HB0769
Handbook
CoreAHBL2AHBL_Bridge v2.0

Microsemi
Power Matters.

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

About Microsemi

Microsemi Corporation (Nasdaq: MSCC) 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. Microsemi is headquartered in Aliso Viejo, California, and has approximately 4,800 employees globally. Learn more at www.microsemi.com.

50200769. 1.0 4/17

Contents

Figures

Tables

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 1.0

Revision 1.0 is the first publication of this document.

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**

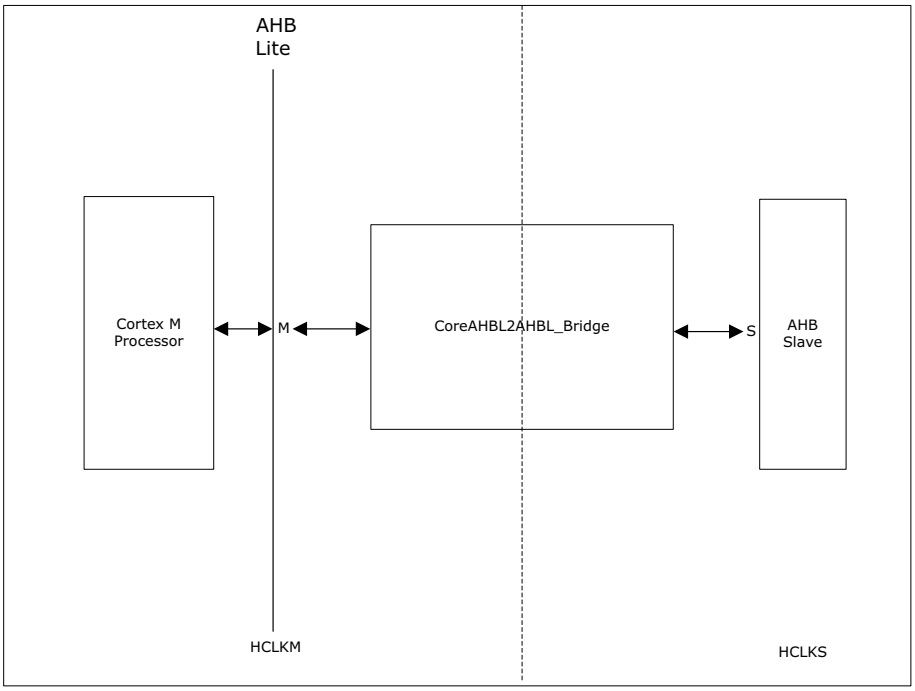
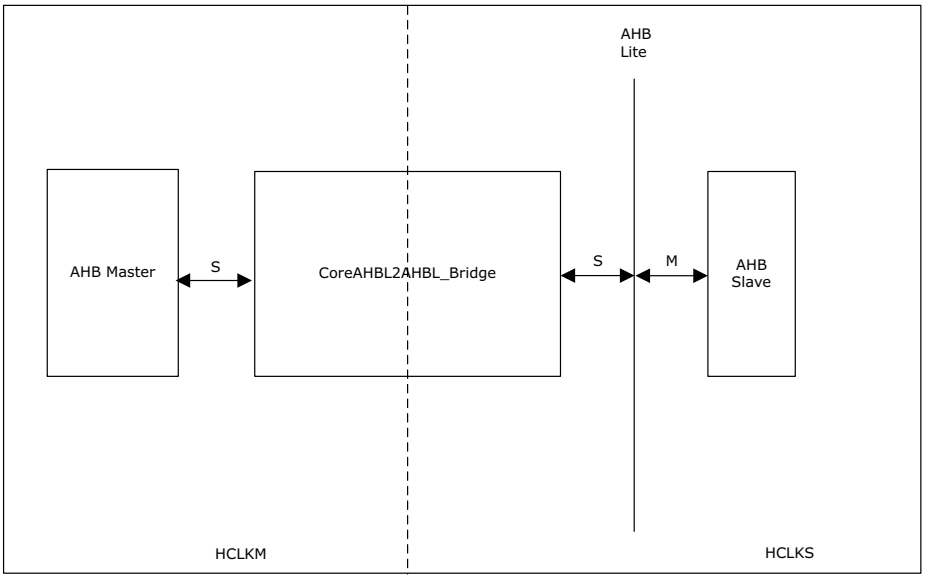


Figure 2 • **Top-Level Functional Block Diagram for CoreAHB2AHBL_Bridge for Use Model 2**



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, INCR4, INCR8,INCR16 with *busy* transfer type
- Extended HREADY

2.2 Core Version

This handbook is for CoreAHBL2AHBL_Bridge version 2.0.

2.3 Supported Families

PolarFire™

2.4 Device Utilization and Performance

The following table summarizes the utilization data for CoreAHBL2AHBL_Bridge under the following conditions:

- Speed grade: STD
- Core voltage: 1.2 V
- Operating condition: IND

Table 1 • **CoreAHBL2AHBL_Bridge Device Utilization and Performance**

FPGA Family	Device	FPGA Resources			Utilization	Clock Rate (MHz)
		Combinatorial	Sequential	Total		
PolarFire	MPF300TS-ES	~588	~653	~1200	~.4%	262 Mhz

