HB0940
Handbook
CoreMRAM_AHB v2.0

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

Microsemi makes no warranty, representation, or guarantee regarding the information contained herein or the suitability of its products and services for any particular purpose, nor does Microsemi assume any liability whatsoever arising out of the application or use of any product or circuit. The products sold hereunder and any other products sold by Microsemi have been subject to limited testing and should not be used in conjunction with mission-critical equipment or applications. Any performance specifications are believed to be reliable but are not verified, and Buyer must conduct and complete all performance and other testing of the products, alone and together with, or installed in, any end-products. Buyer shall not rely on any data and performance specifications or parameters provided by Microsemi. It is the Buyer’s responsibility to independently determine suitability of any products and to test and verify the same. The information provided by Microsemi hereunder is provided “as is, where is” and with all faults, and the entire risk associated with such information is entirely with the Buyer. Microsemi does not grant, explicitly or implicitly, to any party any patent rights, licenses, or any other IP rights, whether with regard to such information itself or anything described by such information. Information provided in this document is proprietary to Microsemi, and Microsemi reserves the right to make any changes to the information in this document or to any products and services at any time without notice.

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

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

The first publication of this document.

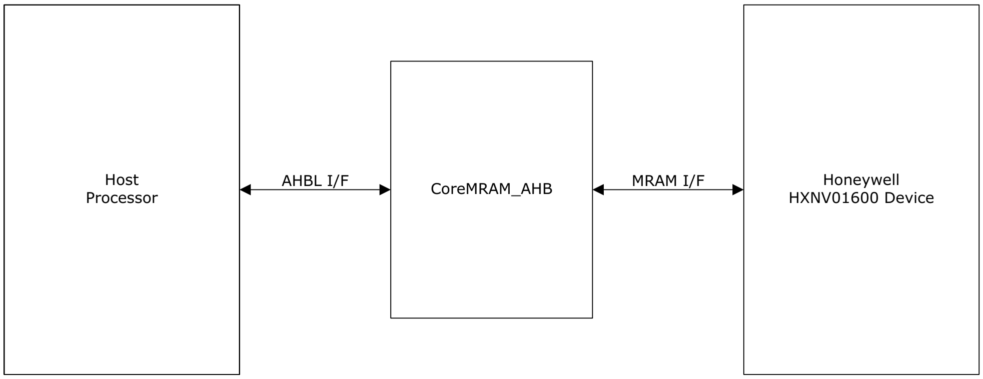
2 Introduction

2.1 Core Overview

CoreMRAM_AHB provides a high-performance interface to single-data-rate (SDR) synchronous Non-Volatile Magneto-Resistive RAM (MRAM) Honeywell HXNV01600 device. CoreMRAM_AHB accepts read and write commands using the advanced high-performance bus (AHBL) slave interface and translates these requests to the command sequences required by the MRAM device.

A typical application using CoreMRAM_AHB is shown in Figure 1.

Figure 1 • **CoreMRAM_AHB Application**



2.2 Core Version

This handbook applies to CoreMRAM_AHB v2.0. The release notes provided with the core, list the known discrepancies between this handbook and the core release associated with the release notes.

2.3 Supported Device Families

- PolarFire® SoC
- PolarFire®
- RTG4TM
- IGLOO®2
- SmartFusion®2

2.4 Key Features

The following key features are supported in the CoreMRAM_AHB module:

- Provides high performance, Non-Volatile Magneto-Resistive RAM (MRAM) controller for standard Honeywell HXNV01600 Device.
- Provides synchronous interface, fully pipelined internal architecture
- Supports up to 16 Megabit of memory
- Configuration supports 16Mb (x8 or x16)
- Supports Core Clock frequencies 12, 24, and 48 MHz
- Supports the ECC feature on internal RAM
- Supports for Advanced High-performance Lite **B**us (AHBL) slave interface

2.5 Device Utilization and Performance

CoreMRAM_AHB has been implemented in several Microsemi$^{®}$ device families. A summary of the implementation data is listed in Table 1.

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

FPGA Family and Device	Parameter Name	Utilization				Frequency (MHz)	
	BYTE_MODE_EN	Sequential	Comb	Total	%	HCLK	CORE_CLK
PolarFire SoC MPFS250T_ES-FCG1152	0	495	738	1230	0.24	200	175
	1	503	731	1234	0.24	199	175
PolarFire MPF500T-1FCG1152E	0	495	714	1209	0.125	219	194
	1	503	724	1227	0.125	233	205
RTG4 RT4G150-1FCG1657M	0	567	828	1395	0.46	157	150
	1	575	845	1420	0.47	145	142
IGLOO2 M2GL050T-FG484	0	567	839	1406	1.25	161	137
	1	575	795	1370	1.21	156	132
SmartFusion2 M2S050T- FG484	0	567	839	1406	1.25	161	137
	1	575	795	1370	1.21	156	132

Note: All data was obtained using a default system configuration. All performance data was obtained under commercial (COM) conditions.

