HB0769
Handbook
CoreAHBL2AHBL_Bridge v2.2

Microsemi
a Microchip company

Microsemi Headquarters
One Enterprise, Aliso Viejo,
CA 92656 USA
Within the USA: +1 (800) 713-4113
Outside the USA: +1 (949) 380-6100
Sales: +1 (949) 380-6136
Fax: +1 (949) 215-4996
Email: sales.support@microsemi.com
www.microsemi.com

About Microsemi

Microsemi, a wholly owned subsidiary of Microchip Technology Inc. (Nasdaq: MCHP), offers a comprehensive portfolio of semiconductor and system solutions for aerospace & defense, communications, data center and industrial markets. Products include high-performance and radiation-hardened analog mixed-signal integrated circuits, FPGAs, SoCs and ASICs; power management products; timing and synchronization devices and precise time solutions, setting the world's standard for time; voice processing devices; RF solutions; discrete components; enterprise storage and communication solutions, security technologies and scalable anti-tamper products; Ethernet solutions; Power-over-Ethernet ICs and midspans; as well as custom design capabilities and services. Learn more at www.microsemi.com.

Contents

1 Revision History . . . 1
1.1 Revision 3.0 . . . 1
1.2 Revision 2.0 . . . 1
1.3 Revision 1.0 . . . 1
2 Introduction . . . 2
2.1 Features . . . 3
2.2 Core Version . . . 3
2.3 Supported Families . . . 3
2.4 Device Utilization and Performance . . . 3
3 Functional Description . . . 4
3.1 Bridge Slave Interface . . . 4
3.2 Bridge Master Interface . . . 4
3.3 CoreAHBL2AHBL_Bridge Use Models . . . 5
4 Interface . . . 6
4.1 Ports . . . 6
4.2 Configuration Parameters . . . 7
5 Timing Diagrams . . . 8
6 Tool Flow . . . 10
6.1 License . . . 10
6.2 RTL . . . 10
6.3 SmartDesign . . . 10
6.4 Configuring CoreAHBL2AHBL_Bridge in SmartDesign . . . 11
6.5 Simulation Flows . . . 11
6.6 Synthesis in Libero . . . 11
6.7 Place-and-Route in Libero . . . 11
6.8 Design Migration . . . 12
6.9 Memory Map Support in Libero SmartDesign . . . 12
7 Test Bench . . . 13
7.1 User Test Bench Use Model 1 . . . 13
7.2 User Test Bench Use Model 2 . . . 13
7.3 Use Case List . . . 14
8 System Integration . . . 15
8.1 Constraints . . . 15

Figures

Figure 1 Top-Level Functional Block Diagram for CoreAHB2AHBL_Bridge for Use Model 1 2
Figure 2 Top-Level Functional Block Diagram for CoreAHB2AHBL_Bridge for Use Model 2 2
Figure 3 CoreAHBL2AHBL_Bridge I/O Signals for Use Model 1 5
Figure 4 CoreAHBL2AHBL_Bridge I/O Signals for Use Model 2 5
Figure 5 Single Write with OKAY Slave Response and Slave HREADY High 8
Figure 6 Incremental Burst Write with OKAY Response and HREADY High 8
Figure 7 Single Read with OKAY Response and HREADY High 9
Figure 8 Incremental Burst Read with OKAY Response & HREADY High 9
Figure 9 SmartDesign CoreAHBL2AHBL_Bridge Instance View 10
Figure 10 SmartDesign CoreAHBL2AHBL_Bridge Configurator 11
Figure 11 CoreAHBL2AHBL_Bridge User Test Bench Use Model 1 13
Figure 12 CoreAHBL2AHBL_Bridge User Test Bench Use Model 2 13
Figure 13 CoreAHBL2AHBL_Bridge System Integration Diagram 15

Tables

Table 1 Device Utilization and Performance . . . 3
Table 2 I/O Signals . . . 6
Table 3 CoreAHBL2AHBL_Bridge Configuration Options . . . 7
Table 4 Parameter Mapping . . . 12
Table 5 Use Case List . . . 14

1 Revision History

The revision history describes the changes that were implemented in the document. The changes are listed by revision, starting with the most current publication.

1.1 Revision 3.0

- Updated for CoreAHBL2AHBL_Bridge v2.2.
- Replaced AHBL_BRIDGE_SEL parameter with two parameters MASTER_BIF_TYPE and SLAVE_BIF_TYPE.
- Added Design Migration section with information required to migrate from v2.1 or lower to v2.2.

1.2 Revision 2.0

Updated for CoreAHBL2AHBL_Bridge v2.1.

1.3 Revision 1.0

The first publication of this document. Created for CoreAHBL2AHBL_Bridge v2.0.

2 Introduction

CoreAHBL2AHBL_Bridge provides solutions for applications where the AHB master and AHB slave operate in two different clock domains that are asynchronous in nature. Its function is to create the link between AHB master transactions going to the AHB slave by functioning as a bridge slave for the AHB master interface and a bridge master for the AHB slave interface

CoreAHBL2AHBL_Bridge manages asynchronous FIFO for handling the different clock domain crossing issue in the design.