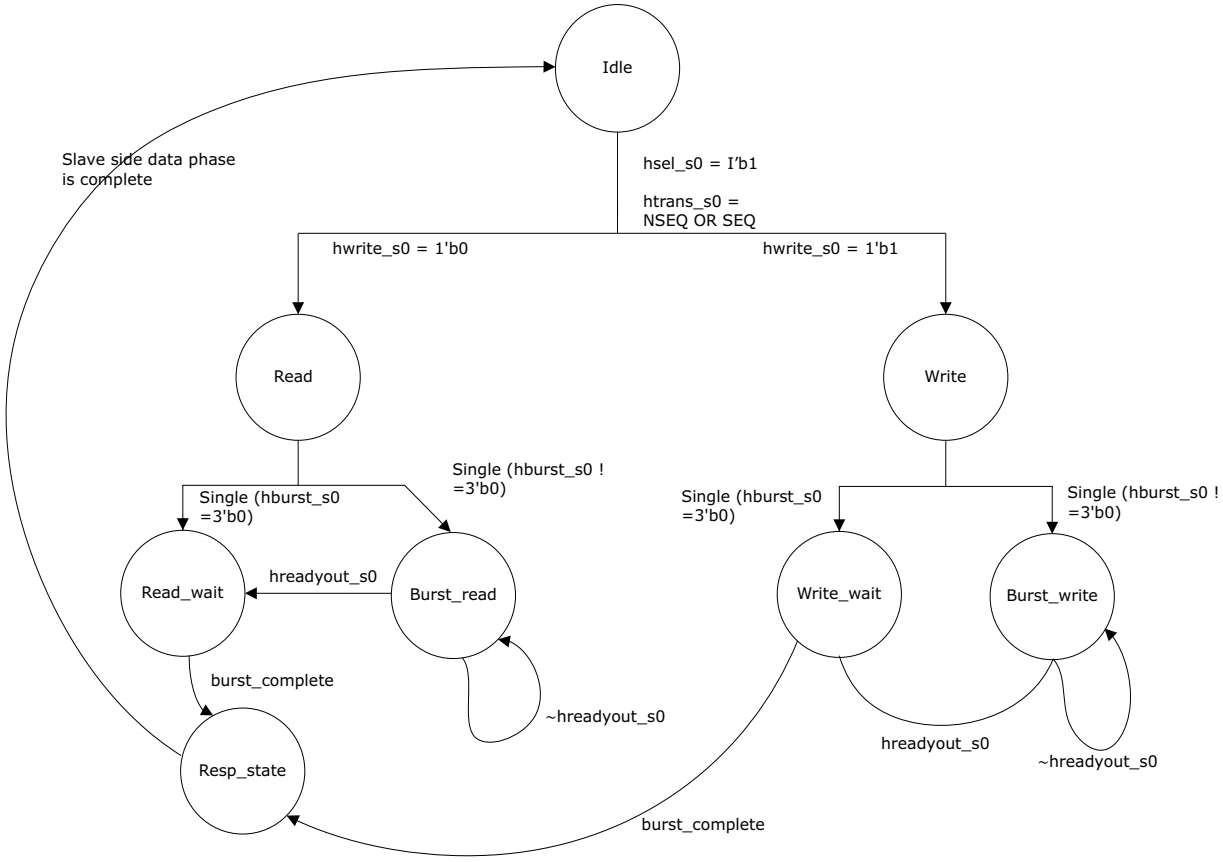
3 Functional Description

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

3.1 Bridge Slave Interface

- The bridge slave accepts transfers for READ & WRITE from the AHB Lite Master. Control information is stored in the control FIFO.
- For write transactions, write data coming from the AHB Master is stored in the write data FIFO.
- For read transactions, data is read from the read data FIFO and sent to the AHB master.

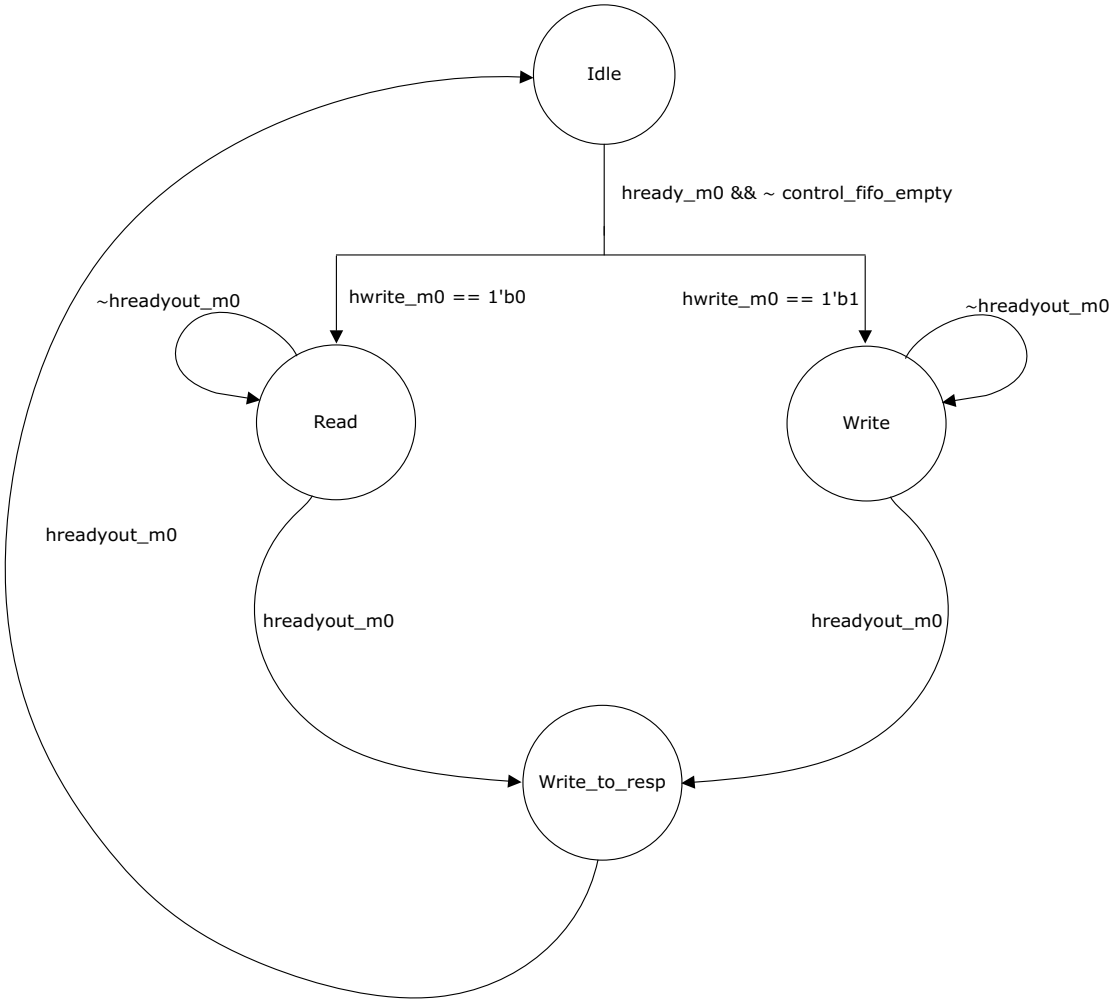
Figure 3 • **Bridge Slave State Machine**