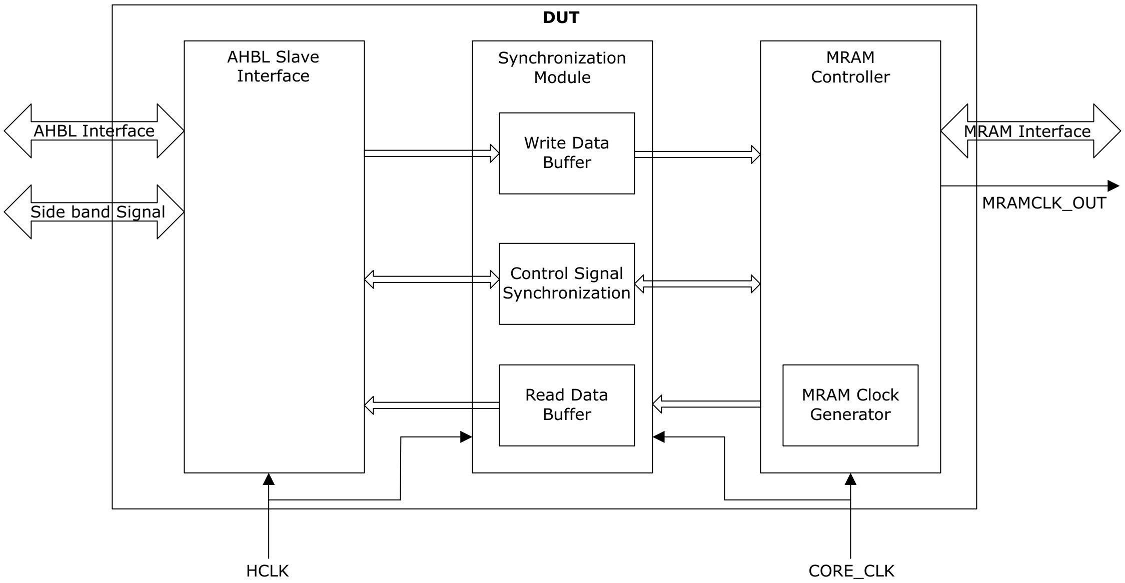
3 Functional Block Description

3.1 Functional Overview

CoreMRAM_AHB consists of the following primary blocks, as shown in Figure 2:

- AHBL Slave Interface - Perform Read and write transactions on the AHBL.
- Synchronization Module - Consist of write data buffer, read data buffer, and control signal synchronization module, which handles asynchronous data transfer between AHB HCLK and CORE_CLK clock domain and vice versa to support AHB Burst transactions.
- MRAM Controller: This block generates the required MRAM interface data, control, and clock signal.
- This core has three clock domains. That is AHB Clock, Core clock, and MRAM clock domain.

Figure 2 • **CoreMRAM_AHB Block Diagram**



3.2 Clocking and Reset

The Memory interface clock (MRAMCLK_OUT) is derived from CORE_CLK inside the IP. The MRAMCLK_OUT minimum required time-period for read and write transactions are 120ns and 140ns respectively. Table 2 shows the Read Cycle Time and Write Cycle Time corresponding to CORE_CLK frequency.

The CORE_CLK can be asynchronous to HCLK. The read/write buffers in the IP core handles the clock domain crossing of signals between HCLK to CORE_CLK and vice-versa.

Table 2 • **Read Cycle AC Timing Requirements**

CORE_CLK (MHz)	Read Cycle Time (ns)	Write Cycle Time (ns)
12	251	166
24	166	166
48	125	145

3.3 Timing Requirements of MRAM Device

3.3.1 Read Cycle

The non-volatile MRAM is synchronous in operation relative to the rising edge of the CLK signal. With the initiation of a rising CLK signal, the Address and the Write Enable (WE) signals are captured, and the read operation begins from the desired memory location. The addressed memory locations are read and compared with the ECC values. Any single bit errors are detected and corrected. If WE is low when captured, the data word is sent to the output drivers. In addition to WE low being captured, Output Enable (OE) must be set to high to enable the DQ output buffers. OE is not captured and may be set high before or after the rising edge of CLK. Figure 3 and Table 3 show the timing requirement for control and address signals.

