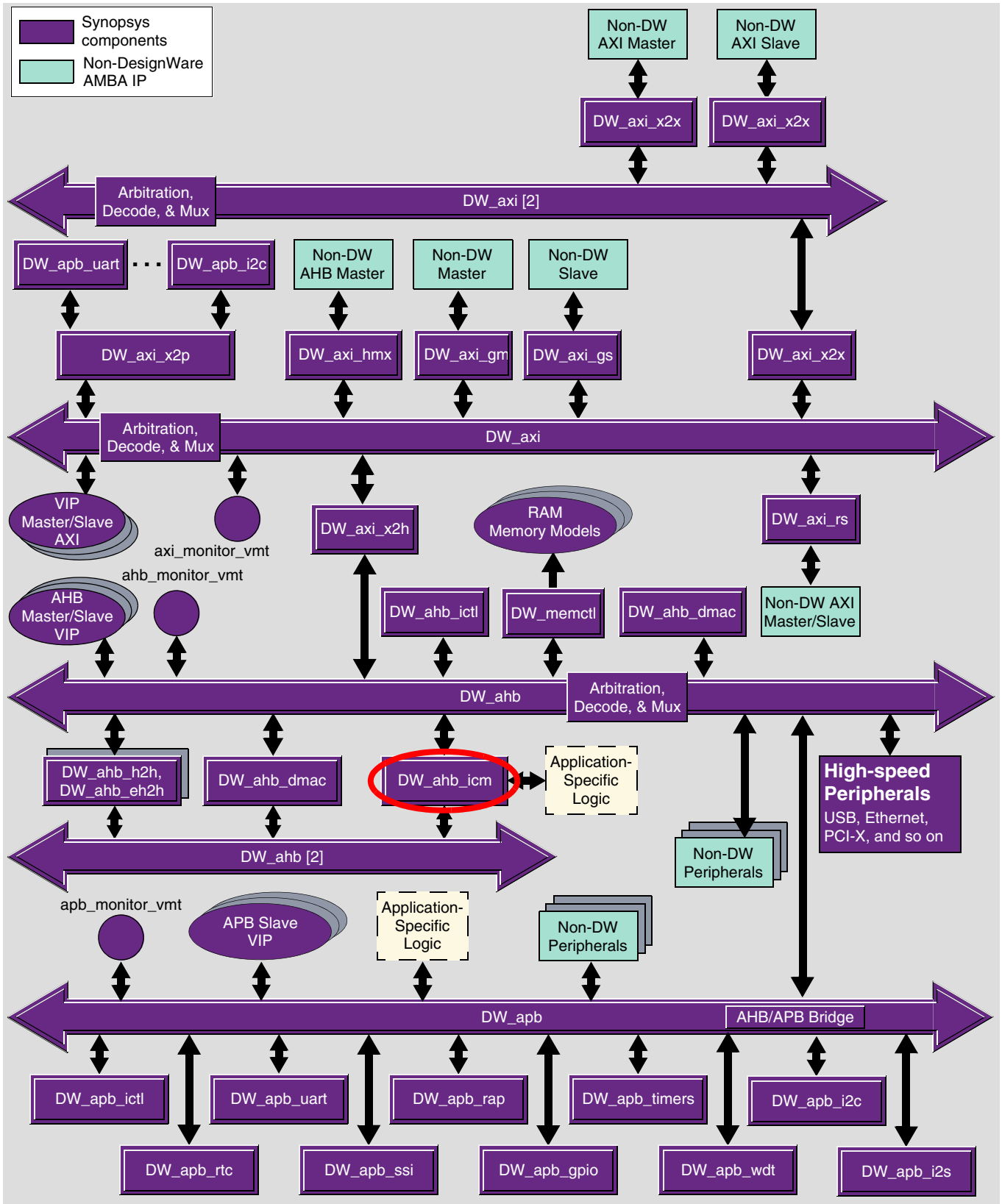
The DW_ahb_icm is a programmable Multi-layer Interconnection Matrix (ICM) peripheral. This component is part of the family of DesignWare Synthesizable Components.

1.1 DesignWare 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 version 3.0-compliant 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. To access the product page and documentation for AMBA components, see the DesignWare IP Solutions for AMBA Interconnect page. (SolvNetPlus ID required)

Figure 1-1 Example of DW_ahb_icm in a Complete System



You can connect, configure, synthesize, and verify the DW_ahb_icm 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_ahb_icm component, you might prefer to use coreConsultant, documentation for which is available in the *coreConsultant User Guide.*

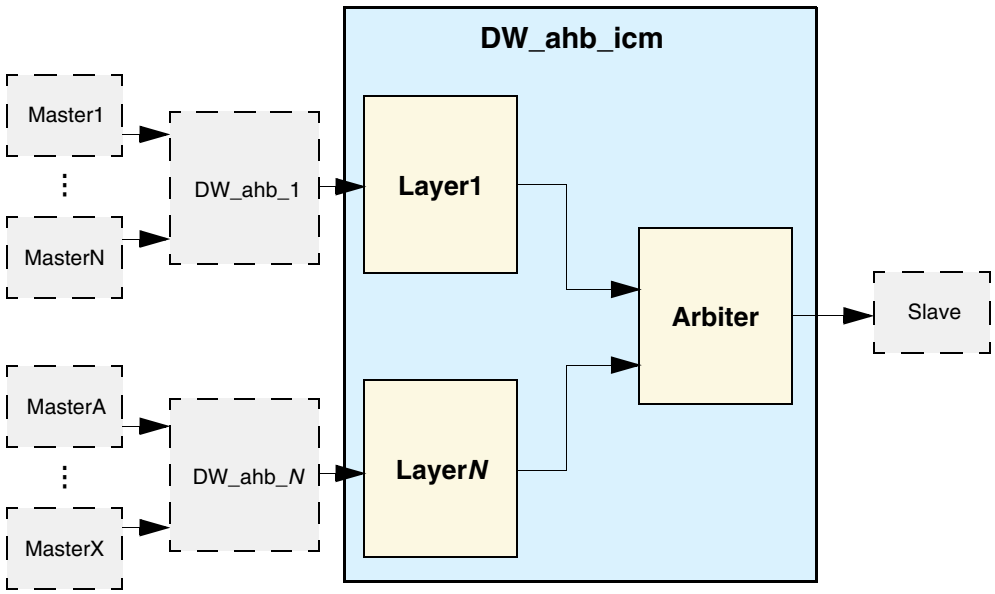
1.2 General Product Description

The Synopsys DW_ahb_icm is a configurable multi-layer interconnection matrix. The DW_ahb_icm is neither an AHB slave nor an AHB master, but facilitates multiple layers to access a slave; there can be only one slave for each DW_ahb_icm.

1.2.1 DW_ahb_icm Block Diagram

Figure 1-2 shows the DW_ahb_icm block diagram.

Figure 1-2 DW_ahb_icm Block Diagram



1.3 Features

DW_ahb_icm has the following features:

- Layer arbitration
- Input stage address and control holding registers for each layer; holding registers keep address and control information presented on bus—they are removed the following cycle
- Mapping of slave response onto the layer that initiated the transfer
- Return of split transaction read data onto layer that initiated transfer
- Common clock and reset shared among all layers; common clock implies that all AHB layers use same clock frequency

- Configuration of DesignWare AHB Lite system
- Support for non-standard Master ID sideband signal (in AHB-Lite mode)
- User-defined parameters:
  - AMBA Lite
  - AHB address bus width (same width on all layers)
  - AHB data bus width (same width on each layer)
  - AHB master layers (up to 16)
  - Split- or non-split-capable slave
  - Slave with or without multiple select lines
  - Slave with or without protection control
  - Slave with or without burst control
  - Slave with or without lock control
  - Layer release scheme
  - Baseline arbitration scheme
  - External arbitration priority control
- AMBA 5 AHB support
  - Extended memory types
  - Secure transfers
  - Exclusive transfers
  - Multiple slave select feature support
  - User signals on each channel

Some uses of the DW_ahb_icm are:

- An add-on interface for AHB slaves to allow them to be used in a multi-layer system
- Support for AHB master to operate in AMBA Lite mode within multi-layer system

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

1.4 Standards Compliance

The DW_ahb_icm component conforms to the *AMBA Specification, Revision 2.0* from Arm®. Readers are assumed to be familiar with this specification.

1.5 Verification Environment Overview

The DW_ahb_icm 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" on page 57 section discusses the specific procedures for verifying the DW_ahb_icm.

1.6 Licenses

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

1.7 Where To Go From Here

At this point, you may want to get started working with the DW_ahb_icm 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_ahb_icm component, see "Overview of the coreConsultant Configuration and Integration Process" in *DesignWare Synthesizable Components for AMBA 2 User Guide*.

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

2 Functional Description

This chapter describes the functional operation of the DW_ahb_icm and describes the components and functionality of the DW_ahb_icm.

There is an option to configure AHB Lite, which is the DesignWare implementation of AMBA 2.0 AHB-Lite. The DesignWare AHB Lite configuration,

- Does not include requesting/granting protocols to the arbiter and split/retry responses from the slaves; all slaves are made non-split capable
- Does not include arbiter as the signals associated with the component are not used: hbusreq and hgrant
- Does not include write data, address, or control multiplexers
- Pause mode is not enabled
- Default master number is changed to 1
- Number of masters is changed to 1

For more information about AHB Lite, see the *DesignWare DW_ahb Databook*.

Figure 2-1 illustrates the block diagram of the DW_ahb_icm.