3.2 Bridge Master Interface

- The bridge master reads the control FIFO information whenever the HREADY from the slave is HIGH and the control FIFO is non-empty in the slave clock domain.
- The bridge master transfers the control information to the Slave
- For write transactions, the bridge master reads the data from write data FIFO and sends it to the slave.
- For read transactions, the bridge master accepts the data coming from the slave and stores it in the read data FIFO.
- The bridge master stores the slave response and also the complete burst information.

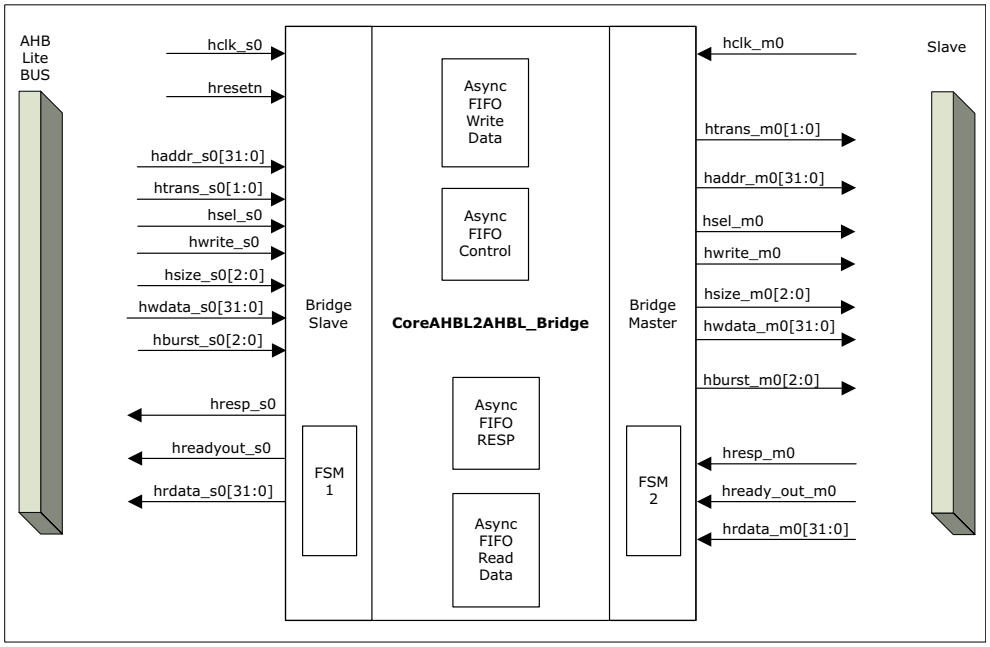
Figure 4 • **Bridge Master State Machine**



3.3 CoreAHBL2AHBL_Bridge Use Models

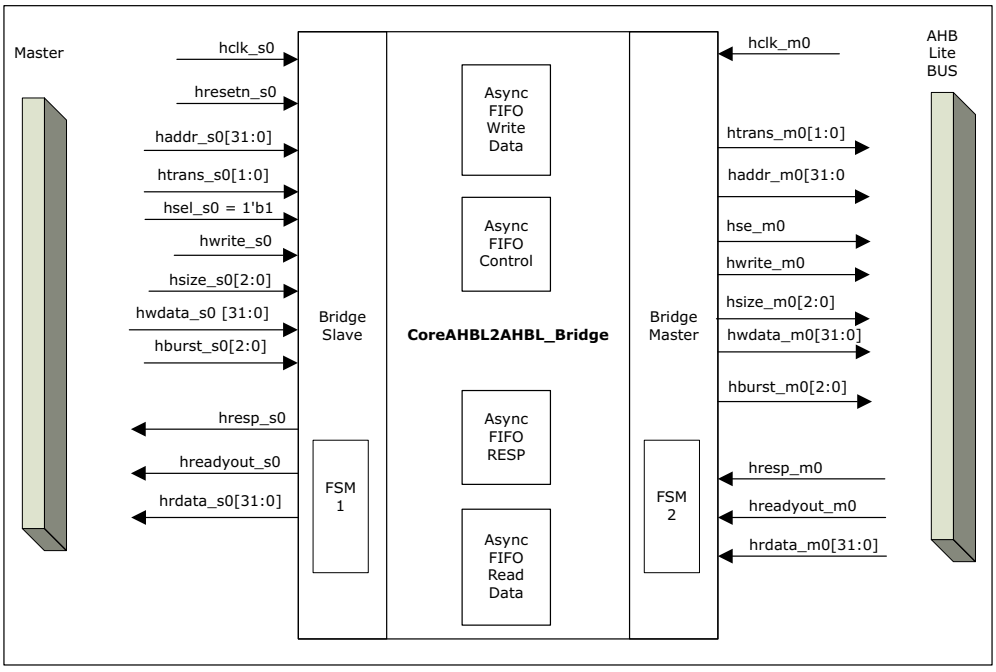
The following figure illustrates the use model where AHBL_BRIDGE_SEL = 1. In this use model, the core is used to connect the AHBLite bus interface with the AHBLite slave device.