Figure 3 • **Read Cycle AC Timing**

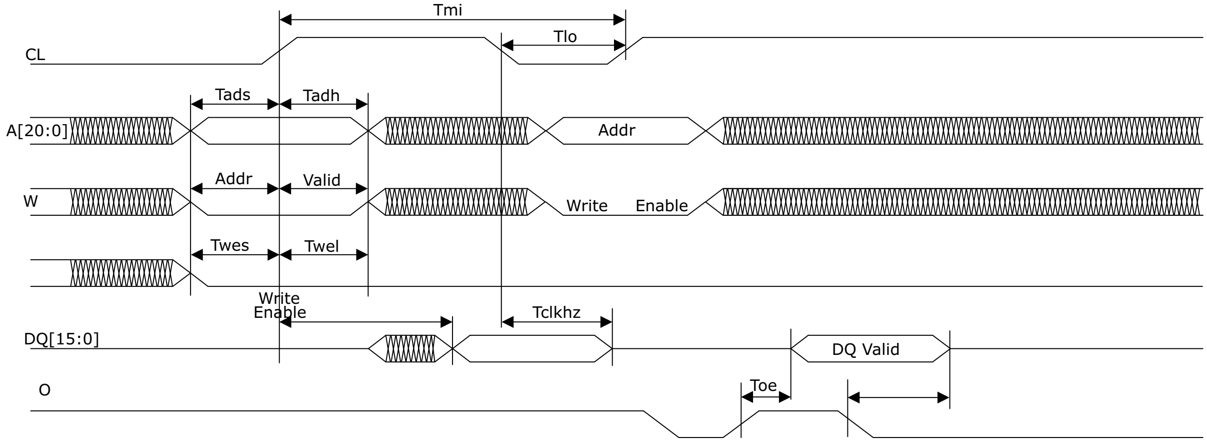


Table 3 • **Read Cycle AC Timing Requirements**

Name	Description	Min	Max	Units
Tads	Address Setup Time	5	-	ns
Tadh	Address Hold Time	15	-	ns
Twes	WE Setup Time	5	-	ns
Tweh	WE Hold Time	15	-	ns
Tcebs	CE B Setup Time	5	-	ns
Tcebh	CE B Hold Time	15	-	ns
Tclkdv	DQ valid with respect to rising edge of CLK	50	95	ns
Tclkhz	Clock Low to DQ Hi-z	1	15	ns
Toedv	OE access time	-	15	ns
Toehz	OE de-asserted to DQ Hi-z	1	15	ns
Tminr	Read Cycle Time	120	-	ns
Tlo	Clock Low Time	15	-	ns

3.3.2 Write Cycle

The MRAM is synchronous in operation relative to the rising edge of the CLK signal. With the initiation of a rising edge CLK signal, the Address, and WE signals are captured into the device. The WRITE CYCLE begins by reading the currently addressed value in memory. The current memory data are compared to the data to be written. If the location needs to change the value, the data are then written. The bit cell construction of this device does not provide a method of simply writing a “1” or a “0” to match the data. The “write” to a bit can only change its state, thus the need to read the bit location first. Only the bits which need to “change state” are written. Figure 4 and Table 4 show the timing requirement for control and address signals with respect to CLK.

Figure 4 • **Write Cycle AC Timing**

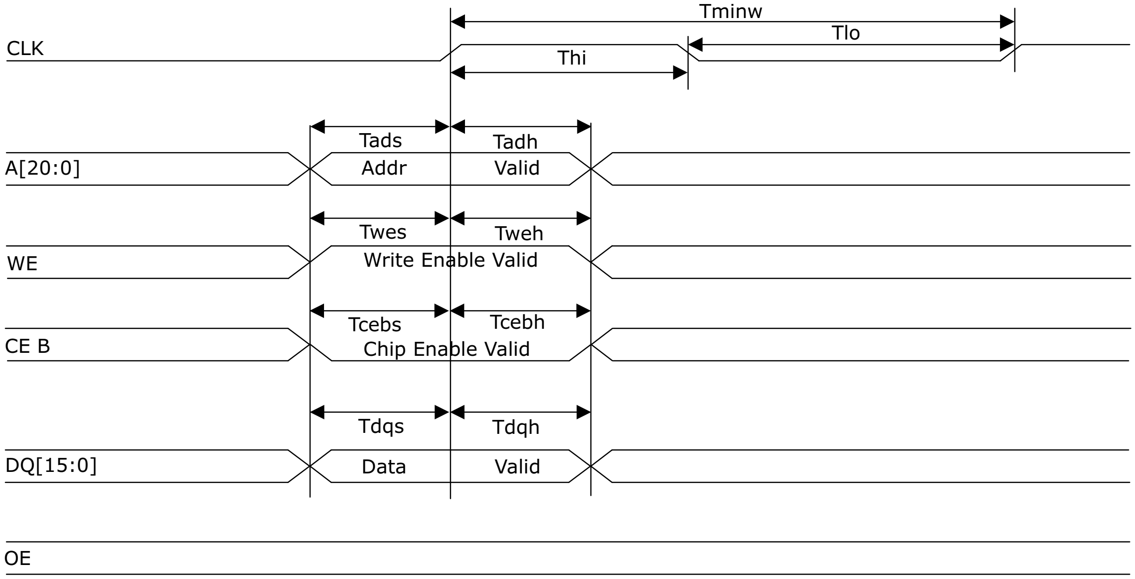


Table 4 • **CoreMRAM Write Cycle AC Timing Requirements**

Name	Description	Min	Max	Units
Tads	Address Setup Time	5	-	ns
Tadh	Address Hold Time	15	-	ns
Twes	WE Setup Time	5	-	ns
Tweh	WE Hold Time	15	-	ns
Tcebs	CE B Setup Time	5	-	ns
Tcebh	CE B Hold Time	15	-	ns
Tdqs	Data Setup Time	5	-	ns
Tdqh	Data Hold Time	15	-	ns
Tminw	Write Cycle Time	140	-	ns
Thi	Clock High Time	15	-	ns
Tlo	Clock Low Time	15	-	ns

4 Tool Flows

4.1 Licenses

No license is required for this core.

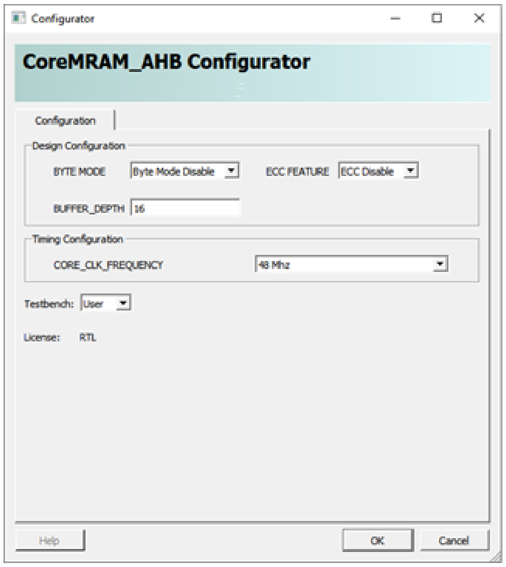
4.2 RTL

Complete RTL source code is provided for the core and testbenches.