Figure 2-1 DW_ahb_icm Block Diagram

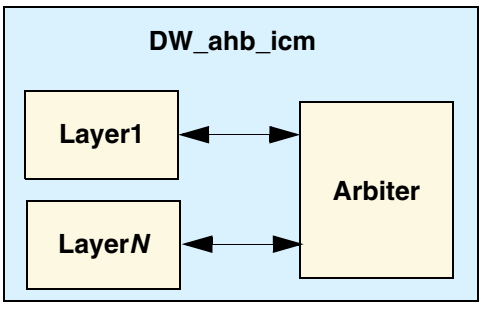
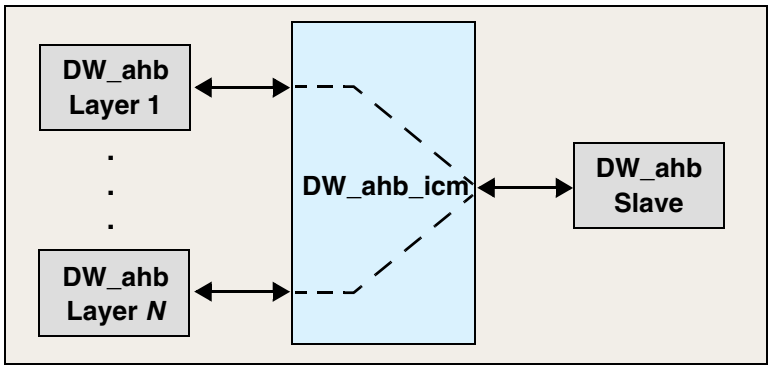


Figure 2-2 illustrates how the DW_ahb_icm might interface between the masters and the slave.

Figure 2-2 Simple DW_ahb_icm Interface



2.1 Input Stage

A layer is referred to as one or more masters that compete together with one master winning ownership.

When there is more than one layer looking for access to the slave at the same time, this is referred to as a clash of requests. Whenever a clash is detected, only one layer can gain access to the slave. The layers that do not gain access to the slave need to have their address and control signals stored into their input stage. When address and control signals are stored into an input stage, then the stored transfer controls the request and lock generation circuitry. When a lower priority layer is in the middle of a burst transfer and a higher priority layer issues a transfer, the higher priority layer is stored and then held off until the lower priority layer completes the transfer.

When the address and control is presented to a slave with hready = 1, the slave must accept the address. The hready comes from the previously addressed slave on the layer through hready_l(x), where x is from 1 to 16, and is an indication of whether the last transfer has completed on the corresponding layer.

Layers can hand over accessing the slave to another layer. With different masters capable of accessing the same slave, the slave can indicate that it has completed the transfer with a master on one layer while it is being accessed by a master on another layer or is in the middle of an access. Therefore, the address and control—and associated address decode (the select)—from the layer requesting the slave need to be stored and the layer held off until after the current master has completed its transfer. The stored transfer can then be passed to the slave and the layer released to generate the next address.

Whenever there is address and control stored in the input stage, the address and control passed to the slave is always from the input stage. The slave can have multiple selects lines—up to eight per layer—but only one set of read data and responses come from the slave, which holds off a master with hready_resp held low. The master then waits with its next address until hready_resp goes high, at which time it determines if hresp was OK in the previous cycle before continuing with the transfer; if not the master cancels the transfer.

2.2 Arbitration Scheme and Layer Selection

The DW_ahb_icm can use either a fixed-priority or a dynamic-priority arbitration scheme between the master layers.

Attention

Priority schemes for arbitration are opposite in DW_ahb and DW_ahb_icm:

- In DW_ahb, higher the priority value, higher the priority of the Master.
- In DW_ahb_icm, lower the priority value, higher the priority of the layer.

2.2.1 Fixed-Priority Arbitration Scheme

In the fixed-priority scheme, each layer is assigned a unique priority. The order of priorities is described by the index of the layer number to which each layer is connected. For example (with ICM_NUM_LAYERS = 16):

- Layer 1 is the highest priority
- Layer 16 is the lowest priority

Each layer generates a select for the slave, hsel_s(y)_l(x). A slave may have multiple select lines, in which case *y* is from 1 to the number of selects. All the slave selects from a layer are combined to generate a single select from each layer. With a single select for the slave, the arbitration reduces to being between one of eight layers. Once a transfer is stored in the input stage, the input stage select is used instead of the layer select in order to generate the index for which layer wins the arbitration.

The arbitration is carried out at the beginning of every burst transfer with the highest priority layer gaining access when two or more layers want to start a new transfer at the same time. The arbitration is changed only when htrans is IDLE or NON-SEQUENTIAL. If the arbitration is on every beat of a transfer, then slaves that use the burst information are compromised, since it is possible that bursts are broken, which violates the protocol. If a slave does not support burst transfers—by not using the hburst control signal—then the arbitration is carried out on every transfer (as the slave does not use the burst type signal).

Each AHB master has the capability of generating locked transfers by asserting hlock, which is translated into hmastlock by the bus arbiter. The hmastlock signal maintains the current layer as the active layer. This allows a sequence of idle, single, or burst transfers from a master to be generated for the slave without having them separated by another layer gaining access to the slave. Whenever a layer generates locked transfers to a slave and starts accessing the slave, no other layer gains access until hmastlock goes low.

A slave access is always reduced to a single request, regardless of the number of select lines that it may have.

A two-tiered arbiter is used to generate the layer arbitration for requests and locks. The two-tiered arbiter is configured with fixed priorities for each layer, and none of the layers are masked. The result of the arbitration is an index of the layer that has won access to the slave. The index selects the correct address and control signals to pass to the slave. The address and control may come directly from the layer or from an input stage.

A master is stopped from generating any more transfers until the slave accepts the stored transfer from the input stage. Only one set of address and control signals from a layer is accepted into the input stage.

2.2.2 Dynamic-Priority Arbitration Scheme

The DW_ahb_icm supports dynamically changing the arbitration priority using the icm_priority signals.

For example, you could create an external round-robin-type arbiter, and connect it to the icm_priority bus to dynamically change the priority. The width of the icm_priority bus is determined by the number of layers. For more details, see the Signal Descriptions description on page 43.

1. Select "true" for External Priority Control? in coreConsultant. This sets the parameter ICM_HAS_XPRIORITY to 1.
2. Create the circuit for an external round-robin arbitration function.
3. Connect the output signal of the round-robin arbiter to the icm_priority signals of the DW_ahb_icm.v file.

Note Priority can be changed at any time. The arbiter considers the priority for the respective layer when the transfer type HTRANS=NSEQ. Except dynamic priority selection on NSEQ, all other aspects are same as described for fixed priority scheme in "Fixed-Priority Arbitration Scheme" on page 21.

2.3 Slave to Layer Data and Response Return

The following sections discuss how data and slave response generation relates to layers.

2.3.1 Data Generation

For write data, the layer index that supplied the address and control signals is stored each time the slave is accessed. This is done so that the data to the slave can be generated from selecting the layer that initiated the transfer in order to supply the write data. The matrix does not store the write data.

For read data, the same data is passed back to all layers. When multiple layers try to read the same slave, only one layer has an active hready_resp_l(x). The master ignores the hrdata_lx signal, unless the associated hready_resp_lx for the same layer is active.

2.3.2 Slave Response Generation

The slave response is passed back to a master on hready_resp_lx in order to indicate that the current transfer has completed. Only when a layer accesses a slave is its response passed back to that layer. The stored index of the layer that generated the address and control signals is used to indicate to which layer the response is returned.

A layer may be ready to start a transfer to the slave after it has completed a transfer to another slave. However, the slave that must be accessed through the DW_ahb_icm may not be ready because it is or was accessed by another layer. In this case, the transfer is accepted into the input-stage, and holding hready_resp_lx low holds off the layer until the slave is ready. At all other times, the slave response comes directly from the slave.

If the layer is held off because of its priority or because another layer accesses the slave, then the DW_ahb_icm can return a RETRY response to the layer. The RETRY is returned if the matrix calculates that the layer has been held off for more than the configured limit of cycles (ICM_RLIMIT_Lx) for that layer. Generating a RETRY because a layer is held for too long is a user-configurable option, (ICM_RELEASE_Lx),

which requires the hready_resp_lx signal for the layer to be brought high. The layer that is retried can then give a higher priority master access to its layer before retrying the access to the shared slave. The stored address and control in the input stage is cancelled. At all other times, the DW_ahb_icm sends back an OKAY response to the layer.

2.4 Slave Return from Split Generation

In a multi-master, single-layer AHB bus configuration, a slave capable of split transaction read data (split) uses hmaster to calculate which master was driving the address and control signals when the transfer was split. The slave pulses a 1 on the hmaster bit of its hsplit bus to allow the split master back into the arbitration scheme.

With multiple layers, there may be many masters on many layers with the same master number. Therefore, when a slave returns an hsplit, it needs to be passed back to the layer that initiated the transfer. This ensures that masters on other layers are not returned from splits because they have the same master number as other masters.

When a slave is split, it sets a bit on its hsplit bus that returns the master from a split. The DW_ahb_icm keeps track of which masters on which layers are split by the common slave, which allows only the correct master on the layer that initiated the transfer to be returned from a split. The slave has no concept of a layer, but of a master number. When the slave indicates that a master should be returned, it can be on any layer. The DW_ahb_icm keeps track of which master on which layer receives a split from the common slave. This allows the master on a layer that receives a split from a slave other than the common slave to not be unmasked from the return-from-split generated by the common slave. If the same master number is split on multiple layers at the same time, then the common slave returns all split layers from the split at the same time.

2.5 Non-Standard Master ID Sideband Signals

The Non-Standard Master ID Sideband signals are available only in the AHB-Lite mode when the MID_WIDTH parameter set to a non zero value. These signals enable the propagation of the ID through DW_ahb_icm. The timing of these signals is similar to other AHB control signals, such as hsize and hprot.