Figure 1 • **Top-Level Functional Block Diagram for CoreAHB2AHBL_Bridge for Use Model 1**

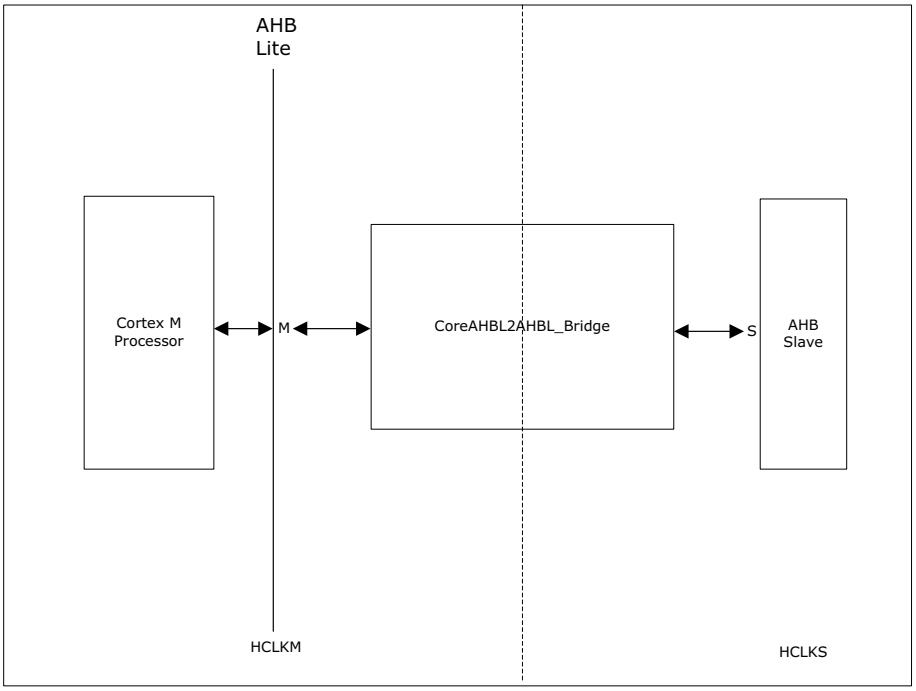
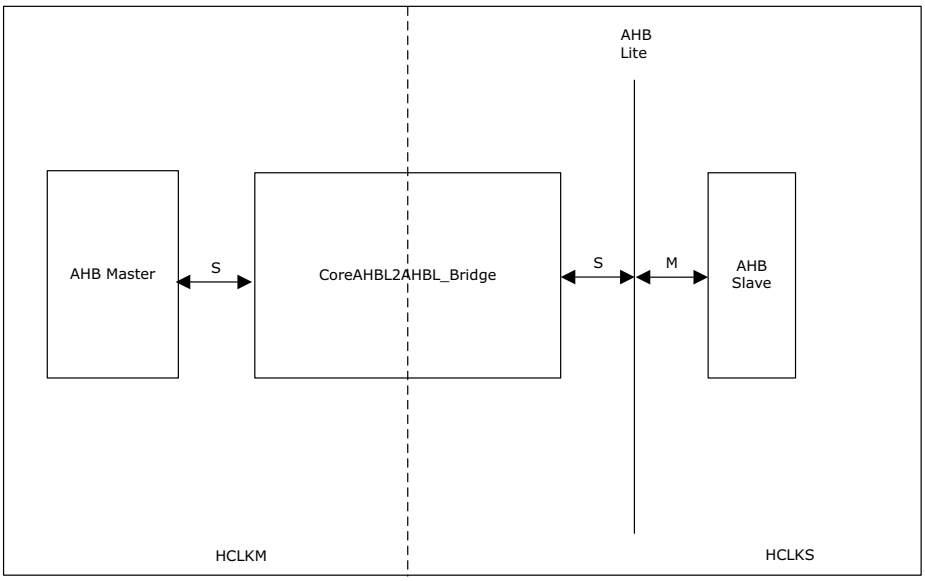


Figure 2 • **Top-Level Functional Block Diagram for CoreAHB2AHBL_Bridge for Use Model 2**



2.1 Features

CoreAHBL2AHBL_Bridge supports the following features:

- Two different asynchronous clock domains for master and slave interfaces
- Single read, single write transactions
- Burst Mode: INCR and WRAP with *busy* transfer type
- Extended HREADY

2.2 Core Version

This handbook is for CoreAHBL2AHBL_Bridge version 2.2.

2.3 Supported Families

- PolarFire® SoC
- PolarFire®
- SmartFusion®2
- IGLOO®2
- RTG4™

2.4 Device Utilization and Performance

Utilization and performance data is listed in the following table for supported device families. The data listed in this table is indicative only. The overall device utilization and performance of the core is system dependent.

Table 1 • **Device Utilization and Performance**

	Parameter	Utilization					Performance	
FAMILY	FIFO_EN	Sequential (DFF)	Combinational (4LUT)	Total	Percentage	RAM Blocks Used	HCLK_M0 Frequency (in MHz)	HCLK_S0 Frequency (in MHz)
SmartFusion2 (M2S150)	0	162	32	194	0.13	0	277	253
	1	405	427	832	0.57	4 RAM64x18	182	204
IGLOO2 (M2GL150)	0	162	32	194	0.13	0	277	253
	1	405	427	832	0.57	4 RAM64x18	182	204
RTG4 (RTG4150)	0	162	34	196	0.13	0	229	219
	1	405	429	834	0.55	4 RAM64x18_RT	120	120
PolarFire (MPF500T)	0	162	32	194	0.04	0	336	322
	1	333	349	682	0.14	6 RAM64x12	223	235
PolarFire SoC (MPFS460T)	0	162	32	194	0.04	0	336	322
	1	333	349	682	0.15	6 RAM64x12	223	235

Note: The data in Table 1 is achieved using Verilog RTL, typical synthesis, and layout settings. Frequency (in MHz) was set to 100, and speed grade was -1. The parameters MASTER_BIF_TYPE is set to 0, SLAVE_BIF_TYPE is set to 0, and SYNC_CLOCK is set to 1.