Figure 5 • **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 6 • **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	Input	Asynchronous active-low 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
hready_s0	Input	Ready signal to 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

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.

Table 3 • **CoreAHBL2AHBL_Bridge Configuration Options**

Parameter Name	Valid Range	Default	Description
AHBL_BRIDGE_SEL	0–1	0	Select the bridge mode. The bridge can be connected in the following two ways: 0: Master-to-slave path. Used to connect the bridge between AHB master and AHBLite bus. 1: Slave-to-master path. Used to connect the bridge between AHBLite bus and AHB slave.

5 Timing Diagrams

This section provides the timing diagrams for CoreAHBL2AHBL_Bridge.

Figure 7 • **Single Write with OKAY Slave Response and Slave HREADY High**

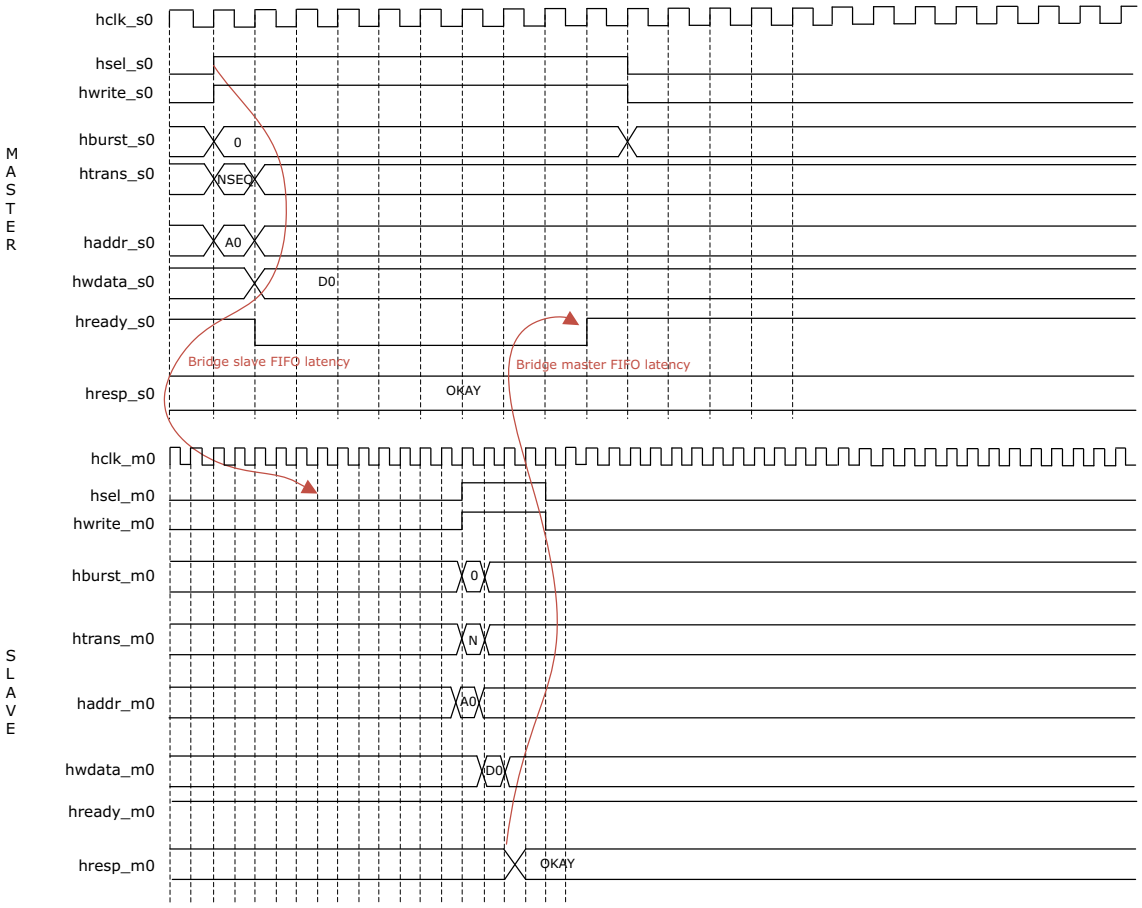


Figure 8 • **Incremental Burst Write with OKAY Response and HREADY High**

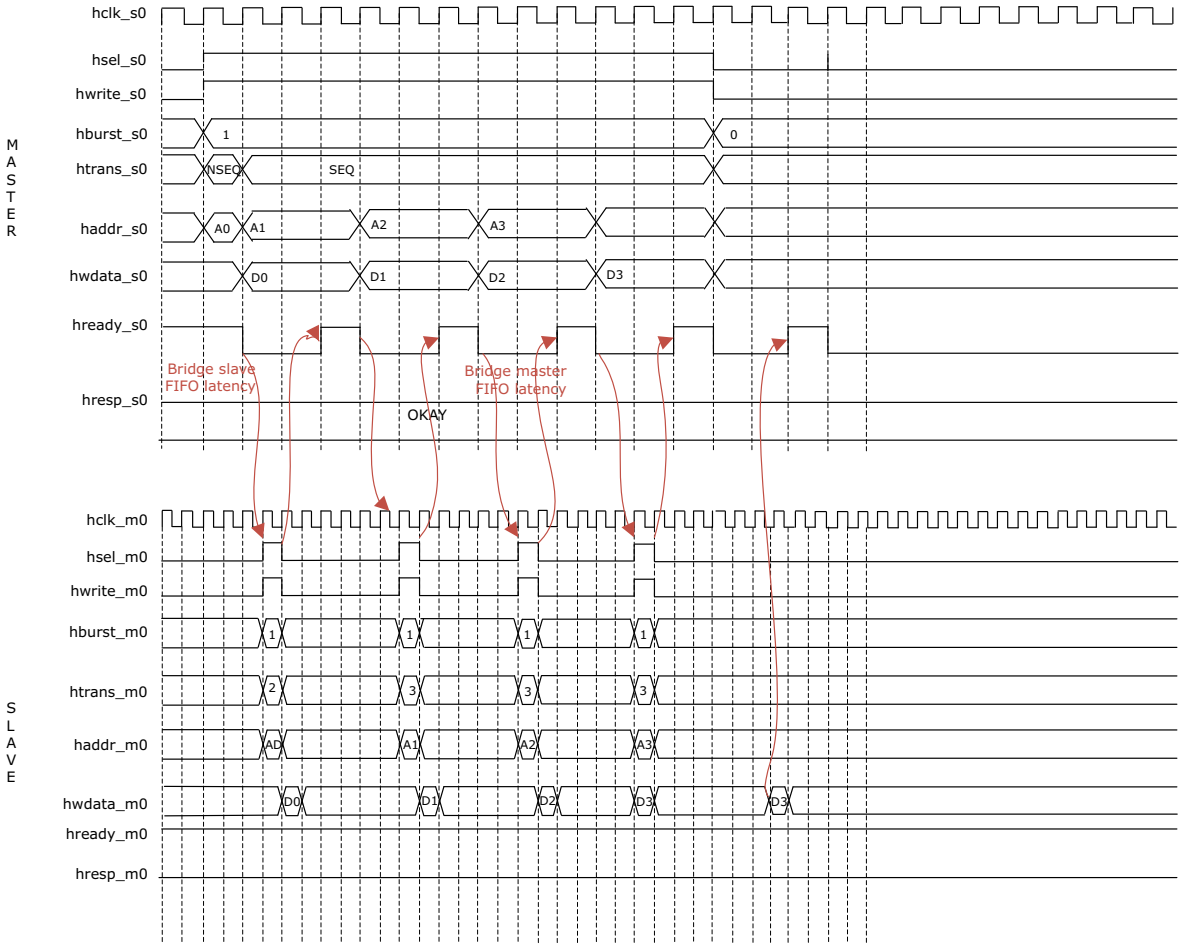


Figure 9 • **Single Read with OKAY Response and HREADY High**

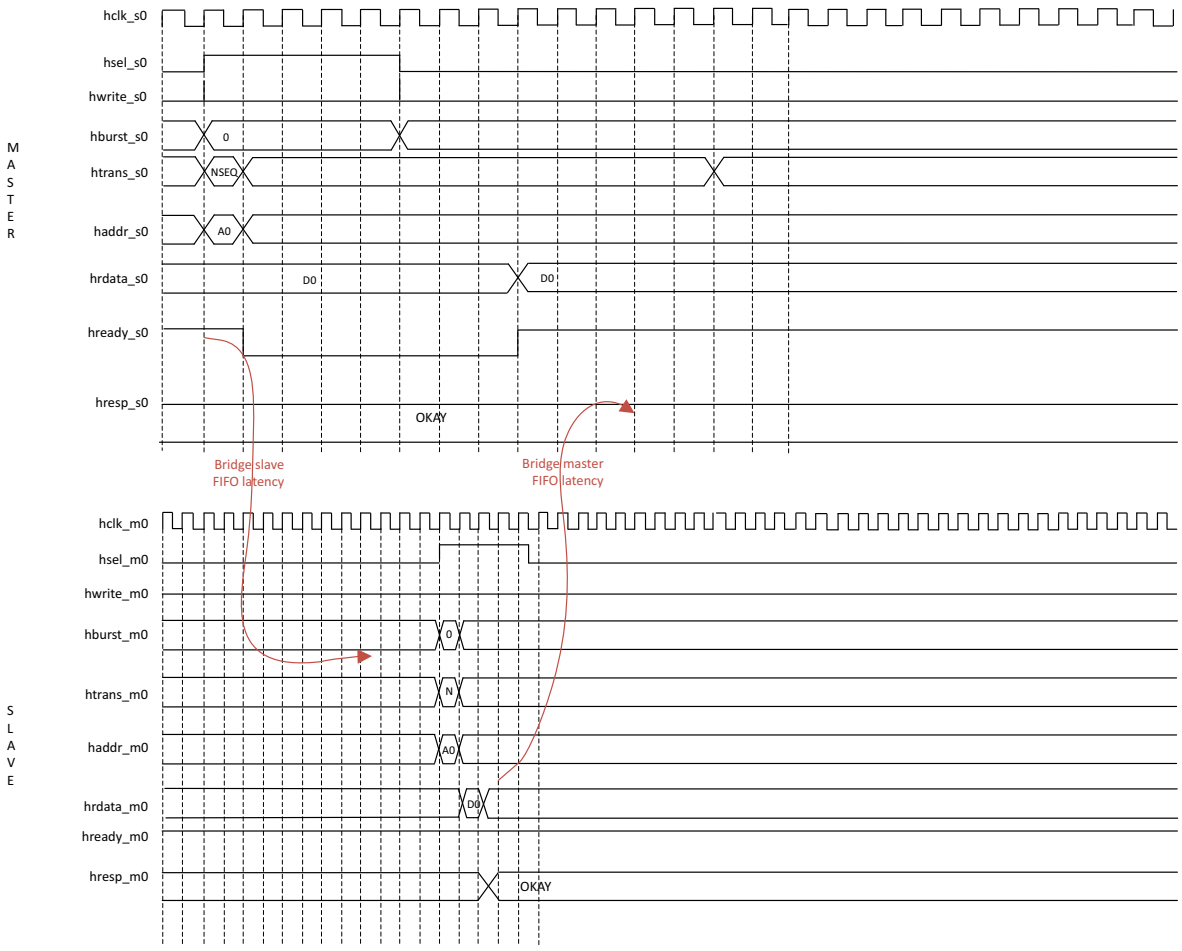


Figure 10 • **Single Read with OKAY Response and HREADY Low for One Clock Cycle**