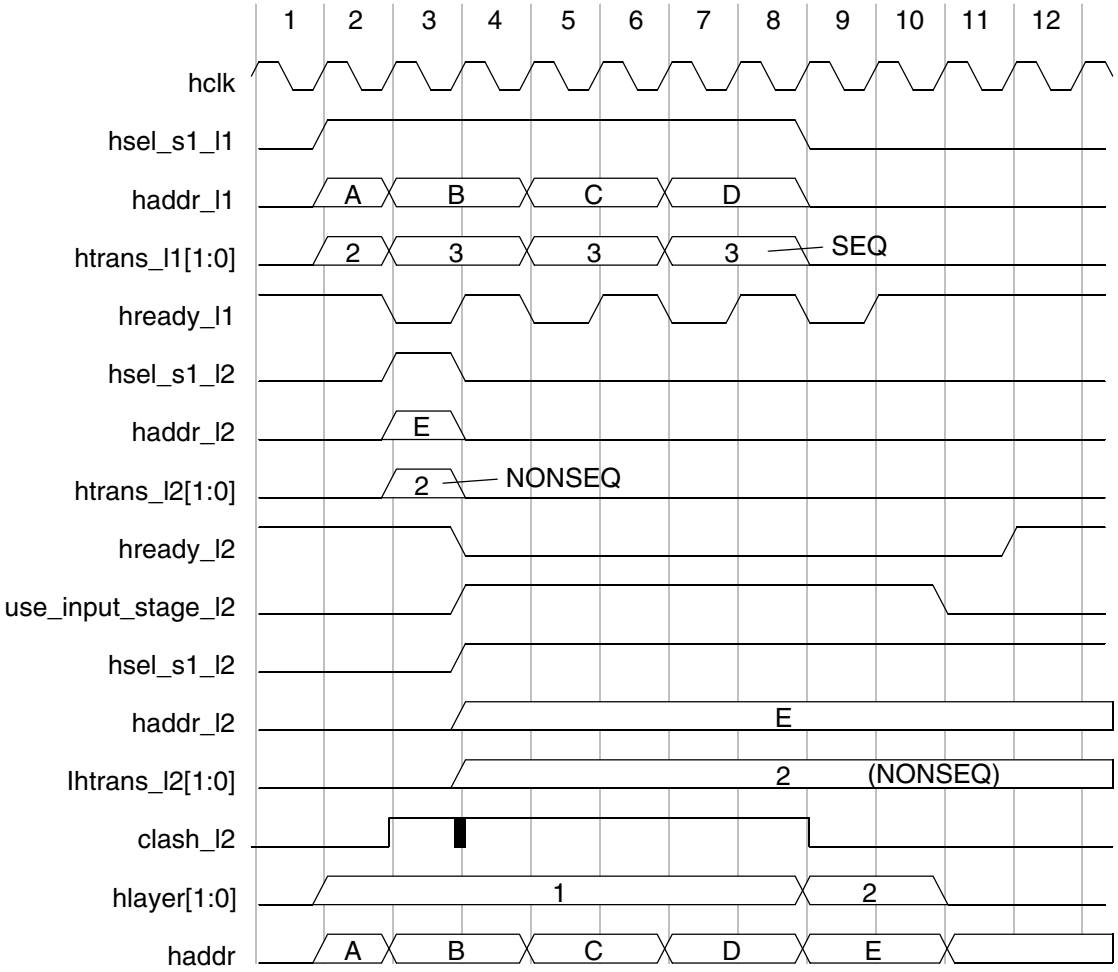
2.6 Timing

The following sections discuss DW_ahb_icm timing.

2.6.1 Layer Arbitration

Figure 2-3 shows the result of two layers trying to access the same slave. Layer 1 wants to complete a burst of transfers, and Layer 2 wants to complete a single transfer.

Figure 2-3 Layer Arbitration



The slave inserts a wait state for each transfer. In the first cycle, there is no clash of requests, and Layer 1 gains control. The arbitration is locked until a new transfer starts or until there is no activity on the bus. Layer 2 inserts its transfer, which is accepted into the input stage by the DW_ahb_icm and is then prevented from generating any other traffic because its hready_l2 is brought low. The transfer is placed into the input stage for Layer 2. As long as there is data in the input stage, the master on Layer 2 is prevented from generating any subsequent transfers.

When the burst transfer from Layer 1 is completed, the ownership of the DW_ahb_icm is swapped to Layer 2. The arbiter sees an idle transfer from Layer 1 (clock cycle 9) so that it can transfer ownership. The address and control for Layer 2 are then taken from the input stage. Only when the transfer is accepted is

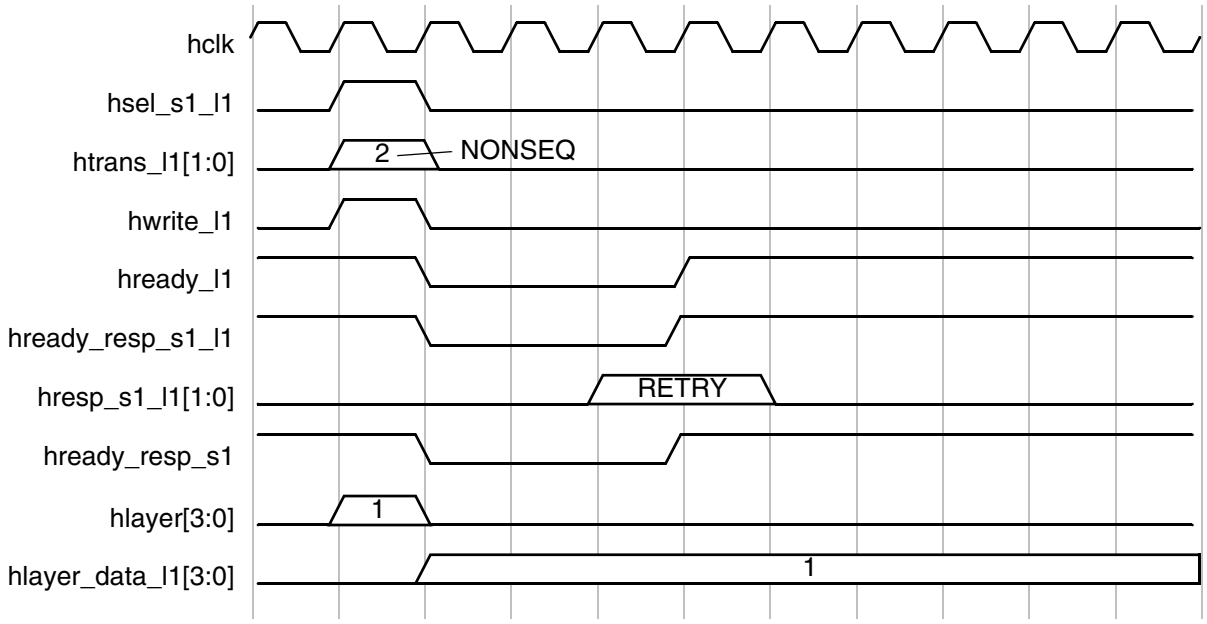
the input stage cleared. If there are subsequent transfers from Layer 2, they come directly from the layer and not from the input stage.

The resulting address bus that is passed to the slave is a sequence of back-to-back transfers. Bursts are allowed to complete from one layer before another layer gains access.

2.6.2 RETRY Generation

In Figure 2-4, a write transfer is generated on Layer 1 for a slave that is being accessed by another layer.

Figure 2-4 DW_ahb_icm Generating RETRY

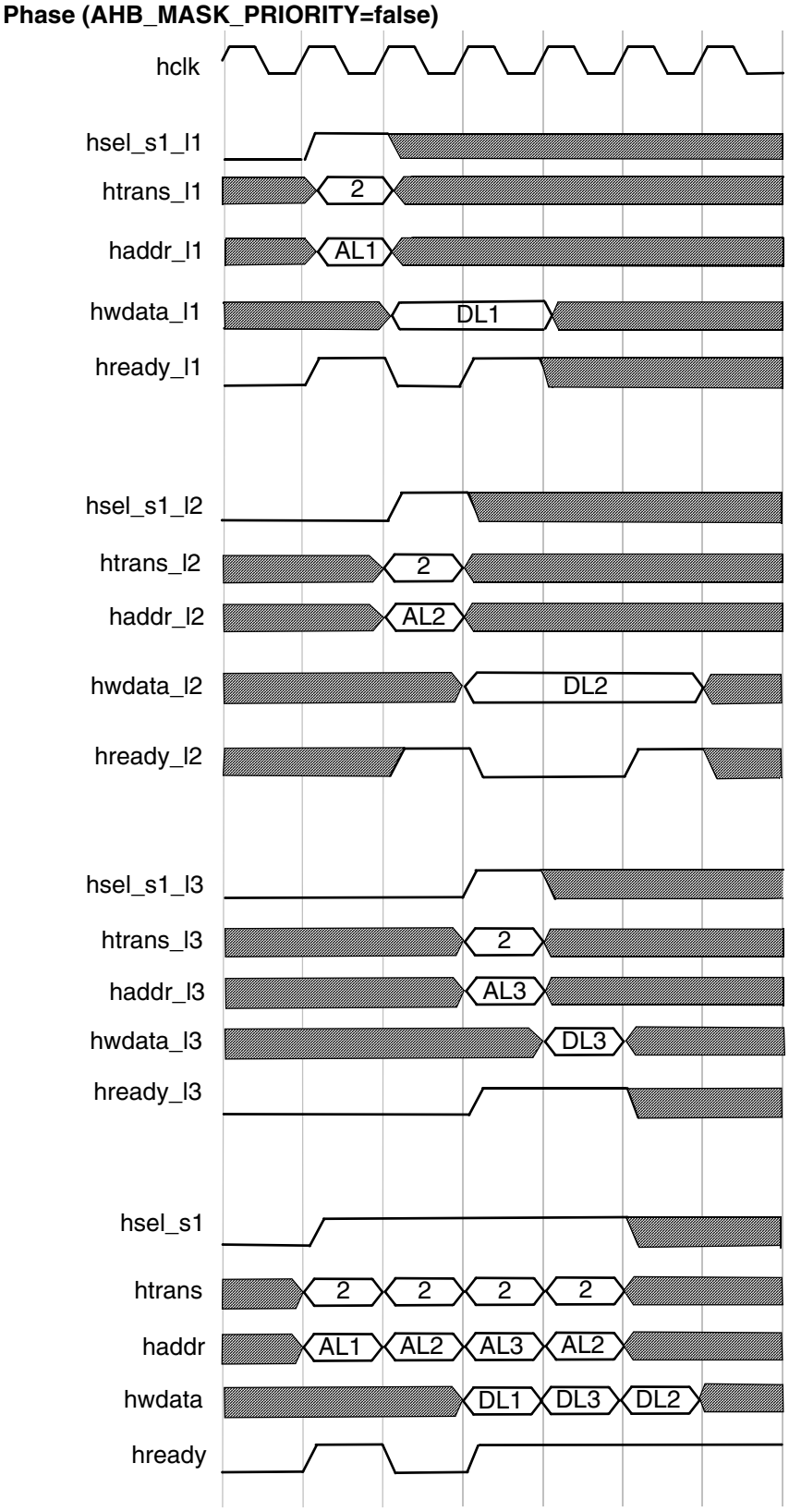


The DW_ahb_icm is configured to release a layer (ICM_RELEASE_L1 = 1). There is a clash of requests for access to the slave. The address and control, and the select, are all moved to an input stage where one waits for the clash of requests to clear. After an ICM_RLIMIT_L1 of cycles, there is still a clash of requests and then a RETRY response is sent back to the master on Layer 1. The address and control in the input stage is released. This RETRY would then free up Layer 1 for a higher priority master to do some other transfers or to retry the write transfer to the common slave. The slave is deemed free when there is no activity on the bus for the slave, or when the current granted layer is driving a NON-SEQUENTIAL transfer.