3 Functional Description

This section provides a detailed description of the CoreAHBL2AHBL_Bridge IP slave and master interfaces.

3.1 Bridge Slave Interface

When FIFO_EN =1

- The bridge slave accepts transfer for READ & WRITE from the AHBLite master.
- For write transaction, write data coming from the AHB Master is stored in the write data FIFO.
- For read transaction, data is read from the read data FIFO and sent to the master.

When FIFO_EN=0

- The bridge slave accepts transfer for READ & WRITE from the AHBLite master. A request pulse is generated for READ and WRITE transfers
- For write transaction, write data coming from AHB master is sent using pulse synchronizer
- For read transaction, using the acknowledge pulse the data is read and sent to master.

3.2 Bridge Master Interface

When FIFO_EN =1

- The bridge master transfers the control information to the slave.
- For write transaction, the bridge master reads the data from write FIFO and sends it to the slave
- For read transaction, the bridge master accepts the data coming from the slave and stores it in the read data FIFO.

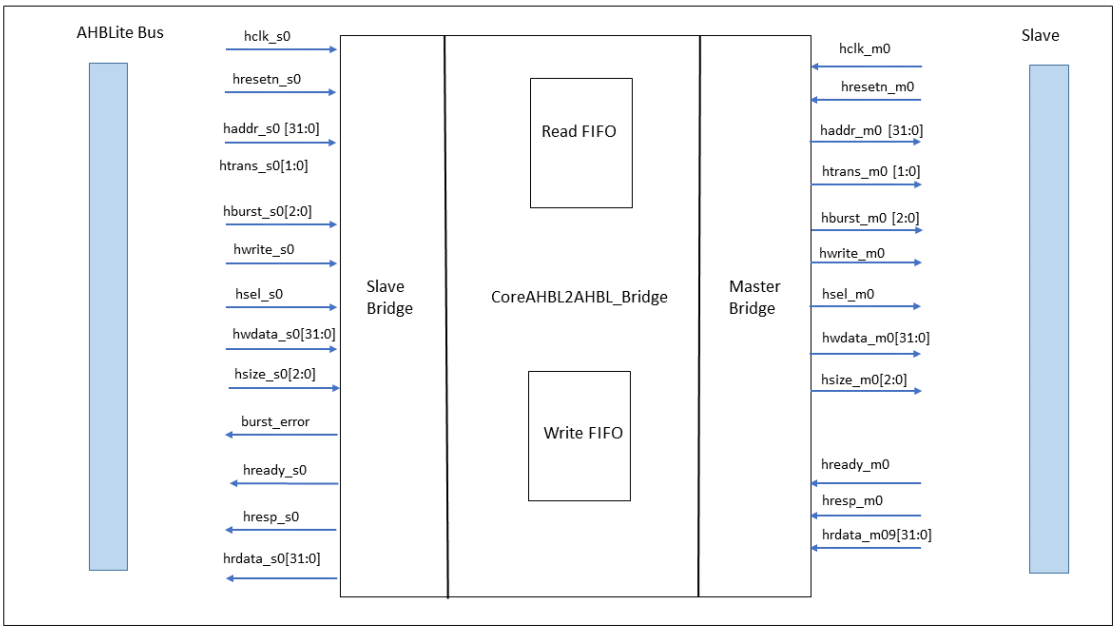
When FIFO_EN=0

- The bridge master transfers the control information to the slave.
- For write transaction, after receiving the request pulse, the bridge master sends the write data to the slave.
- For read transaction, the bridge master accepts the read data coming from the slave and sent to the master using pulse synchronizer.

3.3 CoreAHBL2AHBL_Bridge Use Models

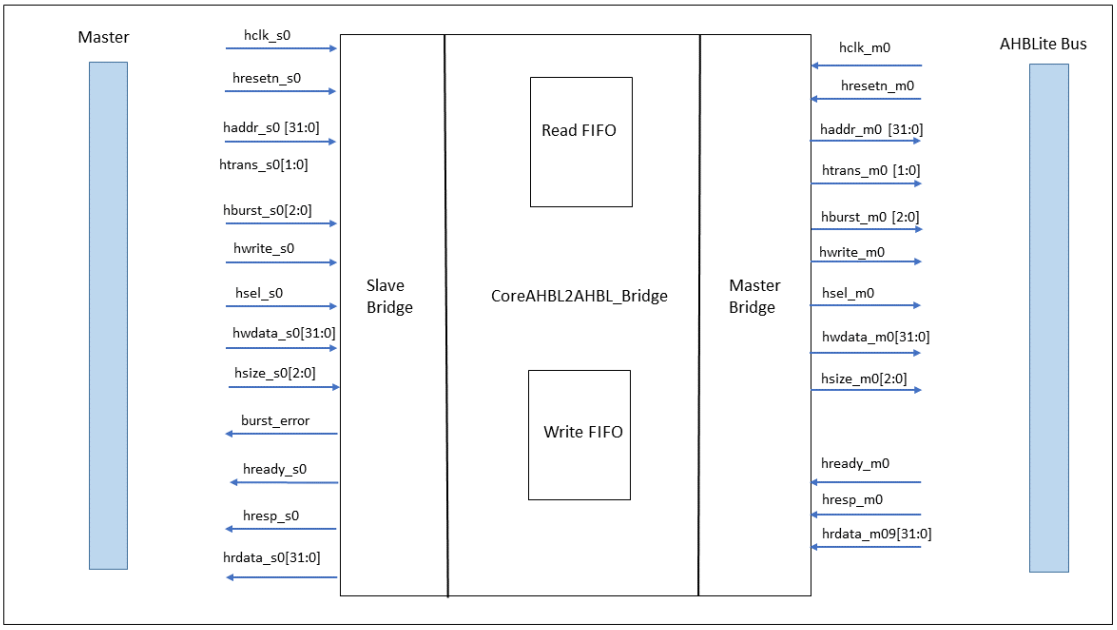
The following figure illustrates the use model where AHBL_BRIDGE_SEL = 1. In this use model, the core is used to connect the AHBLite bus interface with the AHBLite slave device.