4.3 SmartDesign

CoreMRAM_AHB 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 on using SmartDesign to configure, connect, and generate cores, refer to the *Using DirectCore in Libero User Guide* or consult the Libero SoC online help. The core can be configured using the configuration GUI within SmartDesign, as shown in Figure 5.

Figure 5 • **CoreMRAM_AHB Configuration Within SmartDesign**



Configurator

CoreMRAM_AHB Configurator

Configuration

Design Configuration

BYTE MODE: Byte Mode Disable

ECC FEATURE: ECC Disable

BUFFER_DEPTH: 16

Timing Configuration

CORE_CLK_FREQUENCY: 48 Mhz

Testbench: User

License: RTL

4.4 Simulation Flows

To run simulations, select the user testbench within the SmartDesign CoreMRAM_AHB configuration GUI, right-click, and select **Generate Design** (Figure 5).

When SmartDesign generates the design files, it will install the appropriate testbench files. To run the simulation, set the design root to the CoreMRAM_AHB instantiation in the Libero design hierarchy pane, and click **Simulation** in the Libero **Design Flow** window. This invokes ModelSim® and automatically runs the simulation.

4.5 Synthesis in Libero

Set the design root appropriately and click the Synthesis icon in the Libero. The synthesis window appears, displaying the Synplicity® project. To perform synthesis, click **Run**.

4.6 Place-and-Route in Libero

After the design has been synthesized, click **Layout** in Libero to invoke the designer. CoreMRAM_AHB requires no special place-and-route settings.

5 Core Parameters

5.1 Parameters/Generics

The generics are listed in Table 5 as required in the source code.

Table 5 • **CoreMRAM_AHB Generics**

Generic	Default Setting	Valid Values	Description
FAMILY	25	19, 24, 25, 26, 27	19 - SmartFusion2 24 - IGLOO2 25 - RTG4 26 - PolarFire 27 - PolarFire SoC
BYTE_MODE_EN	0	0-1	Select 16 bits / 8bits Memory interface 0 - enable x16 memory interface 1 - enable x8 memory interface
CORE_CLK_ FREQUENCY	12	12,24,48	CORE_CLK frequency in MHz
ECC	0	0-1	0 - ECC Disabled 1 - ECC Enabled **Note:** ECC is only available for the RTG4, PolarFire, and PolarFire SoC device families.
BUFFER_DEPTH	16	16-1024	Configurable FIFO depth, can be configure power of 2 values within 16 to 1024 range.

6 Interface Descriptions

The port signals for CoreMRAM_AHB are defined in Table 6 and Table 7. The port signals are also shown in Figure 6, page 11 and Figure 7, page 11. All signals are designated either input (input-only) or output (output-only), except DQ, which is bidirectional.

6.1 AHB Interface Signals

The user interface to CoreMRAM_AHB is referred to as the local bus interface. The local bus signals are listed in Table 6.

Table 6 • **Local Bus Signals**

Signal	I/O	Description
CORE_CLK	Input	MRAM Controller Clock
CORECLK_RESETN	Input	Asynchronous reset (Active low and asynchronous)
HCLK	Input	AHB clock
HRESETN	Input	AHB reset (Active low and asynchronous)
HADDR[31:0]	Input	AHB address
HREADYIN	Input	AHB ready in
HTRANS[1:0]	Input	AHB transfer type
HWRITE	Input	AHB write/read
HSIZE[2:0]	Input	AHB transfer size
HBURST[2:0]	Input	AHB burst type
HSEL	Input	AHB slave select
HREADY	Output	AHB ready out
HRESP[1:0]	Output	AHB response
HWDATA[31:0]	Input	AHB data in
HRDATA[31:0]	Output	AHB data out
ECC_ERROR_SB	Output	Single bit error correct ECC status signal, when set to logic 1, indicates that the data buffer/FIFO has a single bit error corrected by ECC logic. Visible at the port when parameter ECC = 1.
ECC_ERROR_DB	Output	Double bit error detects ECC status signal, when set to logic 1, indicates that the data buffer/FIFO has double bit error detected by ECC logic. Visible at the port when parameter ECC = 1.

6.2 MRAM Interface Signals

The external interface to the Non-Volatile Magneto-Resistive RAM device is referred to as the MRAM interface. The MRAM interface signals are listed in Table 7.

Table 7 • **MRAM Interface Signals**

Signal	I/O	Description
MRAMCLK_OUT	Output	MRAM interface clock
A[20:0]	Output	Address Bus
WE	Output	Write Enable
X8	Output	Byte Mode configuration
OE	Output	Output Enable, tristate control for DQ data
DQ [15:0]	Input/Output	MRAM in/out data When OE=1, it drives read data, otherwise it drives write data.
AUTO_INCR	Output	AUTO_INCR pin drive logic 0, as Auto Increment Feature is not supported.
OVERFLOW_I	Output	OVERFLOW_I pin drive logic 0, as Auto Increment Feature is not supported.
INIT	Output	INIT pin drive logic 0, as Auto Increment Feature is not supported.
DONE	Output	DONE pin drive logic 0, as Auto Increment Feature is not supported.
OVERFLOW_O	Input	Internal Overflow Counter Indicator, as Auto Increment Feature is not supported this signal does not drive logic inside IP core.