2.6.3 Mask Priority or Arbitration

Figure 2-5 shows the result of multiple layers attempting to access the common slave during the wait cycle of the previous transfer data phase with AHB_MASK_PRIORITY set to false.

Figure 2-5 Multiple Layers Attempting to Access the Common Slave during Wait Cycle of Previous Transfer Data



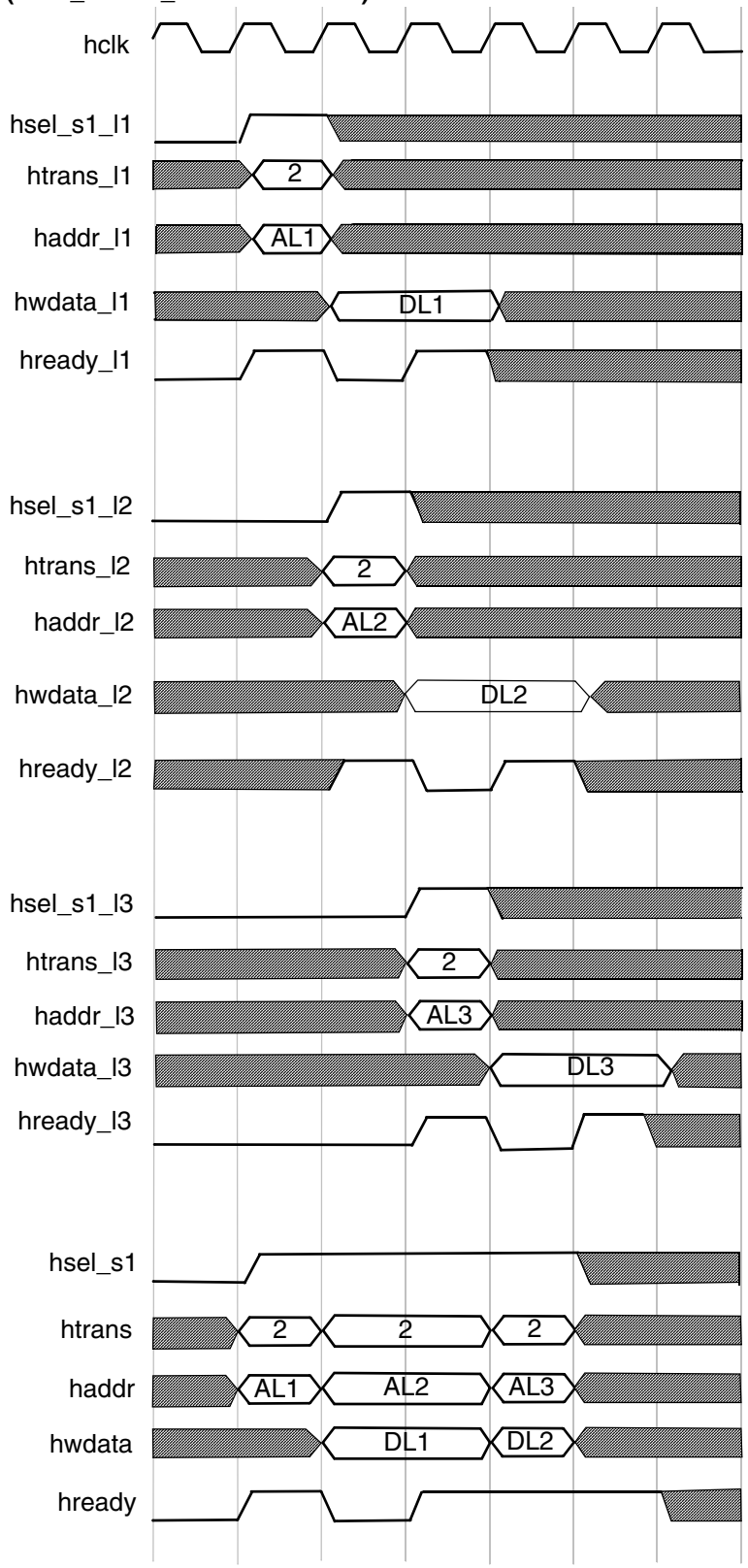
In Figure 2-5, the assumption is that AHB Layer 1 - AHB Layer 3 => Lowest Priority - Highest Priority

The AHB Layer 1 requests the DW_ahb_icm to the access the common slave and gets the grant because AHB Layer 1 is the highest priority layer requesting the common slave. The common slave inserts a wait cycle during the data phase of the AHB Layer 1 transfer.

During the wait cycle, AHB Layer 2 requests the common slave and it gets the grant temporarily because the arbitration is not locked to a particular AHB layer until hready is set to high, when AHB_MASK_PRIORITY is false. But, the actual grant or access to the common slave is given to the AHB Layer 3, as the AHB Layer 3 has the highest priority AHB layer requesting the common slave and hready is high at the same time.

Figure 2-6 shows the result of multiple layers trying to access the common slave during the wait cycle of the previous transfer data phase with AHB_MASK_PRIORITY set to true.

Figure 2-6 Multiple Layers Attempting to Access the Common Slave during Wait Cycle of Previous Transfer Data Phase (AHB_MASK_PRIORITY=true)



This scenario is same as discussed in Figure 2-5 except that AHB_MASK_PRIORITY is set to true. During the first wait cycle, AHB Layer 2 requests the common slave and it gets the grant. But, the difference is that the grant is locked to the AHB Layer 2 until it is accepted by the common slave, that is, until hready is high. The grant is not provided to any high priority AHB Layer thereafter until the granted AHB layer request is accepted by the common slave.

2.7 AMBA 5 AHB Features

Note You must have DWC-AMBA-AHB5-Fabric-Source add-on license to access AHB5 features (ICM_AHB_INTERFACE_TYPE=1).

This section describes the following features supported by DW_ahb_icm to compy with AMBA 5 AHB protocol specification.

- "Extended Memory Types" on page 30
- "Secure Transfers" on page 30
- "Exclusive Transfers" on page 30
- "Multiple Slave Select" on page 31
- "User Signals" on page 31

To enable these properties, select the AHB interface type as AHB5 (ICM_AHB_INTERFACE_TYPE=1) through the configuration parameter ICM_AHB_INTERFACE_TYPE.

2.7.1 Extended Memory Types

DW_ahb_icm supports Extended Memory Types feature of the AMBA 5 AHB protocol. When the configuration parameter ICM_HAS_EXTD_MEMTYPE is set to 1. To support this feature, DW_ahb_icm includes additional signals hprot_lx[6:4] and hprot[6:4] to master layer x and slave interfaces respectively. The extended memory type information is propagated unmodified across the DW_ahb_icm. For more information on these signals, see the *Signal Descriptions* chapter.

2.7.2 Secure Transfers

DW_ahb_icm supports secure transfers feature of AMBA 5 AHB protocol by adding the hnonsec_lx and the hnonsec signals to its master layer x and slave interfaces respectively. This feature is enabled by setting the configuration parameter ICM_HAS_SECURE_XFER to 1.

2.7.3 Exclusive Transfers

DW_ahb_icm supports Exclusive transfers property of AMBA 5 AHB protocol when the configuration parameter ICM_HAS_EXC_XFER is set to True. An exclusive transfer is indicated to the DW_ahb_icm by hexcl_lx signal. DW_ahb_icm intimates to the slave that the current transfer is part of an Exclusive access sequence through the hexcl signal. It is slave's responsibility to monitor the Exclusive access sequence and indicate the success or failure of an Exclusive transfer. The response of an exclusive transfer is indicated through the hexokay and hexokay_lx signals.

The master identifier signals hmaster_lx and hmaster exist in AHB5 configuration when the ICM_AHB5_HMASTER_WIDTH parameter is set to a non-zero value. These signals can be used by the slave to distinguish between different masters or to differentiate between multiple exclusive threads of a master. In AHB5 configuration, the hmaster_lx signal is appended with the layer number to differentiate between masters with the same index from different layers.

2.7.4 Multiple Slave Select

DW_ahb_icm supports Multiple Slave Select feature. To enable this feature, set the configuration parameter ICM_HAS_MHSEL to 1. The number of select lines can be configured through the ICM_NUM_HSEL parameter.

2.7.5 User Signals

DW_ahb_icm supports user signals on each channel. The configuration parameters ICM_R_UBW, ICM_W_UBW and ICM_A_UBW determine the existence and width of the user signals on read data, write data, and address channels. The User signals associated with a transfer are transported unmodified across DW_ahb_icm.

The timing of address user signals is same as that of address and other control signals; and the timing of read and write data channel user signals is same as that of the corresponding data signals.

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.