Figure 3 • **CoreAHBL2AHBL_Bridge I/O Signals for Use Model 1**



The following figure illustrates the use model where AHBL_BRIDGE_SEL = 0. In this use model, the core is used to connect the AHBL master with the AHBLite bus interface.

Figure 4 • **CoreAHBL2AHBL_Bridge I/O Signals for Use Model 2**



4 Interface

4.1 Ports

The following table describes the CoreAHBL2AHBL_Bridge I/O signals.

Table 2 • **I/O Signals**

Port	Direction	Description
Clocks and Resets		
HCLK_S0	Input	Bridge slave input clock
HRESETN_S0	Input	Bridge slave input reset
HRESETN_M0	Input	Bridge master input reset
HCLK_M0	Input	Bridge master input clock
Bridge Slave IF		
HADDR_S0 [31:0]	Input	Address bus for the slave
HBURST_S0 [2:0]	Input	Burst type indication
HSEL_S0	Input	Slave select
HSIZE_S0 [1:0]	Input	Transfer size
HTRANS_S0 [1:0]	Input	Transfer type
HWDATA_S0 [31:0]	Input	Write data bus from the AHB master to the bridge slave
HWRITE_S0	Input	Write indication for the slave
HREADYOUT_S0	Output	Ready signal to the AHB master from the bridge slave
HRESP_S0	Output	Response signal to the AHB master from the bridge slave
HRDATA_S0 [31:0]	Output	Read data bus to the AHB master from the bridge slave
Bridge Master IF		
HRDATA_M0 [31:0]	Input	Read data from the AHB slave
HREADYOUT_M0	Input	Ready signal output from the AHB slave
HRESP_M0	Input	Response signal output from the AHB slave
HADDR_M0 [31:0]	Output	Address bus from the bridge master to the AHB slave
HBURST_M0 [2:0]	Output	Burst type information from the bridge master to the AHB slave
HREADY_M0	Output	Ready indication from the bridge master to the AHB slave
HSEL_M0	Output	AHB slave selection
HSIZE_M0 [2:0]	Output	AHB transfer size to the AHB slave
HTRANS_M0 [1:0]	Output	Transfer type from the bridge master to the AHB slave
HWDATA_M0 [31:0]	Output	AHB write data bus from the bridge master to bridge slave
HWRITE_M0	Output	Write indication to the AHB slave
Error Response		
BURST_ERROR	Output	Indicates burst transfer status. **Note:** Valid only when FIFO_EN = 1.

4.2 Configuration Parameters

The following table shows the configurable parameters for CoreAHBL2AHBL_Bridge. If a setting other than the default is required, use the configuration dialog box in SmartDesign to select appropriate values for the configurable options.

The AHBL_BRIDGE_SEL parameter present in CoreAHBL2AHBL_Bridge v2.1 or lower versions is replaced with new set of parameters MASTER_BIF_TYPE and SLAVE_BIF_TYPE in CoreAHBL2AHBL_Bridge v2.2. For more information about mapping from old parameters to new parameters, refer to Design Migration, page 12.

Table 3 • **CoreAHBL2AHBL_Bridge Configuration Options**

Parameter Name	Valid Range	Default	Description
MASTER_BIF_TYPE	0 or 1	0	Select the AHB BIF type for AHBL Master side Interface when the core is instantiated in the Libero SmartDesign. 0 - AHB Mirror Master BIF 1 - AHB Slave BIF **Note:** Memory map support will be available only when MASTER_BIF_TYPE = 1 and SLAVE_BIF_TYPE = 1.
SLAVE_BIF_TYPE	0 or 1	1	Select the AHB BIF type for AHBL Slave side Interface when the core is instantiated in the Libero SmartDesign. 0 - AHB Mirror Slave BIF 1 - AHB Master BIF **Note:** Memory map support will be available only when MASTER_BIF_TYPE = 1 and SLAVE_BIF_TYPE = 1.
SYNC_CLOCK	0 or 1	0	0 - Indicates asynchronous HCLK_S0 and HCLK_M0 clocks 1 - Indicates synchronous HCLK_S0 and HCLK_M0 clocks. **Note:** Clocks can have different frequency and known phase.
FIFO_EN	0 or 1	0	0 - The data transfer is done using handshake mechanism. This utilises less resources. 1 - The data transfer is done using FIFO. This improves the performance when performing burst transactions. **Note:** If FIFO_EN is set to 1 and SYNC_CLOCK is set to 1, then synchronous FIFO is used. **Note:** If FIFO_EN is set to 1 and SYNC_CLOCK is set to 0, then asynchronous FIFO is used. **Note:** It is recommended to use FIFO_EN = 1, when performing multiple burst transaction.