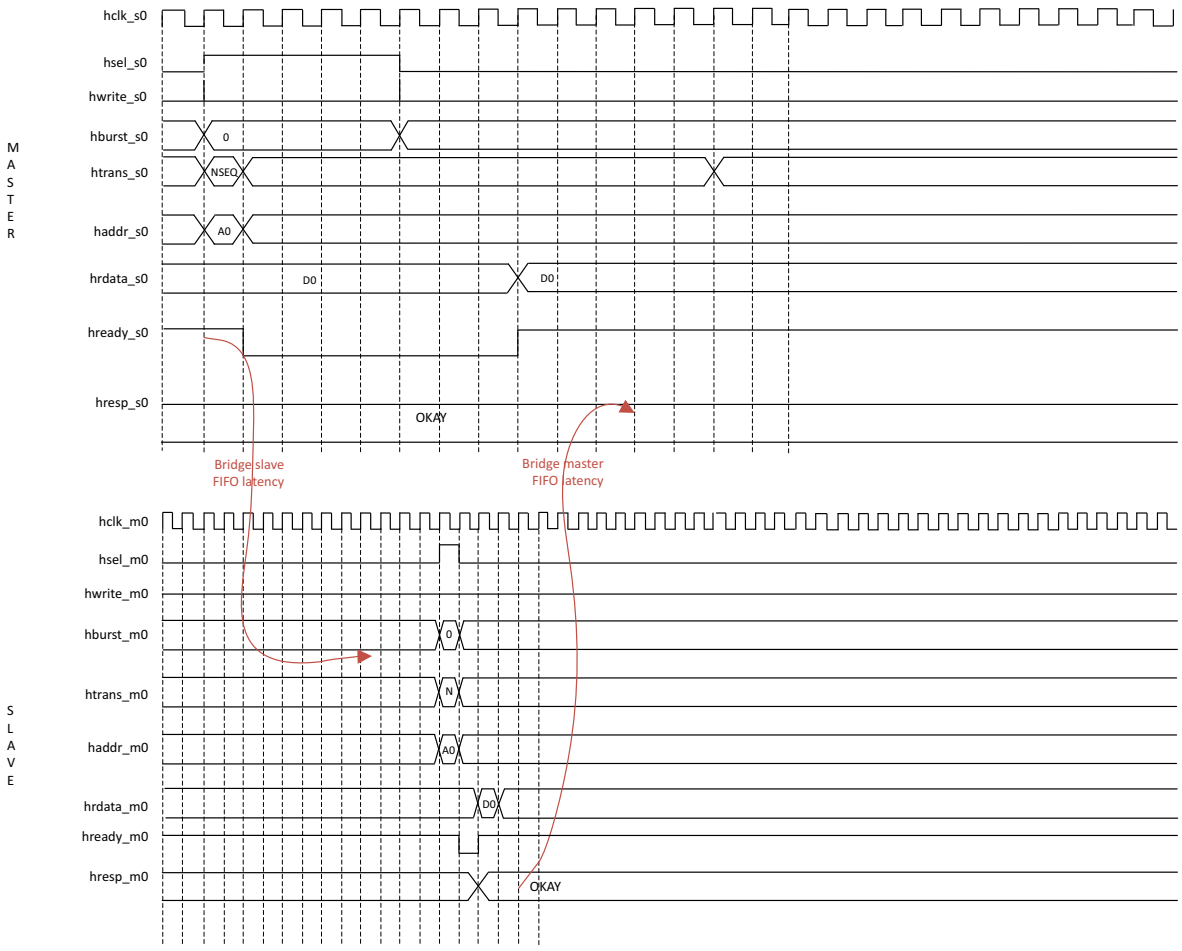
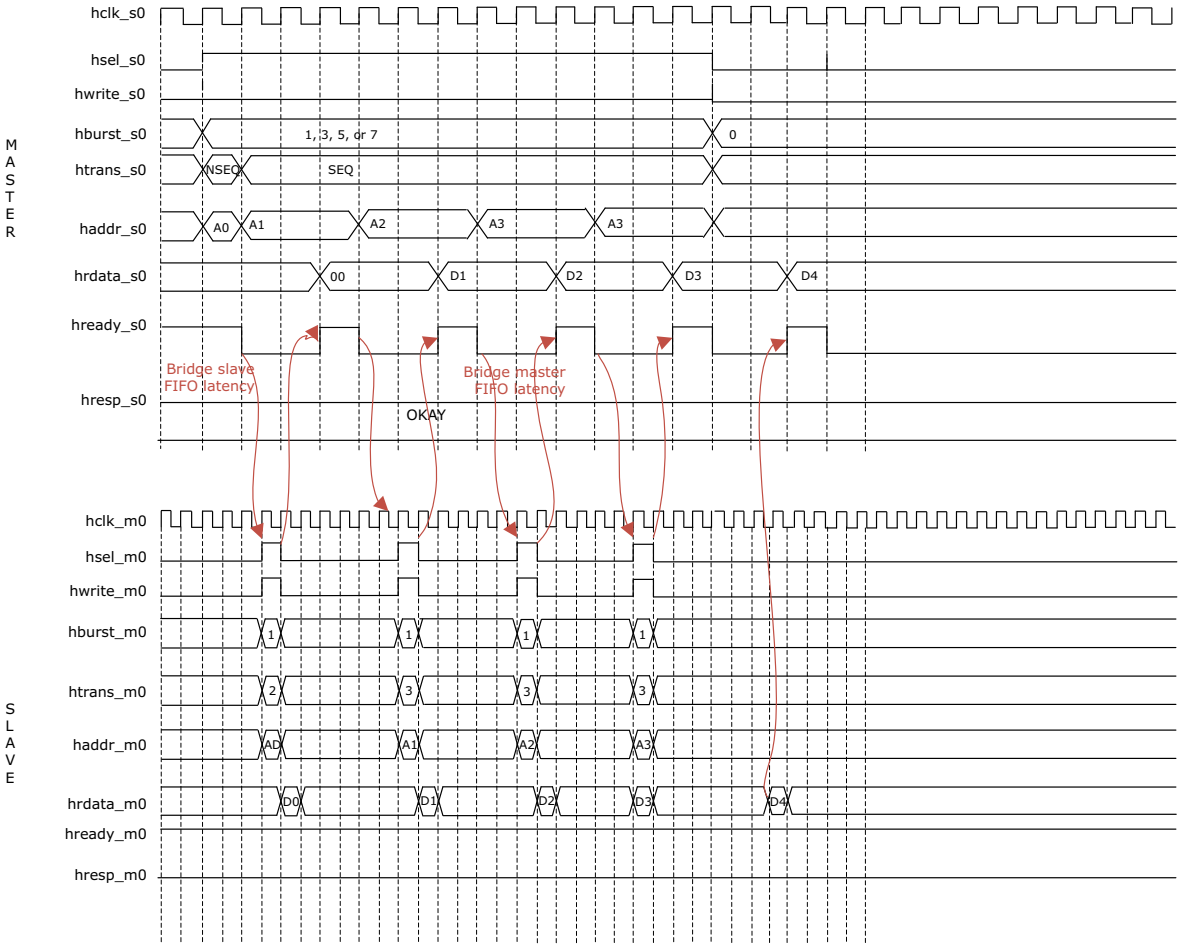