- Top Level Parameters on page 34
- Top Level Parameters / DW_ahb_icm Source Code Configuration on page 39
- AMBA 5 AHB Configuration on page 40
- User Signal Configuration on page 41

3.1 Top Level Parameters

Table 3-1 Top Level Parameters

Label	Description
System Configuration	
AMBA Lite ?	When this is set to Yes, no master in the system is capable of accepting a RETRY or a SPLIT response from a slave. Therefore, it is not possible to have a split-capable slave within the system, nor is it possible for the DW_ahb_icm to generate the RETRY. On a layer, this setting implies only a single master with no dummy master, and some parameters are disabled and overwritten with their default values. **Values:** ■ false (0) ■ true (1) **Default Value:** false **Enabled:** Always **Parameter Name:** AHB_LITE
Single-bit AHB Response in AHB-Lite mode?	When set to true, hresp is a single-bit signal, else it is of 2 bits. **Values:** 0, 1 **Default Value:** 0 **Enabled:** AHB_LITE==1 **Parameter Name:** AHB_SCALAR_HRESP
AHB Interface Type?	This parameter selects the AHB interface type. **Values:** ■ AHB (0) ■ AHB5 (1) **Default Value:** AHB **Enabled:** AHB_LITE==1 and DWC-AMBA-AHB5-Fabric-Source Add on Source license exists. **Parameter Name:** ICM_AHB_INTERFACE_TYPE
Mask the Priority during the progress of previous transaction ?	If this parameter is set to True, the DW_ahb_icm masks the bus arbitration to all AHB layers, except for AHB Layer for which transaction is in progress under following conditions: A NONSEQ transfer is issued from one AHB Layer, if Slave issues a wait state through pulling hready low (data phase of previous transaction). **Values:** ■ false (0) ■ true (1) **Default Value:** false **Enabled:** Always **Parameter Name:** AHB_MASK_PRIORITY

Table 3-1 Top Level Parameters (Continued)

Label	Description
AHB System Address Width	Specifies the width of the connected address bus. **Values:** ■ 32 (32) ■ 64 (64) **Default Value:** 32 **Enabled:** Always **Parameter Name:** HADDR_WIDTH
AHB Data Bus Width	Specifies the width of the AHB data bus. **Values:** 32, 64, 128, 256 **Default Value:** 32 **Enabled:** Always **Parameter Name:** AHB_DATA_WIDTH
Non Standard Master ID Sideband Signal Width	Specifies the width of a non standard Master ID sideband signals. When set to 0, the Master ID sideband signals are removed. **Values:** 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12 **Default Value:** 0 **Enabled:** AHB_LITE == 1 && ICM_AHB_INTERFACE_TYPE==0 **Parameter Name:** MID_WIDTH
Layer Control	
Number of layers?	The number of AHB layers that the DW_ahb_icm must support. **Values:** 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16 **Default Value:** 2 **Enabled:** Always **Parameter Name:** ICM_NUM_LAYERS
Slave Select Control	
Slave Has Multiple Selects?	This parameter controls the connected slave select lines. The connected slave can have multiple slave-select lines and share a common AHB interface layer, or it may have only one. **Values:** ■ false (0) ■ true (1) **Default Value:** =[<functionof> equal $::shell_activity_mode "assembler"]?1:0 **Enabled:** ![<functionof> equal $::shell_activity_mode "assembler"] **Parameter Name:** ICM_HAS_MHSEL

Table 3-1 Top Level Parameters (Continued)

Label	Description
Number of Select Line?	Number of select lines attached to a slave. **Values:** 1, 2, 3, 4, 5, 6, 7, 8 **Default Value:** 1 **Enabled:** ICM_HAS_MHSEL==1 **Parameter Name:** ICM_NUM_HSEL
Interface Control	
Slave uses burst control?	The connected slave uses and requires the burst control from each of the masters. **Values:** ■ false (0) ■ true (1) **Default Value:** true **Enabled:** Always **Parameter Name:** ICM_HAS_BURST
Slave uses lock control?	The connected slave uses and requires the lock control from each of the masters. **Values:** ■ false (0) ■ true (1) **Default Value:** true **Enabled:** Always **Parameter Name:** ICM_HAS_LOCK
Slave uses protection control?	The connected slave uses and requires the protection control from each of the masters. **Values:** ■ false (0) ■ true (1) **Default Value:** false **Enabled:** Always **Parameter Name:** ICM_HAS_PROT

Table 3-1 Top Level Parameters (Continued)

Label	Description
External Priority Control?	This setting, when True (1), enables you to control the arbitration priority from external signals. This is the only way for you to dynamically change the priorities. When priorities are internal, they are fixed with Layer 1 the highest priority down to Layer 8 with the lowest priority. The external priorities are controlled by driving the icm_priority bus input. When there are: ■ 2 layers - bus is 2-bits wide; each layer has 1 bit for priority. ■ 3 layers - bus is 6-bits wide; each layer has 2 bits for priority. ■ 4 layers - bus is 8-bits wide; each layer has 2 bits for priority. ■ 5 layers - bus is 15-bits wide; each layer has 3 bits for priority. ■ 6 layers - bus is 18-bits wide; each layer has 3 bits for priority. ■ 7 layers - bus is 21-bits wide; each layer has 3 bits for priority. ■ 8 layers - bus is 24-bits wide; each layer has 3 bits for priority. ■ 9 layers - bus is 36-bits wide, each layer has 4 bit for priority. ■ 10 layers - bus is 40-bits wide; each layer has 4 bits for priority. ■ 11 layers - bus is 44-bits wide; each layer has 4 bits for priority. ■ 12 layers - bus is 48-bits wide; each layer has 4 bits for priority. ■ 13 layers - bus is 52-bits wide; each layer has 4 bits for priority. ■ 14 layers - bus is 56-bits wide; each layer has 4 bits for priority. ■ 15 layers - bus is 60-bits wide; each layer has 4 bits for priority. ■ 16 layers - bus is 64-bits wide; each layer has 4 bits for priority. The bit order of the icm_priority is always: [layer16]_[layer15]_[layer14]_[layer13]_[layer12]_[layer11]_[layer10]_[layer9] [layer8]_[layer7]_[layer6]_[layer5]_[layer4]_[layer3]_[layer2]_[layer1] For a four-layer system, the following is an example of driving icm_priority: icm_priority[1:0] => layer 1 (2'b00) Highest Priority icm_priority[3:2] => layer 2 (2'b11) Fourth Priority icm_priority[5:4] => layer 3 (2'b01) Second Priority icm_priority[7:6] => layer 4 (2'b10) Third Priority If the priorities are the same, then the layer index is used to calculate the highest priority. If layer 1 and layer 2 have the same priority then layer 1 is deemed as the higher priority. **Values:** ■ false (0) ■ true (1) **Default Value:** false **Enabled:** Always **Parameter Name:** ICM_HAS_XPRIORITY

Table 3-1 Top Level Parameters (Continued)

Label	Description
Split Control	
Slave is split capable?	The connected slave is split-capable and needs to know which master is driving the address and control. **Values:** ■ false (0) ■ true (1) **Default Value:** false **Enabled:** AHB_LITE == 0 **Parameter Name:** ICM_HAS_SPLIT
Holding Control Layer x	
Matrix can release layer x? (for x = 1; x <= 16)	Enables a layer to give the option for a higher priority layer to be granted access to the slave. A layer may be prevented from getting access to the slave if other layers have higher priority or are taking a long time. Meanwhile, masters on the layer can do nothing with that slave and may need to access other slaves. When the slave becomes available, the DW_ahb_icm allows the layer to RETRY, whereby a master on that layer can continue with the original transfer or enable another master to have access. If there is no other master requesting access, then the master that was being held off can RETRY the transfer. **Values:** ■ false (0) ■ true (1) **Default Value:** false **Enabled:** This option is not available when the component is configured as an AMBA Lite system (AHB_LITE = 1). **Parameter Name:** ICM_RELEASE_Lx
Wait Limit layer x (for x = 1; x <= 16)	The number of cycles that a layer can be held off before the matrix generates a RETRY response. Once a RETRY is generated, any stored address and control is cancelled and the master with the highest priority on Layer 1 gains access. **Values:** 0x4, 0x5, 0x6, 0x7, 0x8, 0x9, 0xa, 0xb, 0xc, 0xd, 0xe, 0xf **Default Value:** 0x4 **Enabled:** This option is not available when the component is configured as an AMBA Lite system (AHB_LITE = 1). **Parameter Name:** ICM_RLIMIT_Lx

3.2 Top Level Parameters / DW_ahb_icm Source Code Configuration Parameters

Table 3-2 Top Level Parameters / DW_ahb_icm Source Code Configuration Parameters

Label	Description
DW_ahb_icm Source Code Configuration	
Use DesignWare Foundation Synthesis Library	The component code utilizes DesignWare Foundation parts for optimal Synthesis QoR. Customers with only a DesignWare license must use Foundation parts. Customers with only a Source license cannot use Foundation parts. Customers with both Source and DesignWare licenses have the option of using DesignWare Foundation parts. **Values:** ■ false (0) ■ true (1) **Default Value:** True if DesignWare License is available; False if no DesignWare License is available **Enabled:** Parameter is enabled if customer has both Source and DesignWare licenses **Parameter Name:** USE_FOUNDATION

3.3 AMBA 5 AHB Configuration Parameters

Table 3-3 AMBA 5 AHB Configuration Parameters

Label	Description
AMBA 5 AHB Configuration	
Include AHB5 Extended Memory Types Property?	Select this parameter to include AHB5 Extended Memory Types Property in DW_ahb_icm. When set to 1, DW_ahb_icm adds hprot[6:4] and hprot_lx[6:4] signals to its interface to support this property. **Values:** ■ false (0) ■ true (1) **Default Value:** false **Enabled:** ICM_AHB_INTERFACE_TYPE==1 && ICM_HAS_PROT==1 **Parameter Name:** ICM_HAS_EXTD_MEMTYPE
Include AHB5 Secure Transfers Property?	Select this parameter to include AHB5 Secure Transfers Property in DW_ahb_icm. When set to 1, DW_ahb_icm adds hnonsec and hnonsec_lx signals to its interface to support this property. **Values:** ■ false (0) ■ true (1) **Default Value:** false **Enabled:** ICM_AHB_INTERFACE_TYPE==1 **Parameter Name:** ICM_HAS_SECURE_XFER
Include AHB5 Exclusive Transfers Property?	Select this parameter to include AHB5 Exculsive Transfers property in DW_ahb_icm. When set to 1, DW_ahb_icm adds hexcl and hexcl_lx signals to its interface to support this property. **Values:** ■ false (0) ■ true (1) **Default Value:** false **Enabled:** ICM_AHB_INTERFACE_TYPE==1 **Parameter Name:** ICM_HAS_EXC_XFER
Width of AHB5 Master Identifier Signal?	This parameter determines the width of the AHB5 master identifier signals hmaster and hmaster_lx in AHB5 configuration. When set to 0, these signals do not exist in AHB5 configuration. **Values:** 0, ..., 12 **Default Value:** 0 **Enabled:** ICM_AHB_INTERFACE_TYPE == 1 **Parameter Name:** ICM_AHB5_HMASTER_WIDTH