5 Timing Diagrams

This section provides the timing diagrams for CoreAHBL2AHBL_Bridge.

Figure 5 • **Single Write with OKAY Slave Response and Slave HREADY High**

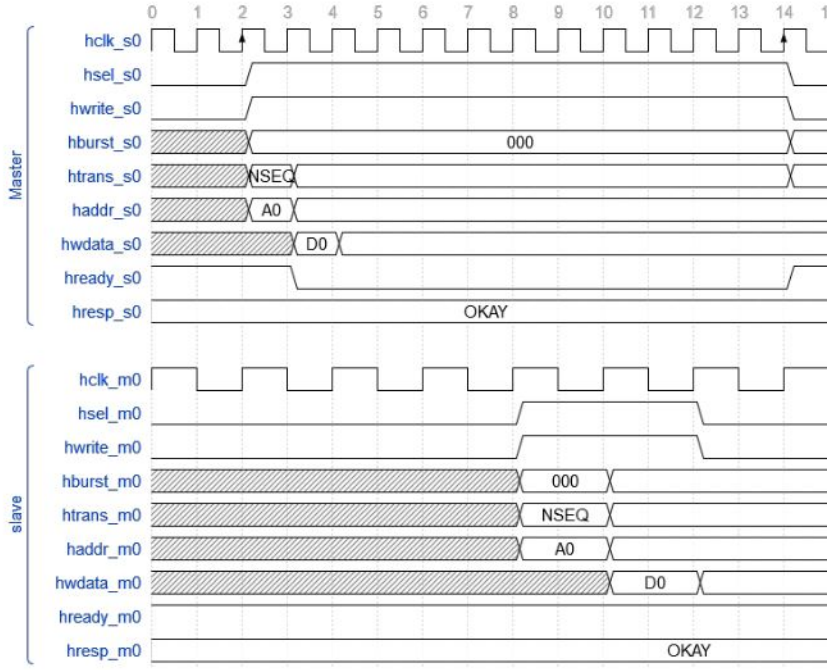


Figure 6 • **Incremental Burst Write with OKAY Response and HREADY High**

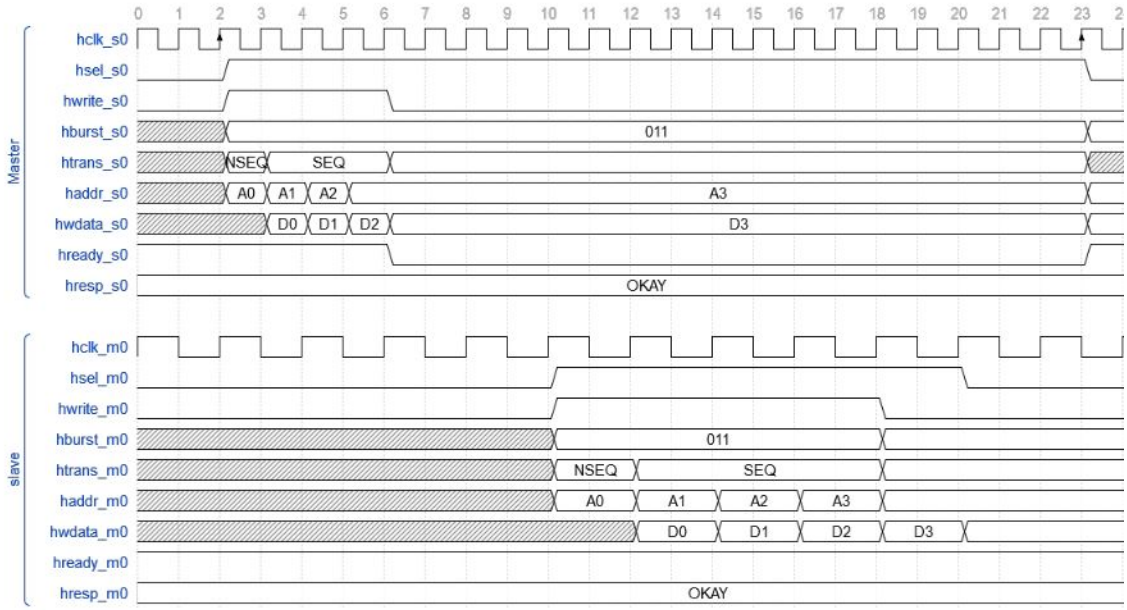


Figure 7 • **Single Read with OKAY Response and HREADY High**

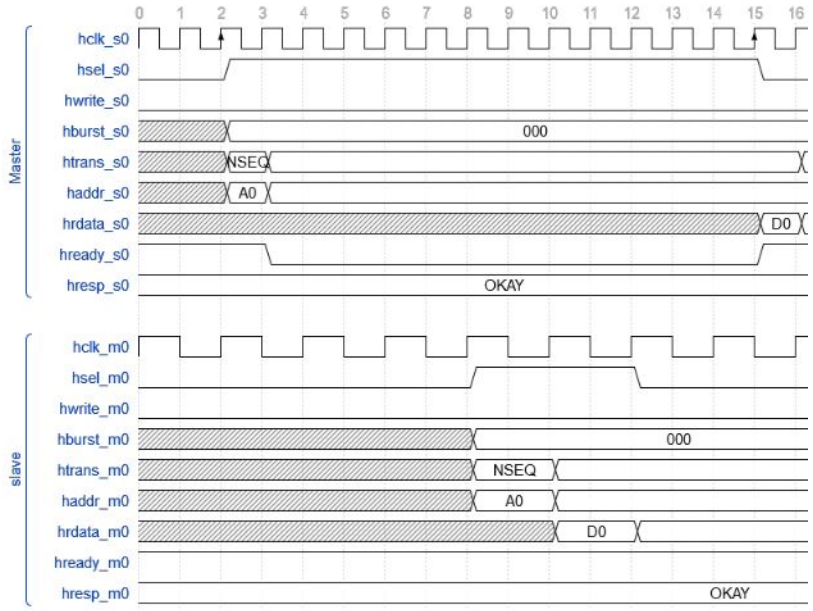
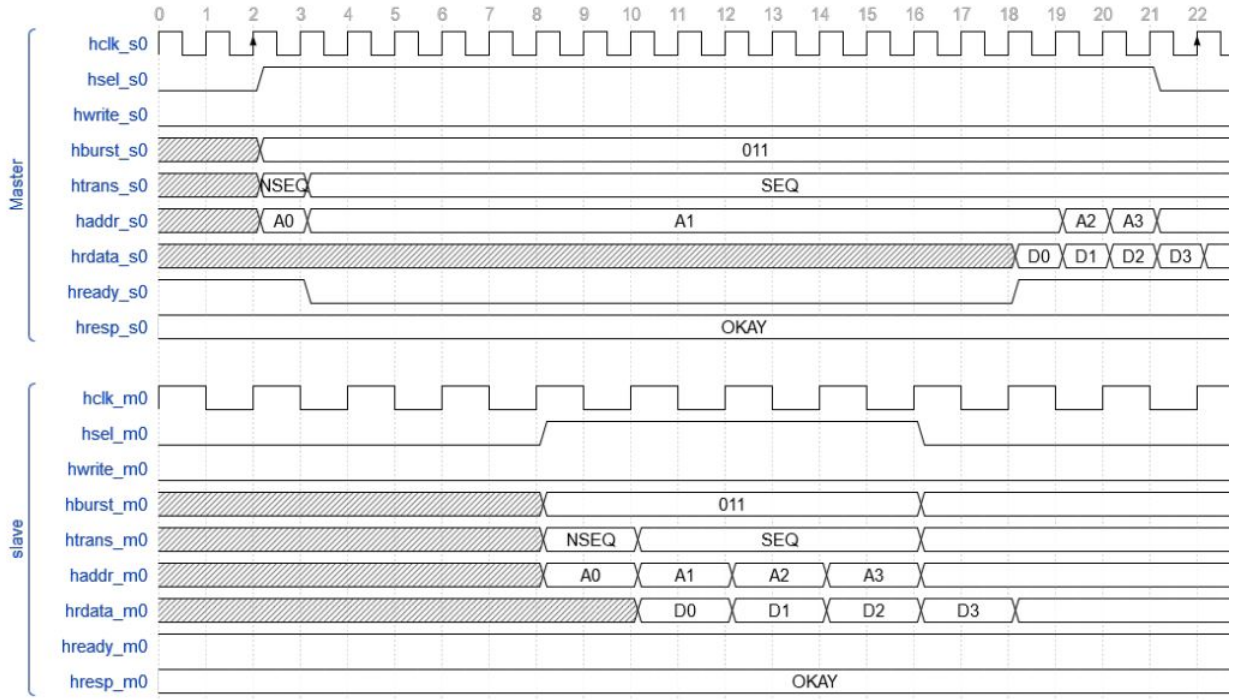


Figure 8 • **Incremental Burst Read with OKAY Response & HREADY High**



6 Tool Flow

6.1 License

CoreAHBL2AHBL_Bridge does not require a license.

6.2 RTL

The complete RTL source code is provided for the core.

6.3 SmartDesign