7 Interface Timings

7.1 MRAM Writes and Reads

Figure 6 shows an example CoreMRAM_AHB write transaction of HSIZE = 2 and 1, BYTE_MODE_EN = 0.

Figure 6 • CoreMRAM_AHB write transaction of HSIZE = 2 and 1, BYTE_MODE_EN = 0

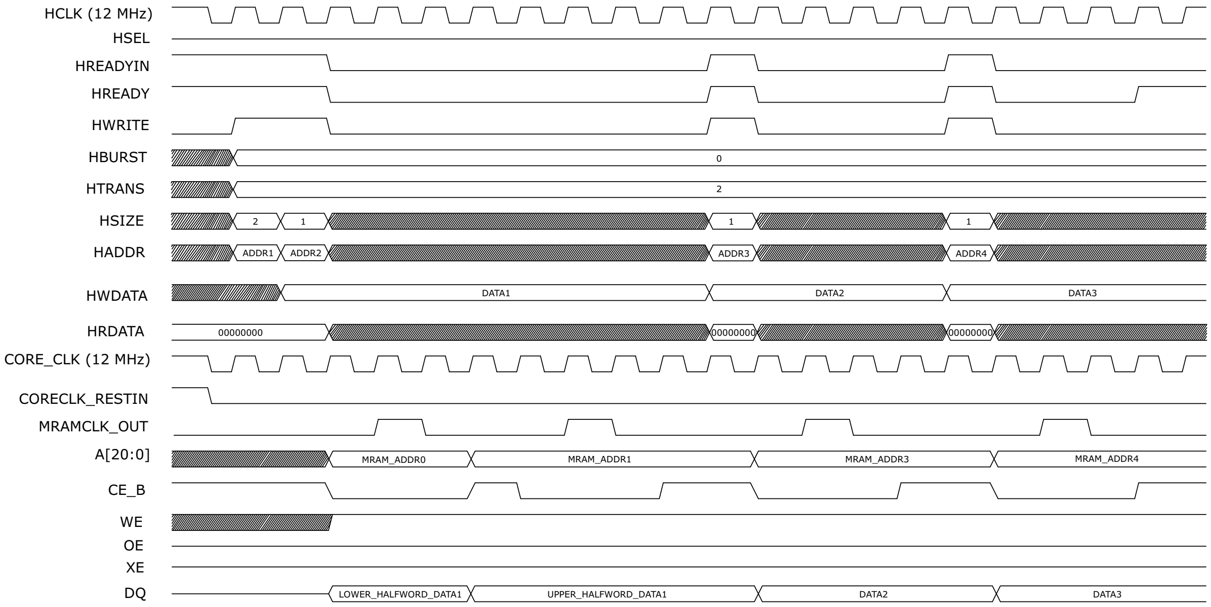
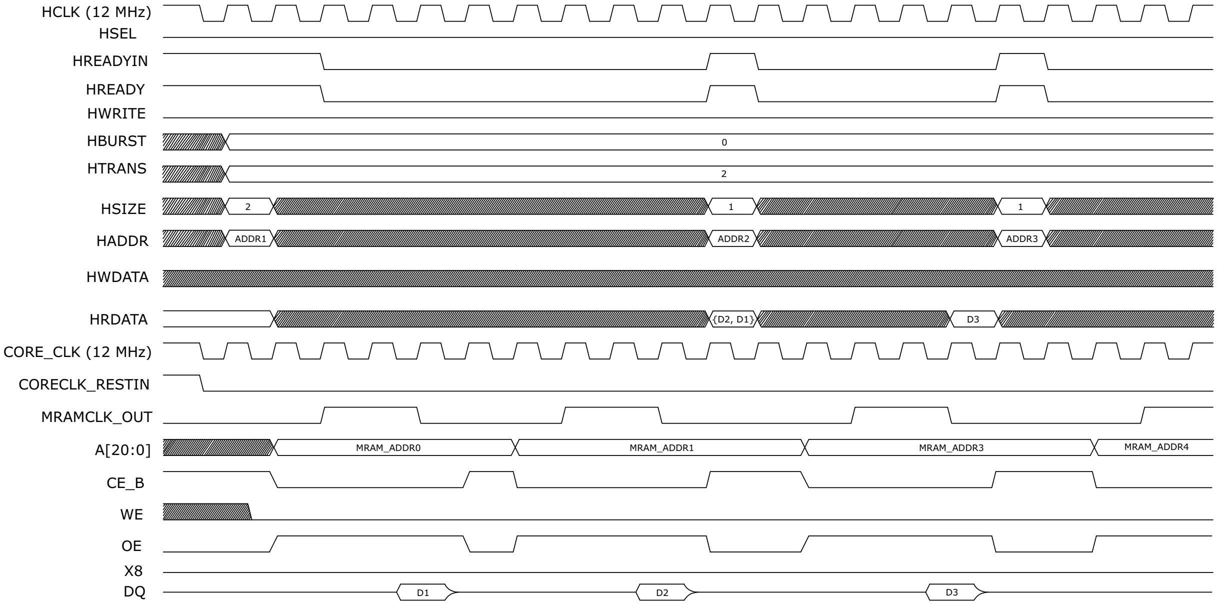


Figure 7 shows an example CoreMRAM_AHB Read transaction of HSIZE = 2 and 1, BYTE_MODE_EN = 0.

Figure 7 • CoreMRAM_AHB Read transaction of HSIZE = 2 and 1, BYTE_MODE_EN = 0



8 Design Constraints

This section describes the timing constraints of the CoreMRAM_AHB IP core.