3.4 User Signal Configuration Parameters

Table 3-4 User Signal Configuration Parameters

Label	Description
User Signal Configuration	
Width of Address Channel User Signal Bus?	This parameter specifies the width of address channel user signal bus. When set to 0, the address channel user signals are removed from the interface. **Values:** 0, ..., 256 **Default Value:** 0 **Enabled:** DWC-AMBA-AHB5-Fabric-Source Add on Source license exists. **Parameter Name:** ICM_A_UBW
Width of Write Data Channel User Signal Bus?	This parameter specifies the width of write data channel user signal bus. When set to 0, the write data channel user signals are removed from the interface. **Values:** 0, ..., 256 **Default Value:** 0 **Enabled:** DWC-AMBA-AHB5-Fabric-Source Add on Source license exists. **Parameter Name:** ICM_W_UBW
Width of Read Data Channel User Signal Bus?	This parameter specifies the width of read data channel user signal bus. When set to 0, the read data channel user signals are removed from the interface. **Values:** 0, ..., 256 **Default Value:** 0 **Enabled:** DWC-AMBA-AHB5-Fabric-Source Add on Source license exists. **Parameter Name:** ICM_R_UBW

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 Reset Interface on page 45
- Arbitration Control on page 46
- Slave Layer Interface on page 47
- Master Layer x Interface on page 52

4.1 Clock and Reset Interface Signals

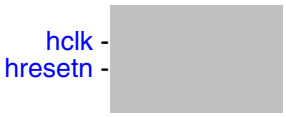


Table 4-1 Clock and Reset Interface Signals

Port Name	I/O	Description
hclk	I	Bus clock. Active on rising edge. Common clock for all layers. **Exists:** Always **Synchronous To:** None **Registered:** N/A **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
hresetn	I	Bus reset. Common reset for all layers. Asynchronous assertion, synchronous de-assertion. The reset must be synchronously de-asserted after the rising edge of hclk. DW_ahb_icm does not contain logic to perform this synchronization, so it must be provided externally. **Exists:** Always **Synchronous To:** Asynchronous **Registered:** N/A **Power Domain:** SINGLE_DOMAIN **Active State:** Low

4.2 Arbitration Control Signals

icm_priority -

Table 4-2 Arbitration Control Signals

Port Name	I/O	Description
icm_priority[((ICM_NUM_LAYERS*ICM_DP_PWIDTH)-1):0]	I	This bus allows you to control the arbitration priority externally. In cases where the number of layers is not a power of 2 (that is, 2, 4, 8 or 16), the priority of a layer should be programmed with only a value from 0 to ICM_NUM_LAYERS-1. The following range of values will cause an illegal operation, if used: ICM_NUM_LAYERS to 2 ^ (ceil(log2ICM_NUM_LAYERS)). **Exists:** (ICM_HAS_XPRIORITY == 1) **Synchronous To:** hclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** High

4.3 Slave Layer Interface Signals

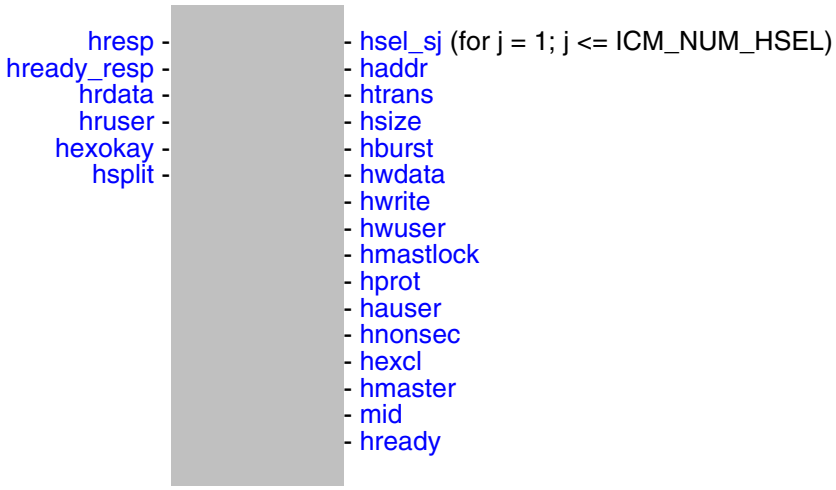


Table 4-3 Slave Layer Interface Signals

Port Name	I/O	Description
hresp[(HRESP_WIDTH-1):0]	I	Response type from slave. **Exists:** Always **Synchronous To:** (AHB_MASK_PRIORITY==1 || ICM_HAS_SPLIT==1) ? "hclk" : "None" **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
hready_resp	I	Transfer complete. **Exists:** Always **Synchronous To:** hclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** High
hrdata[(AHB_DATA_WIDTH-1):0]	I	Read data from slave. **Exists:** Always **Synchronous To:** hclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A

Table 4-3 Slave Layer Interface Signals (Continued)

Port Name	I/O	Description
hruser[(ICM_R_UBW-1):0]	I	Slave user bus for read data channel. **Exists:** (ICM_R_UBW > 0) **Synchronous To:** hclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
hexokay	I	Exclusive transfer response from slave. When asserted, this signal indicates that the Exclusive transfer has been successful. When deasserted it indicates that the Exclusive transfer has failed. **Exists:** (ICM_HAS_EXC_XFER == 1) **Synchronous To:** hclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** High
hsplit[15:0]	I	Split response from slave. Used to clear the internal arbitration to allow the master to re-arbitrate. One bit for each master on a layer. The response is passed to the layer that was split and that initiated the transfer. **Exists:** (ICM_HAS_SPLIT == 1) **Synchronous To:** hclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** High
hsel_sj (for j = 1; j <= ICM_NUM_HSEL)	O	Slave select line. jth slave signal. **Exists:** ((ICM_HAS_MHSEL==1) && (ICM_NUM_HSEL>=j)) **Synchronous To:** hclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** High
haddr[(HADDR_WIDTH-1):0]	O	Slave address. **Exists:** Always **Synchronous To:** hclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A

Table 4-3 Slave Layer Interface Signals (Continued)

Port Name	I/O	Description
htrans[1:0]	O	Slave transfer control. htrans = 00 --> IDLE htrans = 01 --> BUSY htrans = 10 --> NONSEQ htrans = 11 --> SEQ **Exists:** Always **Synchronous To:** hclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
hsize[2:0]	O	Slave transfer size. **Exists:** Always **Synchronous To:** hclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
hburst[(HBURST_WIDTH-1):0]	O	Slave transfer type. **Exists:** (ICM_HAS_BURST == 1) **Synchronous To:** hclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
hwdata[(AHB_DATA_WIDTH-1):0]	O	Slave write data. **Exists:** Always **Synchronous To:** hclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
hwrite	O	Slave write control. **Exists:** Always **Synchronous To:** hclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** High
hwuser[(ICM_W_UBW-1):0]	O	Slave user bus for write data channel. **Exists:** (ICM_W_UBW > 0) **Synchronous To:** hclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A

Table 4-3 Slave Layer Interface Signals (Continued)

Port Name	I/O	Description
hmastlock	O	Slave locked transfer control. **Exists:** (ICM_HAS_LOCK == 1) **Synchronous To:** hclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** High
hprot[(HPROT_WIDTH-1):0]	O	Slave protection control. When the DW_ahb_icm is configured with AHB5 interface(ICM_AHB_INTERFACE_TYPE=1) and the parameter ICM_HAS_EXTD_MEMTYPE is set to 1 width of the hprot is increased to 7 bits. The 3-bit extension of the protection control signal, hprot[6:4], indicates the extended memory types. These extended signals indicate the lookup, allocate and shareable attributes of the transfer. **Exists:** (ICM_HAS_PROT == 1) **Synchronous To:** hclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
hauser[(ICM_A_UBW-1):0]	O	Slave user bus for address channel. **Exists:** (ICM_A_UBW > 0) **Synchronous To:** hclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
hnonsec	O	Slave Non-secure or Secure transfer type. When asserted, this signal indicates that the current transfer is a Non-secure transfer. When de-asserted the transfer is a Secure transfer. **Exists:** (ICM_HAS_SECURE_XFER == 1) **Synchronous To:** hclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** High
hexcl	O	Slave Exclusive transfer type. When asserted, this signal indicates to slave that the current transfer is part of an Exclusive access sequence. **Exists:** (ICM_HAS_EXC_XFER == 1) **Synchronous To:** hclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** High

Table 4-3 Slave Layer Interface Signals (Continued)

Port Name	I/O	Description
hmaster[(HMASTER_OUT_WIDTH-1):0]	O	The master that is currently accessing the slave, Each layer keeps track of which master is driving and presents this to the slave. **Exists:** (ICM_HAS_SPLIT == 1) || (ICM_AHB5_HMASTER_WIDTH > 0) **Synchronous To:** hclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
mid[(MID_LAYNO_WIDTH-1):0]	O	Non standard Master ID sideband signal output (in AHB-Lite mode). **Exists:** (MID_WIDTH!=0) **Synchronous To:** hclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
hready	O	Current transfer is complete. **Exists:** Always **Synchronous To:** hclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** High