CoreAHBL2AHBL_Bridge is available for download in the Libero® SoC IP catalog through the web repository. Once it is listed in the catalog, the core can be instantiated using the SmartDesign flow. For information about using SmartDesign to configure, connect, and generate cores, refer to the Libero SoC online help. An example instantiated view is shown in the following figure.

After configuring and generating the core instance, the basic functionality can be simulated using the test bench provided with the CoreAHBL2AHBL_Bridge. The testbench parameters automatically adjust to the CoreAHBL2AHBL_Bridge configuration. The CoreAHBL2AHBL_Bridge can be instantiated as a component of a larger design.

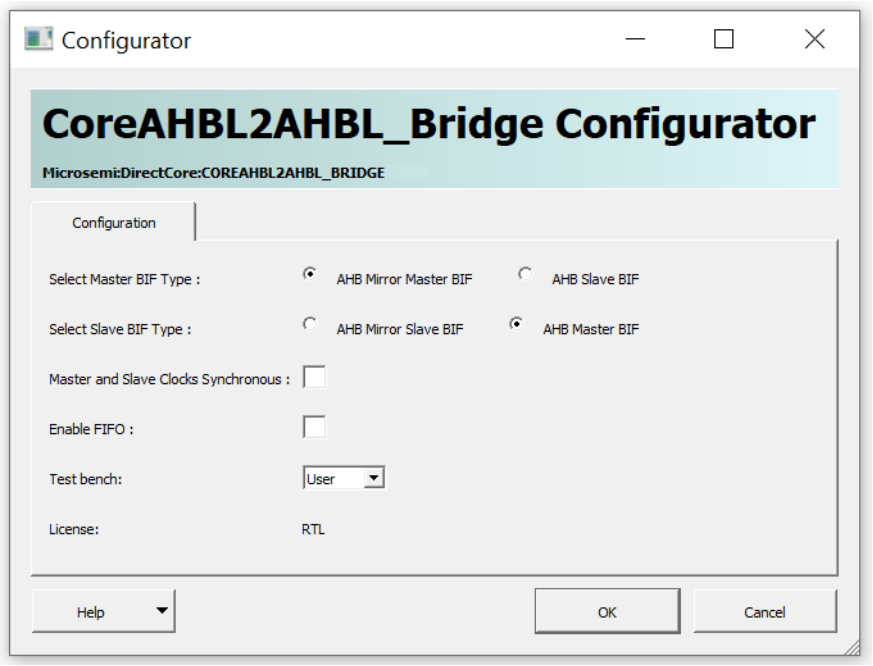
CoreAHBL2AHBL_Bridge is compatible with Libero SoC.