8.1 Timing Constraints

Asynchronous FIFO and Pulse Synchronizer used in the core to transfer data between asynchronous clock domains and to synchronize control signal in all the clock domains respectively, requires timing constraint for synthesis, place and route, and timing verification. Add the constraints for the path from HCLK clock domain to CORE_CLK clock domain and from CORE_CLK clock domain to HCLK domain is as follows:

CoreMRAM_AHB_0--- Instance name of CoreMRAM_AHB module.

- When the parameter BYTE_MODE_EN is set to 0

```
set_false_path -from [get_cells
{CoreMRAM_AHB_0/U_COREMRAM_AHBLIF/command_burst*} ] -to [get_cells
{CoreMRAM_AHB_0/U_COREMRAM_AHBLIF/rx_fifo_rd_cnt*} ]
set_false_path -from [get_cells
{CoreMRAM_AHB_0/U_COREMRAM_AHBLIF/command_burst*} ] -to [get_cells
{CoreMRAM_AHB_0/U_COREMRAM_AHBLIF/rx_fifo_rd_done*} ]
set_false_path -from [get_cells
{CoreMRAM_AHB_0/U_COREMRAM_AHBLIF/number_of_mram_trans*} ] -to [get_cells
{CoreMRAM_AHB_0/U_COREMRAM_AHBLIF/number_of_mram_trans_ahb_s*} ]
set_false_path -from [get_cells
{CoreMRAM_AHB_0/U_COREMRAM_AHBLIF/trans_type_ahb*} ] -to [get_cells
{CoreMRAM_AHB_0/U_COREMRAM_AHBLIF/trans_type_core_s*} ]
set_false_path -from [get_cells
{CoreMRAM_AHB_0/U_COREMRAM_AHBLIF/wr_follow_rd*} ] -to [get_cells
{CoreMRAM_AHB_0/U_COREMRAM_AHBLIF/wr_follow_rd_core_s*}]
set_false_path -from [get_cells
{CoreMRAM_AHB_0/U_COREMRAM_AHBLIF/command*} ] -to [get_cells
{CoreMRAM_AHB_0/U_COREMRAM_AHBLIF/command_addr*} ]
set_false_path -from [get_cells
{CoreMRAM_AHB_0/U_COREMRAM_AHBLIF/command*} ] -to [get_cells
{CoreMRAM_AHB_0/U_COREMRAM_AHBLIF/command_size*}]
set_false_path -from [get_cells
{CoreMRAM_AHB_0/U_COREMRAM_AHBLIF/command*} ] -to [get_cells
{CoreMRAM_AHB_0/U_COREMRAM_AHBLIF/command_burst*}]
set_false_path -from [get_cells
{CoreMRAM_AHB_0/U_COREMRAM_AHBLIF/number_of_mram_trans_cnt*} ] -to
[get_cells {CoreMRAM_AHB_0/U_COREMRAM_MRAMIF/DQ_out*}]
set_false_path -from [get_cells
{CoreMRAM_AHB_0/U_COREMRAM_AHBLIF/number_of_mram_trans_cnt*} ] -to
[get_cells {CoreMRAM_AHB_0/U_COREMRAM_MRAMIF/A_int*}]
set_false_path -from [get_cells
{CoreMRAM_AHB_0/U_COREMRAM_AHBLIF/CORESYNC_TERMINATE/pulse_gen_i/toggle_o
ut*} ] -to [get_cells
{CoreMRAM_AHB_0/U_COREMRAM_AHBLIF/CORESYNC_TERMINATE/pulse_cdc_sync_i/syn
c_ff*}]
```

```
set_false_path -from [get_cells
{CoreMRAM_AHB_0/U_COREMRAM_AHBLIF/CORESYNC_NEW_TRANS_MRAM/pulse_gen_i/tog
gle_out*} ] -to [get_cells
{CoreMRAM_AHB_0/U_COREMRAM_AHBLIF/CORESYNC_NEW_TRANS_MRAM/pulse_cdc_sync_
i/sync_ff*}]
set_false_path -from [get_cells
{CoreMRAM_AHB_0/U_COREMRAM_AHBLIF/CORESYNC_TX_FIFO_RD_EN/pulse_gen_i/togg
le_out*} ] -to [get_cells
{CoreMRAM_AHB_0/U_COREMRAM_AHBLIF/CORESYNC_TX_FIFO_RD_EN/pulse_cdc_sync_i
/sync_ff*}]
set_false_path -from [get_cells
{CoreMRAM_AHB_0/U_COREMRAM_AHBLIF/CORESYNC_RX_ADDR_LOAD_EN/pulse_gen_i/to
ggle_out*} ] -to [get_cells
{CoreMRAM_AHB_0/U_COREMRAM_AHBLIF/CORESYNC_RX_ADDR_LOAD_EN/pulse_cdc_sync
_i/sync_ff*}]
set_false_path -from [get_cells
{CoreMRAM_AHB_0/U_COREMRAM_AHBLIF/CORESYNC_SHIFT_MRAM_ADDR_EN/pulse_gen_i
/toggle_out*} ] -to [get_cells
{CoreMRAM_AHB_0/U_COREMRAM_AHBLIF/CORESYNC_SHIFT_MRAM_ADDR_EN/pulse_cdc_s
ync_i/sync_ff*}]
set_false_path -from [get_cells
{CoreMRAM_AHB_0/U_COREMRAM_AHBLIF/CORESYNC_TRANSACTION_DONE/pulse_gen_i/t
oggle_out*} ] -to [get_cells
{CoreMRAM_AHB_0/U_COREMRAM_AHBLIF/CORESYNC_TRANSACTION_DONE/pulse_cdc_syn
c_i/sync_ff*}]
set_false_path -from [get_cells
{CoreMRAM_AHB_0/U_COREMRAM_AHBLIF/coresync_load_address/pulse_gen_i/toggl
e_out*} ] -to [get_cells
{CoreMRAM_AHB_0/U_COREMRAM_AHBLIF/coresync_load_address/pulse_cdc_sync_i/
sync_ff*}]
set_false_path -from [get_cells
{CoreMRAM_AHB_0/U_COREMRAM_AHBLIF/TX_DATA_FIFO/wrGrayCounter/cntGray*} ]
-to [get_cells {CoreMRAM_AHB_0/U_COREMRAM_AHBLIF/TX_DATA_FIFO/wrPtr_s1*}]
set_false_path -from [get_cells
{CoreMRAM_AHB_0/U_COREMRAM_AHBLIF/RX_DATA_FIFO/wrGrayCounter/cntGray*} ]
-to [get_cells {CoreMRAM_AHB_0/U_COREMRAM_AHBLIF/RX_DATA_FIFO/wrPtr_s1*}]
```