4.4 Master Layer x Interface Signals

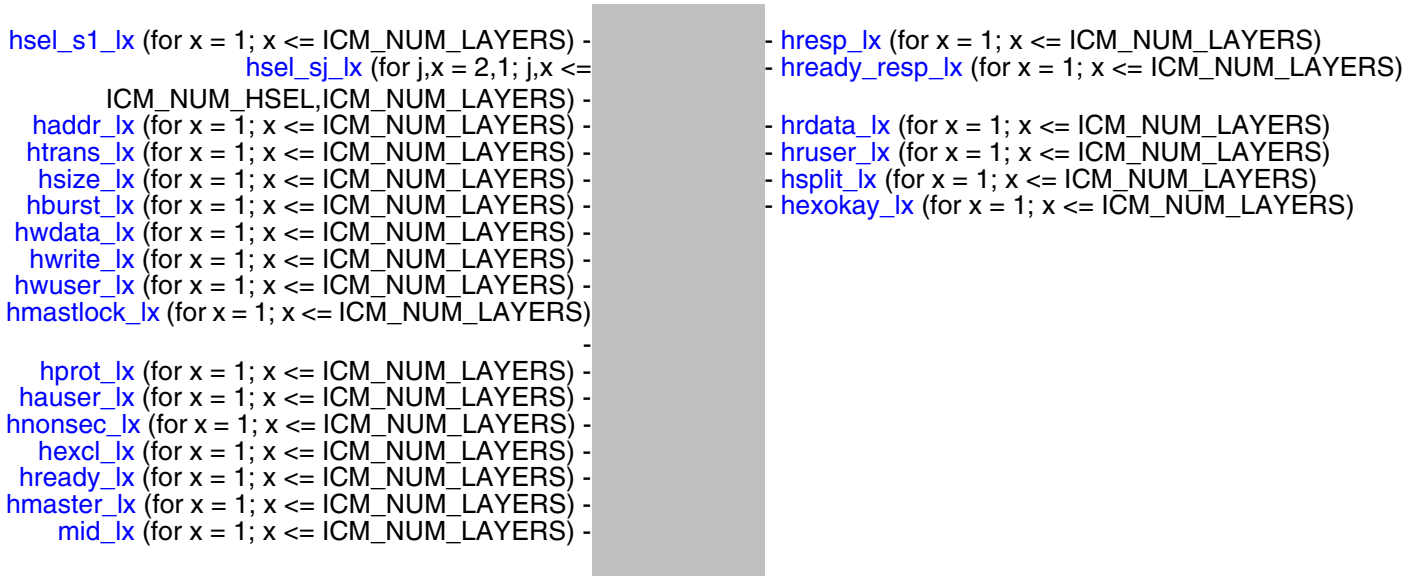


Table 4-4 Master Layer x Interface Signals

Port Name	I/O	Description
hsel_s1_lx (for x = 1; x <= ICM_NUM_LAYERS)	I	Slave select from the layer decoder. 1st slave select signal on Layer x. **Exists:** (ICM_NUM_LAYERS>=x) **Synchronous To:** hclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** High
hsel_sj_lx (for j,x = 2,1; j,x <= ICM_NUM_HSEL,ICM_NUM_LAYERS)	I	Slave select from the layer decoder. jth slave select signal on Layer x. **Exists:** ((ICM_HAS_MHSEL==1) && (ICM_NUM_HSEL>=j) && (ICM_NUM_LAYERS>=x)) **Synchronous To:** hclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** High
haddr_lx[(HADDR_WIDTH-1):0] (for x = 1; x <= ICM_NUM_LAYERS)	I	Address. **Exists:** Always **Synchronous To:** hclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A

Table 4-4 Master Layer x Interface Signals (Continued)

Port Name	I/O	Description
htrans_lx[1:0] (for x = 1; x <= ICM_NUM_LAYERS)	I	Transfer control. htrans_lx = 00 --> IDLE htrans_lx = 01 --> BUSY htrans_lx = 10 --> NONSEQ htrans_lx = 11 --> SEQ **Exists:** Always **Synchronous To:** hclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
hsize_lx[2:0] (for x = 1; x <= ICM_NUM_LAYERS)	I	Transfer size. **Exists:** Always **Synchronous To:** hclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
hburst_lx[(HBURST_WIDTH-1):0] (for x = 1; x <= ICM_NUM_LAYERS)	I	Transfer type. **Exists:** (ICM_HAS_BURST == 1) **Synchronous To:** hclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
hwdata_lx[(AHB_DATA_WIDTH-1):0] (for x = 1; x <= ICM_NUM_LAYERS)	I	Write data. **Exists:** Always **Synchronous To:** hclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
hwrite_lx (for x = 1; x <= ICM_NUM_LAYERS)	I	Write control. **Exists:** Always **Synchronous To:** hclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** High
hwuser_lx[(ICM_W_UBW-1):0] (for x = 1; x <= ICM_NUM_LAYERS)	I	User bus for write data channel. **Exists:** (ICM_W_UBW > 0) **Synchronous To:** hclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A

Table 4-4 Master Layer x Interface Signals (Continued)

Port Name	I/O	Description
hmastlock_lx (for x = 1; x <= ICM_NUM_LAYERS)	I	Locked transfer control. **Exists:** (ICM_HAS_LOCK == 1) **Synchronous To:** hclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** High
hprot_lx[(HPROT_WIDTH-1):0] (for x = 1; x <= ICM_NUM_LAYERS)	I	Protection control. When the DW_ahb_icm is configured with AHB5 interface (ICM_AHB_INTERFACE_TYPE=1) and the parameter ICM_HAS_EXTD_MEMTYPE is set to 1, width of hprot_lx is increased to 7 bits. The 3-bit extension of the protection control signal, hprot_lx[6:4], indicates the extended memory types. These extended signals indicate the lookup, allocate and shareable attributes of the transfer. **Exists:** (ICM_HAS_PROT == 1) **Synchronous To:** hclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
hauser_lx[(ICM_A_UBW-1):0] (for x = 1; x <= ICM_NUM_LAYERS)	I	User bus for address channel. **Exists:** (ICM_A_UBW > 0) **Synchronous To:** hclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
hnonsec_lx (for x = 1; x <= ICM_NUM_LAYERS)	I	Non-secure or Secure transfer type from Layer x. When asserted, this signal indicates that the current transfer is a Non-secure transfer. When de-asserted the transfer is a Secure transfer. **Exists:** (ICM_HAS_SECURE_XFER == 1) **Synchronous To:** hclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** High
hexcl_lx (for x = 1; x <= ICM_NUM_LAYERS)	I	Exclusive transfer type from Layer x. When asserted, this signal indicates to slave that the current transfer is part of an Exclusive access sequence. **Exists:** (ICM_HAS_EXC_XFER == 1) **Synchronous To:** hclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** High

Table 4-4 Master Layer x Interface Signals (Continued)

Port Name	I/O	Description
hready_lx (for x = 1; x <= ICM_NUM_LAYERS)	I	Previous transfer is complete. **Exists:** Always **Synchronous To:** hclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** High
hmaster_lx[(HMASTER_WIDTH-1):0] (for x = 1; x <= ICM_NUM_LAYERS)	I	Index of the master that is active on the layer. When a slave is split capable, it needs to know which master was driving the address and control signals so that it is able to release the correct master when it returns from a split. **Exists:** (ICM_HAS_SPLIT == 1) || (ICM_AHB5_HMASTER_WIDTH > 0) **Synchronous To:** hclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
mid_lx[(MID_WIDTH-1):0] (for x = 1; x <= ICM_NUM_LAYERS)	I	Non standard Master ID sideband signal input (in AHB-Lite mode). **Exists:** (MID_WIDTH!=0) **Synchronous To:** hclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
hresp_lx[(HRESP_WIDTH-1):0] (for x = 1; x <= ICM_NUM_LAYERS)	O	Response type from slave to the Layer x. **Exists:** Always **Synchronous To:** hclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
hready_resp_lx (for x = 1; x <= ICM_NUM_LAYERS)	O	Transfer complete. When asserted, current transfer has been successful. **Exists:** Always **Synchronous To:** hclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** High

Table 4-4 Master Layer x Interface Signals (Continued)

Port Name	I/O	Description
hrdata_lx[(AHB_DATA_WIDTH-1):0] (for x = 1; x <= ICM_NUM_LAYERS)	O	Read data from slave to the Layer x. **Exists:** Always **Synchronous To:** hclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
hruser_lx[(ICM_R_UBW-1):0] (for x = 1; x <= ICM_NUM_LAYERS)	O	Read data channel user bus from slave to the Layer x. **Exists:** (ICM_R_UBW > 0) **Synchronous To:** hclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
hsplit_lx[15:0] (for x = 1; x <= ICM_NUM_LAYERS)	O	Split data from slave. **Exists:** (ICM_HAS_SPLIT == 1) **Synchronous To:** hclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** High
hexokay_lx (for x = 1; x <= ICM_NUM_LAYERS)	O	Exclusive transfer response from slave to Layer x. When asserted, this signal indicates that the Exclusive transfer has been successful. When deasserted it indicates that the Exclusive transfer has failed. **Exists:** (ICM_HAS_EXC_XFER == 1) **Synchronous To:** hclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** High

5 Verification

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


Note The DW_ahb_icm verification testbench is built with DesignWare Verification IP (VIP). Make sure you have the supported version of the VIP components for this release, otherwise, you may experience some tool compatibility problems. For more information about supported tools in this release, see the *DesignWare Synthesizable Components for AMBA 2/AMBA 3 AXI Installation Guide.*


Note The packaged test benches are only for validating the IP configuration in coreConsultant GUI. It is not for system level validation.
IPs that have the Vera test bench packaged, these test benches are encrypted.

5.1 Overview of Vera Tests

The DW_ahb_icm verification testbench performs the following set of tests that have been written to exhaustively verify functionality and have also achieved maximum RTL code coverage.

5.1.1 Access

This test verifies that it is possible for each layer to access the slave. At no time during this test do the layers compete for access at the same time to the slave. All types of transfers from read and write transfers to single and burst transfers are generated for the slave from each layer. The slave is configured to respond with random wait states for each access. The test writes data from one layer and reads it with another layer. This test verifies that slaves with multiple select lines are capable of getting access for each bit of the select. The test generates transfers to the slave for addresses within all of its regions. This test also verifies that burst, lock, and protection control are channeled through the DW_ahb_icm from the layer that initiated the transfer.

5.1.2 Priority Scheme

This test verifies the priority scheme between the layers. Multiple layers compete for access to the slave at the same time. As soon as transfers are started from lower priority layers, transfers to the slave from higher priority layers are started. The accesses vary from single transfers to burst transfers. Different scenarios are generated where one layer starts a burst before another layer starts.

5.1.3 Locked Transfers

This test verifies that once a transfer that is locked starts, no other layer can gain access to the slave until the lock is removed, regardless of its priority. This test verifies that once burst transfers start, they are allowed to complete without another layer gaining access to the slave.

5.1.4 Responses

This test verifies the following:

- Generation of RETRY response by the matrix
- Generation of responses from the common slave
- Channelling of the responses to the layer that initiated the transfer

This test is broken into two parts:

1. Slave generates all response types. Split responses are generated only if the slave is configured to be capable of generating a split response. RETRY responses are generated only if the matrix is not connected to AHB Lite master layers.
2. Matrix generates RETRY and is only run when the matrix is configured to implement a watchdog on the length of time a layer can be held off before it has to generate a retry response.