Figure 9 • **SmartDesign CoreAHBL2AHBL_Bridge Instance View**

6.4 Configuring CoreAHBL2AHBL_Bridge in SmartDesign

The following figure shows the CoreAHBL2AHBL_Bridge Configurator in SmartDesign.

Figure 10 • **SmartDesign CoreAHBL2AHBL_Bridge Configurator**



6.5 Simulation Flows

To run simulations, select the user test bench in the core configuration window. After generating the CoreAHBL2AHBL_Bridge, the pre-synthesis test bench hardware description language (HDL) files are installed in Libero SoC.

6.6 Synthesis in Libero

To run the synthesis on CoreAHBL2AHBL_Bridge, set the SmartDesign and click Synthesis in Libero SoC. The Synthesis window displays the Synplify® project. To run the synthesis, click **Run**.

6.7 Place-and-Route in Libero

After the design is synthesized, run the compilation and then place-and-route the tools. CoreAHBL2AHBL_Bridge requires no specific place-and-route settings.

6.8 Design Migration

This section provides information required to migrate from CoreAHBL2AHBL_Bridge v2.1 or lower versions to CoreAHBL2AHBL_Bridge v2.2

The below Table provides the mapping between the old parameter (AHBL_BRIDGE_SEL in v2.1 or lower) and the new parameters (MASTER_BIF_TYPE and SLAVE_BIF_TYPE in v2.2) of the core.

Table 4 • **Parameter Mapping**

	Old Parameter	New Parameters	
Configuration	**AHBL_BRIDGE_SEL**	**MASTER_BIF_TYPE**	**SLAVE_BIF_TYPE**
Master-to-Slave Path	0	0	1
Slave-to-Master Path	1	1	0

Note: The remaining two combinations of MASTER_BIF_TYPE and SLAVE_BIF_TYPE is also valid. These combinations are listed below.

- MASTER_BIF_TYPE = 0 and SLAVE_BIF_TYPE = 0
- MASTER_BIF_TYPE = 1 and SLAVE_BIF_TYPE = 1

6.9 Memory Map Support in Libero SmartDesign

Memory map support will be available only when MASTER_BIF_TYPE = 1 and SLAVE_BIF_TYPE = 1.

Memory map support will not be available when mirror BIF is selected for either AHBL Bridge Master side interface or AHBL Bridge Slave side interface.

7 Test Bench

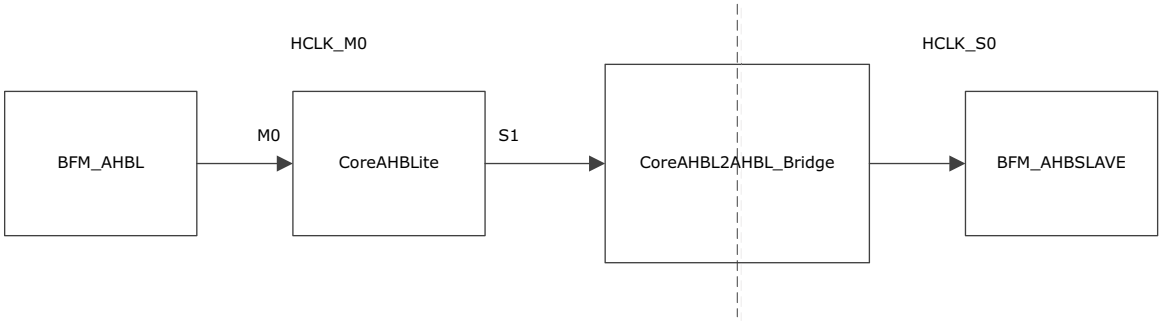
A unified test bench, referred to as the user test bench, is used to verify and test CoreAHBL2AHBL_Bridge.

7.1 User Test Bench Use Model 1

The user test bench is included with each release of CoreAHBL2AHBL_Bridge to verify the CoreAHBL2AHBL_Bridge features.

In this use model, CoreAHBLite bus is running at the same clock as the master whereas the slave is running on a different clock, as shown in figure.