- When the parameter BYTE_MODE_EN is set to 1:

```
set_false_path -from [get_cells
{CoreMRAM_AHB_0/U_COREMRAM_AHBLIF/command_burst*} ] -to [get_cells
{CoreMRAM_AHB_0/U_COREMRAM_AHBLIF/rx_fifo_rd_cnt*} ]
set_false_path -from [get_cells
{CoreMRAM_AHB_0/U_COREMRAM_AHBLIF/command_burst*} ] -to [get_cells
{CoreMRAM_AHB_0/U_COREMRAM_AHBLIF/rx_fifo_rd_done*} ]
set_false_path -from [get_cells
{CoreMRAM_AHB_0/U_COREMRAM_AHBLIF/number_of_mram_trans*} ] -to [get_cells
{CoreMRAM_AHB_0/U_COREMRAM_AHBLIF/number_of_mram_trans_ahb_s*} ]
set_false_path -from [get_cells
{CoreMRAM_AHB_0/U_COREMRAM_AHBLIF/trans_type_ahb*} ] -to [get_cells
{CoreMRAM_AHB_0/U_COREMRAM_AHBLIF/trans_type_core_s*} ]
```

```
set_false_path -from [get_cells
{CoreMRAM_AHB_0/U_COREMRAM_AHBLIF/command*} ] -to [get_cells
{CoreMRAM_AHB_0/U_COREMRAM_AHBLIF/command_addr*} ]
set_false_path -from [get_cells
{CoreMRAM_AHB_0/U_COREMRAM_AHBLIF/command*} ] -to [get_cells
{CoreMRAM_AHB_0/U_COREMRAM_AHBLIF/command_size*}]
set_false_path -from [get_cells
{CoreMRAM_AHB_0/U_COREMRAM_AHBLIF/command*} ] -to [get_cells
{CoreMRAM_AHB_0/U_COREMRAM_AHBLIF/command_burst*}]
set_false_path -from [get_cells
{CoreMRAM_AHB_0/U_COREMRAM_AHBLIF/number_of_mram_trans_cnt*} ] -to
[get_cells {CoreMRAM_AHB_0/U_COREMRAM_MRAMIF/DQ_out*}]
set_false_path -from [get_cells
{CoreMRAM_AHB_0/U_COREMRAM_AHBLIF/number_of_mram_trans_cnt*} ] -to
[get_cells {CoreMRAM_AHB_0/U_COREMRAM_MRAMIF/A_int*}]
set_false_path -from [get_cells
{CoreMRAM_AHB_0/U_COREMRAM_AHBLIF/CORESYNC_TERMINATE/pulse_gen_i/toggle_o
ut*} ] -to [get_cells
{CoreMRAM_AHB_0/U_COREMRAM_AHBLIF/CORESYNC_TERMINATE/pulse_cdc_sync_i/syn
c_ff*}]
set_false_path -from [get_cells
{CoreMRAM_AHB_0/U_COREMRAM_AHBLIF/CORESYNC_NEW_TRANS_MRAM/pulse_gen_i/tog
gle_out*} ] -to [get_cells
{CoreMRAM_AHB_0/U_COREMRAM_AHBLIF/CORESYNC_NEW_TRANS_MRAM/pulse_cdc_sync_
i/sync_ff*}]
set_false_path -from [get_cells
{CoreMRAM_AHB_0/U_COREMRAM_AHBLIF/CORESYNC_TX_FIFO_RD_EN/pulse_gen_i/togg
le_out*} ] -to [get_cells
{CoreMRAM_AHB_0/U_COREMRAM_AHBLIF/CORESYNC_TX_FIFO_RD_EN/pulse_cdc_sync_i
/sync_ff*}]
set_false_path -from [get_cells
{CoreMRAM_AHB_0/U_COREMRAM_AHBLIF/CORESYNC_RX_ADDR_LOAD_EN/pulse_gen_i/to
ggle_out*} ] -to [get_cells
{CoreMRAM_AHB_0/U_COREMRAM_AHBLIF/CORESYNC_RX_ADDR_LOAD_EN/pulse_cdc_sync
_i/sync_ff*}]
set_false_path -from [get_cells
{CoreMRAM_AHB_0/U_COREMRAM_AHBLIF/CORESYNC_SHIFT_MRAM_ADDR_EN/pulse_gen_i
/toggle_out*} ] -to [get_cells
{CoreMRAM_AHB_0/U_COREMRAM_AHBLIF/CORESYNC_SHIFT_MRAM_ADDR_EN/pulse_cdc_s
ync_i/sync_ff*}]
set_false_path -from [get_cells
{CoreMRAM_AHB_0/U_COREMRAM_AHBLIF/CORESYNC_TRANSACTION_DONE/pulse_gen_i/t
oggle_out*} ] -to [get_cells
{CoreMRAM_AHB_0/U_COREMRAM_AHBLIF/CORESYNC_TRANSACTION_DONE/pulse_cdc_syn
c_i/sync_ff*}]
set_false_path -from [get_cells
{CoreMRAM_AHB_0/U_COREMRAM_AHBLIF/coresync_load_address/pulse_gen_i/toggl
e_out*} ] -to [get_cells
{CoreMRAM_AHB_0/U_COREMRAM_AHBLIF/coresync_load_address/pulse_cdc_sync_i/
sync_ff*}]
```

```
set_false_path -from [get_cells
{CoreMRAM_AHB_0/U_COREMRAM_AHBLIF/TX_DATA_FIFO/wrGrayCounter/cntGray*} ]
-to [get_cells {CoreMRAM_AHB_0/U_COREMRAM_AHBLIF/TX_DATA_FIFO/wrPtr_s1*}]
set_false_path -from [get_cells
{CoreMRAM_AHB_0/U_COREMRAM_AHBLIF/RX_DATA_FIFO/wrGrayCounter/cntGray*} ]
-to [get_cells {CoreMRAM_AHB_0/U_COREMRAM_AHBLIF/RX_DATA_FIFO/wrPtr_s1*}]
```