Figure 11 • **Incremental Burst Read with OKAY Response & HREADY High**



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.

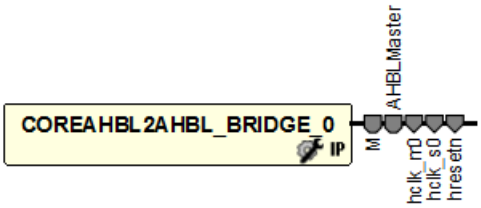
6.3 SmartDesign

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.

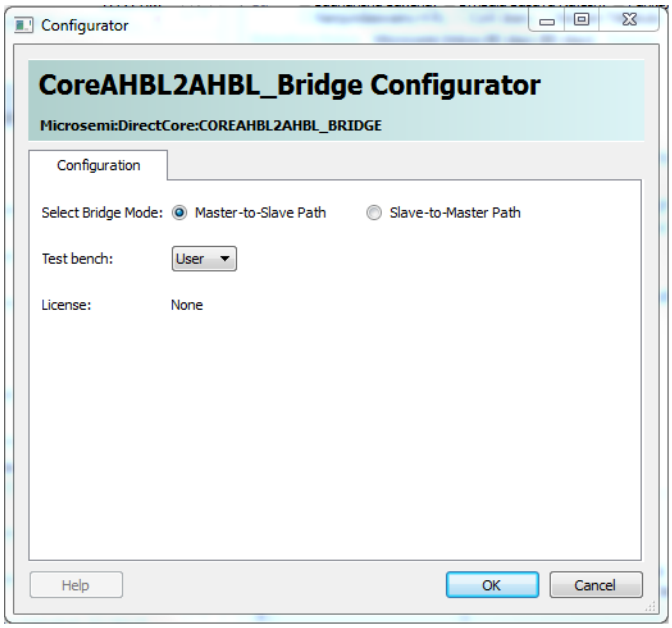
Figure 12 • **SmartDesign CoreAHBL2AHBL_Bridge Instance View**



6.4 Configuring CoreAHBL2AHBL_Bridge in SmartDesign

The following figure shows the CoreAHBL2AHBL_Bridge Configurator in SmartDesign.

Figure 13 • **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.

7 Test Bench

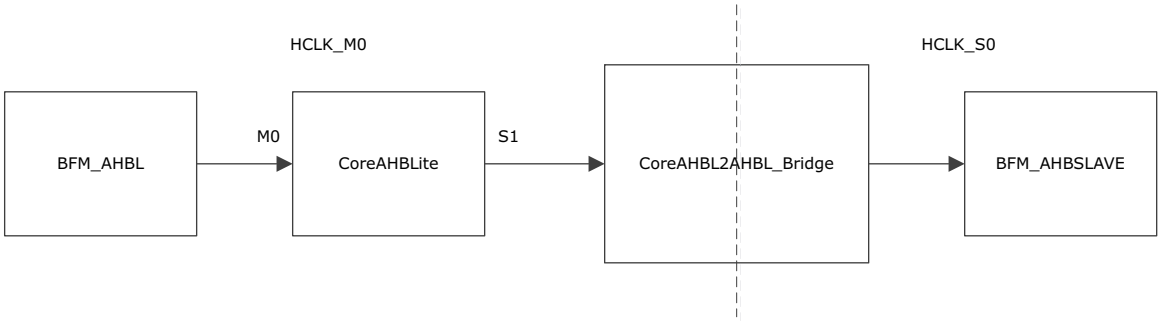
A unified test bench, referred to as the user test bench, is used to verify and test CoreAHBL2AHBL_Bridge.

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.