The test configures the slave for all responses. Split responses are configured to return from the split after a number of different cycles. The test configures a clash between layers and generates bus traffic where layers are held off for more than their limit, ensuring the DW_ahb_icm releases the layer.

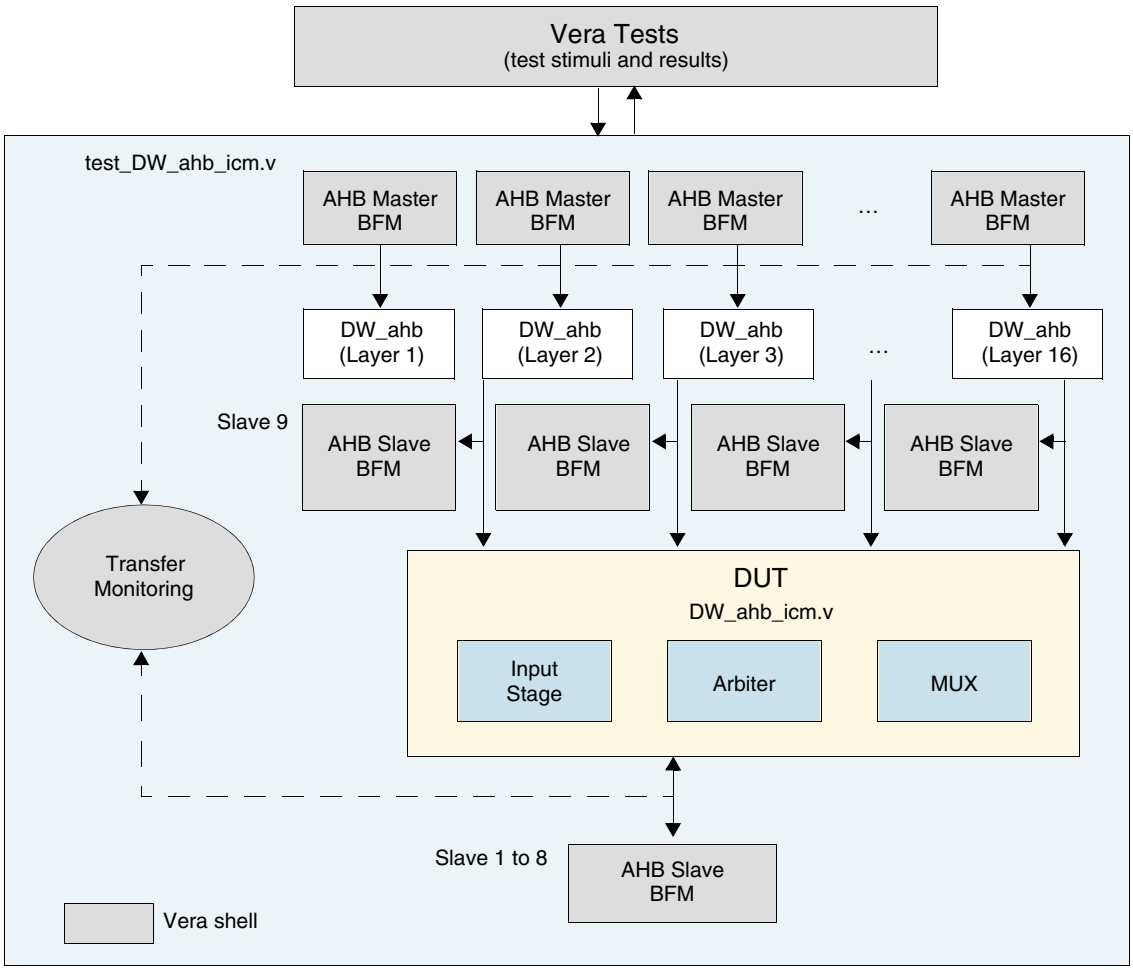
For more information on AHB Lite, see “Functional Description” chapter in the *DesignWare DW_ahb Databook*.

5.2 DW_ahb_icm Testbench

As illustrated in Figure 5-1, the DW_ahb_icm Verilog testbench includes:

- Instantiation of the Design Under Test (DUT)
- Multiple AHB bus models
- Vera shell

Figure 5-1 DW_ahb_icm Testbench



The Vera shell consists of the following:

- AHB Master Bus Functional Models (BFM)
- AHB Slave BFMs
- Transfer Monitor
- Test stimuli
- BFM configuration
- Test results

The Transfer Monitor tracks the transfers from the masters to the slaves, ensuring all are completed and none are lost even during retry/split transfers.

The file, test_DW_ahb_icm.v, shows the instantiation of the top-level MacroCell in a testbench and resides in the *workspace*/src directory. The testbench tests the user configuration specified in the Specify Configuration task of coreConsultant and is self-checking. When a coreKit is unpacked and configured, the verification environment is stored in *workspace*/sim. Files in *workspace*/sim/test_icm form the actual testbench for DW_ahb_icm.

6 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.

Table 6-1 provides information about the synthesis results (power consumption, frequency, and area) of the DW_ahb_icm using the industry standard 28nm technology library and how it affects performance.

6.1 Performance

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

6.1.1 Power Consumption, Frequency, and Area and DFT Coverage

Table 6-1 provides information about the synthesis results (power consumption, frequency and area) and DFT coverage of the DW_ahb_icm using the industry standard 7nm technology library.

Table 6-1 Synthesis Results for DW_ahb_icm

Configuration	Operating Frequency	Gate Count	Power Consumption		TetraMax Coverage (%)		SpyGlass StuckAtCov(%)
			Static Power	Dynamic Power	StuckAtTest	Transition	
Default Configuration	hclk = 300MHz	1163	3 nW	0.028 mW	100	100	99.9
Typical Configuration 1 AHB_LITE = 1 AHB_MASK_PRIORITY = 1 ICM_NUM_LAYERS = 16 ICM_NUM_HSEL = 8 ICM_HAS_MHSEL = 1 ICM_HAS_PROT = 1 MID_WIDTH = 12 ICM_AHB_INTERFACE_TYPE = 1 ICM_HAS_EXTD_MEMTYPE = 1 ICM_HAS_SECURE_XFER = 1 ICM_HAS_EXC_XFER = 1 ICM_AHB5_HMASTER_WIDTH = 12 ICM_A_UBW = 256 ICM_W_UBW = 256 ICM_R_UBW = 256	hclk = 300MHz	71780	200 nW	1.510 mW	100	100	100

Table 6-1 Synthesis Results for DW_ahb_icm (Continued)

Configuration	Operating Frequency	Gate Count	Power Consumption		TetraMax Coverage (%)		SpyGlass StuckAtCov(%)
			Static Power	Dynamic Power	StuckAtTest	Transition	
Typical Configuration 2 AHB_MASK_PRIORITY = 1 HADDR_WIDTH = 64 AHB_DATA_WIDTH = 256 ICM_NUM_LAYERS = 16 ICM_HAS_MHSEL = 1 ICM_NUM_HSEL = 8 ICM_HAS_PROT = 1 ICM_HAS_SPLIT = 1 ICM_HAS_BURST = 1 ICM_HAS_LOCK = 1 ICM_HAS_XPRIORITY = 1 ICM_RELEASE_L1 = 1 ICM_RELEASE_L2 = 1 ICM_RELEASE_L3 = 1 ICM_RELEASE_L4 = 1 ICM_RELEASE_L5 = 1 ICM_RELEASE_L6 = 1 ICM_RELEASE_L7 = 1 ICM_RELEASE_L8 = 1 ICM_RELEASE_L9 = 1 ICM_RELEASE_L10 = 1 ICM_RELEASE_L11 = 1 ICM_RELEASE_L12 = 1 ICM_RELEASE_L13 = 1 ICM_RELEASE_L14 = 1 ICM_RELEASE_L15 = 1	hclk = 300MHz	29319	80 nW	0.543 mW	99.9	99.77	100

Table 6-1 Synthesis Results for DW_ahb_icm (Continued)

Configuration	Operating Frequency	Gate Count	Power Consumption		TetraMax Coverage (%)		SpyGlass StuckAtCov(%)
			Static Power	Dynamic Power	StuckAtTest	Transition	
ICM_RELEASE_L16 = 1 ICM_RLIMIT_L4 = 6 ICM_RLIMIT_L8 = 15 ICM_RLIMIT_L11 = 9							

A
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 A-1 Internal Parameters

Parameter Name	Equals To
BUSY	2'b01
HBURST_WIDTH	3
HMASTER_OUT_WIDTH	{ICM_AHB_INTERFACE_TYPE ==0 ? 4 : (ICM_AHB5_HMASTER_WIDTH ==0 ? 0 : (ICM_AHB5_HMASTER_WIDTH + [::DW_ahb_icm::log2_icm_num_layers ICM_NUM_LAYERS]))}
HMASTER_WIDTH	((ICM_AHB_INTERFACE_TYPE ==1) ? ICM_AHB5_HMASTER_WIDTH : 4)
HPROT_WIDTH	{ICM_HAS_EXTD_MEMTYPE ==1 ? 7 : 4}
HRESP_WIDTH	{AHB_SCALAR_HRESP ==1 ? 1 : 2}
ICM_DP_PWIDTH	{[::DW_ahb_icm::PriorityBusWidth]}
IDLE	2'b00
MID_LAYNO_WIDTH	((ICM_NUM_LAYERS<=2) ? (1 + MID_WIDTH) : ((ICM_NUM_LAYERS<=4) ? (2 + MID_WIDTH) : ((ICM_NUM_LAYERS<=8) ? (3 + MID_WIDTH) : (4 + MID_WIDTH))))

Table A-1 Internal Parameters (Continued)

Parameter Name	Equals To
NONSEQ	2'b10
RETRY	2'b10
SEQ	2'b11
SPLIT	2'b11

B
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 information about the BCMs used in DW_ahb_icm.

B.1 BCM Library Components

Table B-1 describes the list of BCM library components used in DW_ahb_icm.

Table B-1 BCM Library Components

BCM Module Name	BCM Description	DWBB Equivalent
DW_ahb_icm_bcm01	Minimum/Maximum Value	DW_minmax
DW_ahb_icm_bcm02	Universal Multiplexer	DW01_mux_any
DW_ahb_icm_bcm52	Arbiter with Dynamic Priority Scheme	DW_arb_dp

C
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 DesignWare Synthesizable Components.
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.
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.