9 Testbench Operation and Modifications

9.1 Testbench Operation

The following testbench is provided with CoreMRAM_AHB:

VHDL testbench with Verilog Memory Model.

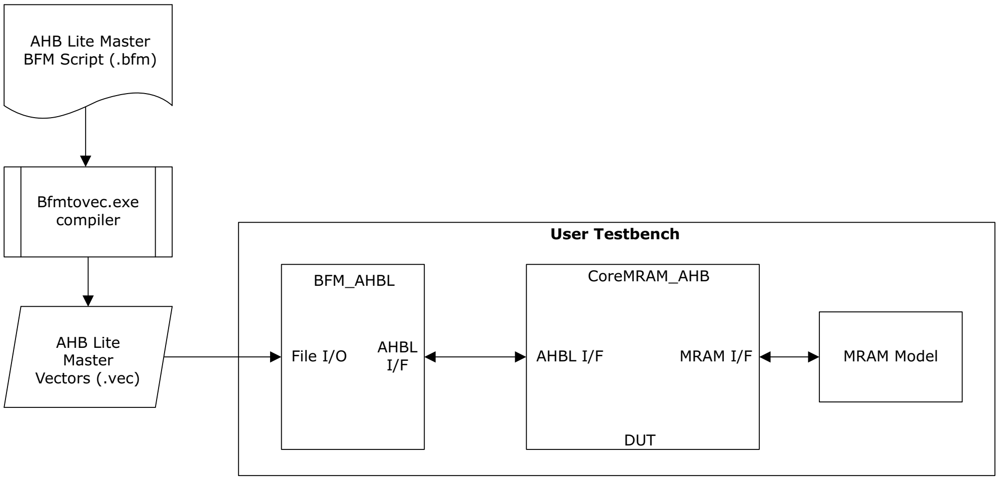
9.1.1 VHDL User Testbench

The VHDL user testbench is provided as a reference and can be modified to suit the requirements. The source code for the VHDL testbench is provided to ease the process of integrating the CoreMRAM_AHB macro into the design and verifying its functionality.

9.2 Testbench Description

A user testbench is included with the RTL release of CoreMRAM_AHB. A simplified block diagram of the testbench is shown in Figure 8. By default, the VHDL version, `tb_user.vhd` instantiates a Honeywell HXNV01600 16 Mbits Model (Honeywell_HXNV01600_0190423.v). The testbench instantiates the design under test (DUT), which is the CoreMRAM_AHB, the MRAM model, as well as the test vector modules that provide stimuli sources for the DUT. A procedural testbench controls each module and applies the sequential stimuli to the DUT.

Figure 8 • **CoreMRAM_AHB Testbench**



A commented bus functional model (BFM) ASCII script source file (`.bfm`) is included in the following directory: YourLiberoProjectDirectory/ simulation, where the Libero Project Dir represents the path to the Libero SoC project where CoreMRAM_AHB is used. The BFM source file is for controlling the AHB-Lite master and is named master.bfm. The BFM source file is automatically recompiled each time the simulation is invoked from Libero SoC by bfmtovec.exe, if running on a Windows® platform, or by bfmtovec.lin, if running on a Linux platform. The output `.vec` file created by the bfmtovec executable is read in by the BFM modules for simulation in ModelSim.

The BFM scripts can be altered if desired. For more information, refer to the *DirectCore AMBA BFM User Guide*. The source code for the user testbench, BFM scripts, and compiled ModelSim simulation library containing the BFM modules are available with the CoreMRAM_AHB RTL release.