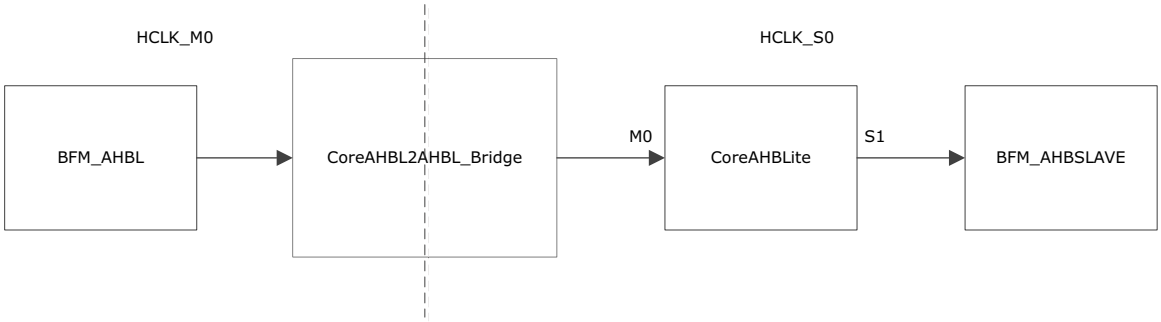
Figure 11 • **CoreAHBL2AHBL_Bridge User Test Bench Use Model 1**



7.2 User Test Bench Use Model 2

In this use model, CoreAHBLite is running at same frequency as its slaves, but master is running on a different clock.

Figure 12 • **CoreAHBL2AHBL_Bridge User Test Bench Use Model 2**



7.3 Use Case List

Table 5 • **Use Case List**

	Normal		Extended		Error Response	
	Read	**Write**	**Read**	**Write**	**Read**	**Write**
Single	single_rd	single_wr	single_ext_rd	single_ext_wr	single_rd_err	single_wr_err
INCR Burst	incr_rd	incr_wr	incr_ext_rd	incr_ext_wr	incr_rd_err	incr_wr_err
INCR4 Burst	incr4_rd	incr4_wr	incr4_ext_rd	incr4_ext_wr	incr4_rd_err	incr4_wr_err
INCR8 Burst	incr8_rd	incr8_wr	incr8_ext_rd	incr8_ext_wr	incr8_rd_err	incr8_wr_err
INCR16 Burst	incr16_rd	incr16_wr	incr16_ext_rd	incr16_ext_wr	incr16_rd_err	incr16_wr_err

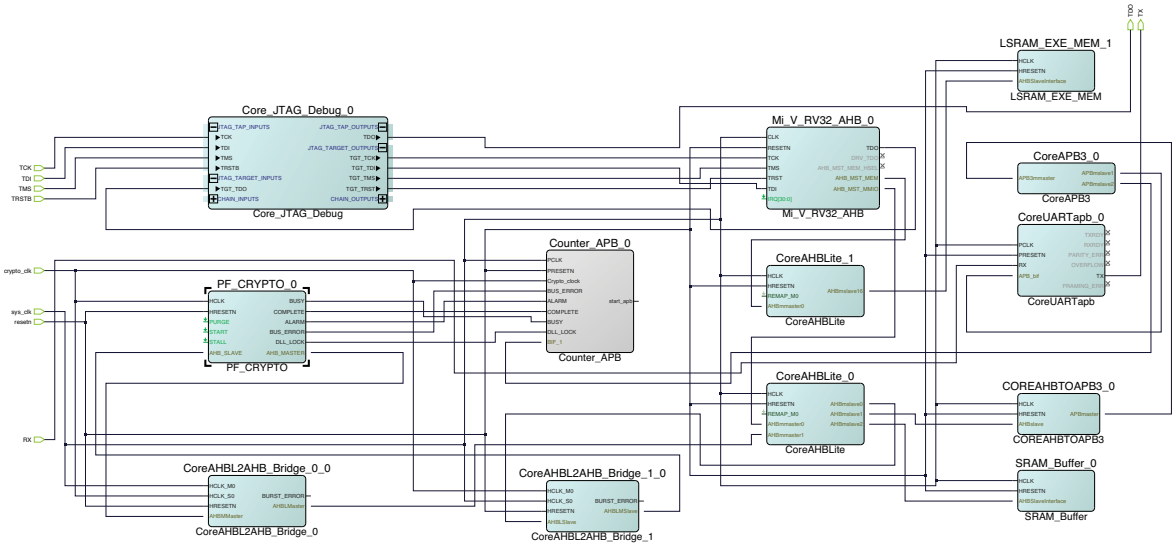
Note: The user test bench is configured for FIFO_EN = 0.

8 System Integration

The following is an example system integration diagram for CoreAHBL2AHBL_Bridge. This example design describes the use of CoreAHBL2AHBL_Bridge IP connected between:

- The AHB master interface of the Athena cryptoprocessor and CoreAHBLite.
- CoreAHBLite and the AHB slave interface of Athena cryptoprocessor.

Figure 13 • **CoreAHBL2AHBL_Bridge System Integration Diagram**



8.1 Constraints

The following constraints are provided for CoreAHBL2AHBL_Bridge v2.2:

- Clock constraints
  - create_clock -name {HCLK_M0} -period 3 -waveform {0 1.5} [get_ports {hclk_m0}]
  - create_clock -name {HCLK_S0} -period 3 -waveform {0 1.5} [get_ports {hclk_s0}]
- False path constraints
  - set_false_path -from [get_pins {*U_bridge_master/mclk_*}] -to [get_pins {*U_bridge_slave/sclk_*}]
  - set_false_path -from [get_pins {*U_bridge_slave/sclk_*}] -to [get_pins {*U_bridge_master/mclk_*}]
  - set_false_path -from [get_pins {*U_bridge_master/mclk_hrdata_m0}] -to [get_ports {HRDATA_S0}]
  - set_false_path -from [get_ports HWDATA_S0] -to [get_pins{*U_bridge_master/hwdata_m0}]