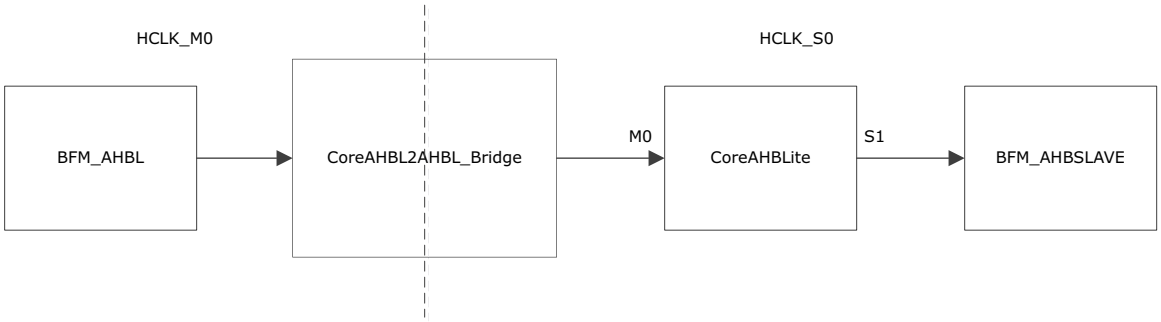
Figure 14 • **CoreAHBL2AHBL_Bridge User Test Bench Use Model 1**



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 15 • **CoreAHBL2AHBL_Bridge User Test Bench Use Model 2**



7.3 Use Case List

Table 4 • **Use Case List**

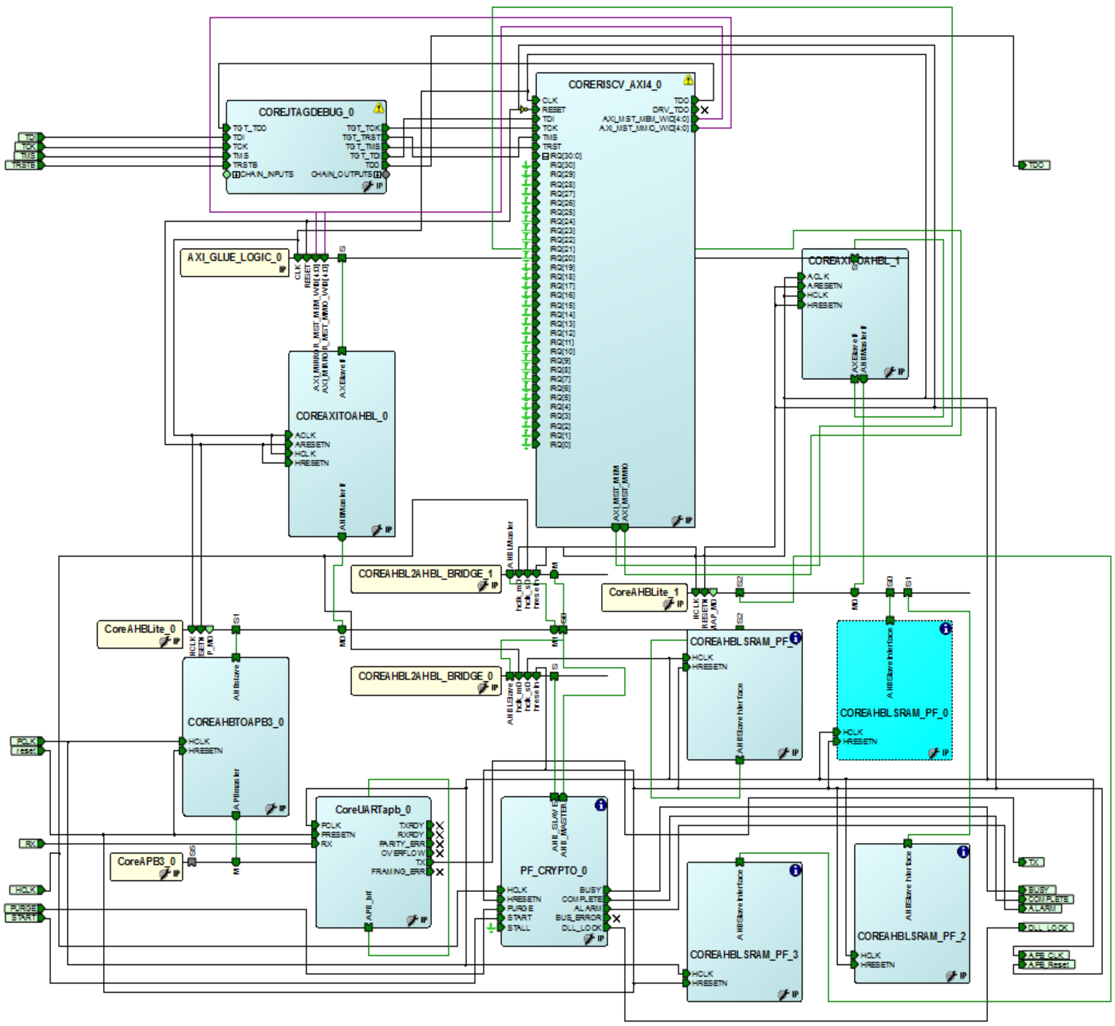
	Normal		Extended		Error Response	
	Read	**Write**	**Read**	**Write**	**Read**	**Write**
Single	single_rd	single_wr	single_ext_rd	single_ext_wr	single_rd_err	single_wr_err
INCR Burst	incr_rd	incr_wr	incr_ext_rd	incr_ext_wr	incr_rd_err	incr_wr_err
INCR4 Burst	incr4_rd	incr4_wr	incr4_ext_rd	incr4_ext_wr	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

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 16 • **CoreAHBL2AHBL_Bridge System Integration Diagram**



Note: The example design is tested on a Microsemi platform and can be obtained from the Microsemi Technical Support team.

8.1 Constraints

The following constraints are provided for CoreAHBL2AHBL_Bridge v2.0:

- 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_clock_group -asynchronous -group [get_clocks {HCLK_M0}] -group [get_clocks {HCLK_S0}]
  - set_clock_group -asynchronous -group [get_clocks {HCLK_S0}] -group [get_clocks {HCLK_M0}]