LMB BRAM Interface Controller v4.0

LogiCORE IP Product Guide

Vivado Design Suite

PG112 April 6, 2016

XILINX
ALL PROGRAMMABLE™

Table of Contents

Appendix B: Debugging

Appendix C: Application Software Development

Appendix D: Additional Resources and Legal Notices

Introduction

This document provides the design specification for the LogiCORE IP Local Memory Bus (LMB) Block RAM (BRAM) Interface Controller core. The LMB BRAM Interface Controller core connects to an lmb_v10 bus.

Version v4.0 of the LMB BRAM Interface Controller core requires MicroBlaze™ v9.0 or higher and lmb_v10 v3.0 or higher.

Features

- LMB v1.0 bus interfaces with byte enable support.
- Used in conjunction with the Block Memory Generator core to provide fast block RAM memory solution for MicroBlaze ILMB and DLMB ports.
- Supports byte, half-word, and word transfers.
- Supports optional BRAM error correction and detection.
- Supports multiple LMB masters.
- Support for extended address up to 64 bits.

LogiCORE IP Facts Table	
Core Specifics	
Supported Device Family(1)	UltraScale+™ Families UltraScale™ Architecture Zynq®-7000 All Programmable SoC 7 Series
Supported User Interfaces	LMB, AXI4-Lite
Resources	Performance and Resource Utilization web page
Provided with Core	
Design Files	Vivado: RTL
Example Design	Not Provided
Test Bench	Not Provided
Constraints File	Not Provided
Simulation Model	VHDL Behavioral
Supported S/W Driver(2)	Standalone
Tested Design Flows(3)	
Design Entry	Vivado® Design Suite
Simulation	For supported simulators, see the Xilinx Design Tools: Release Notes Guide.
Synthesis	Vivado Synthesis
Support	
Provided by Xilinx at the Xilinx Support web page	

Notes:

1. For a complete listing of supported devices, see the Vivado IP Catalog.
2. Standalone driver details can be found in the SDK directory (<*install_directory*>/doc/usenglish/xilinx_drivers.htm). Linux OS and driver support information is available from the Xilinx Wiki page.
3. For the supported versions of the tools, see the Xilinx Design Tools: Release Notes Guide.

Overview

The LMB BRAM Interface Controller core is the interface between the LMB and the Xilinx® Block Memory Generator core. A block RAM memory subsystem consists of the controller and the bram_block peripheral or Block Memory Generator core.

The input/output signals of the LMB BRAM Interface Controller core are shown in Figure 1-1. The detailed list of signals are listed and described in Table 2-1. See the description of LMB Signals in the MicroBlaze™ Bus Interfaces chapter in the *MicroBlaze Processor Reference Guide* (UG984) [Ref 1].

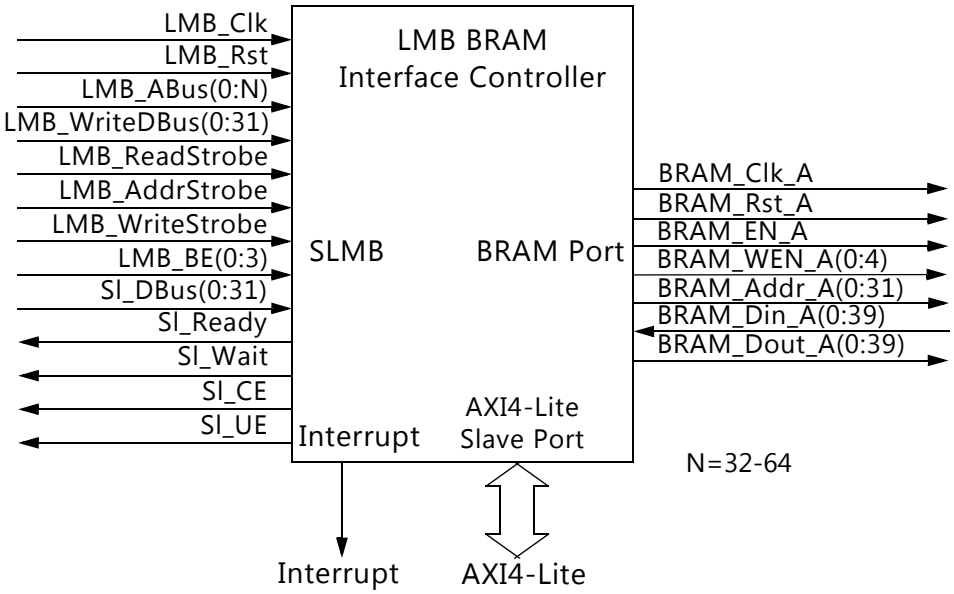

Figure 1-1: **LMB BRAM Interface Controller Core Block Diagram**

Feature Summary

Provides a low area, high frequency and low latency connection for the MicroBlaze DLMB and ILMB ports to device block RAM. The supported block RAM sizes are determined by the Block Memory Generator (nominally 4–256 KB), with the possibility of performing 32-bit word, 16-bit half word, as well as byte accesses.

Error Correction Codes (ECC) is available as an option for providing a solution suitable for applications with higher reliability requirements. When enabled, the ECC function corrects all single bit errors and detects all double bit errors. A set of optional ECC control and status registers are available, making it possible to tailer the ECC function to meet different requirements on ECC error injection, monitoring and signaling. The optional ECC registers are connected to MicroBlaze through an AXI4-Lite interface.

The LMB BRAM Interface Controller core supports multiple LMB masters, making it possible to use only one of the ports of the block RAM. This allows the other port of the block RAM to be used for low latency/low overhead data movement to and from MicroBlaze local memory.

When MicroBlaze is configured to use an extended data address from 32 to 64 bits, the data side LMB BRAM Interface Controller uses the extended address to determine if the local memory is accessed.

Licensing and Ordering Information

This Xilinx LogiCORE™ IP module is provided at no additional cost with the Xilinx Vivado® Design Suite under the terms of the Xilinx End User License. Information about this and other Xilinx LogiCORE IP modules is available at the Xilinx Intellectual Property page. For information about pricing and availability of other Xilinx LogiCORE IP modules and tools, contact your local Xilinx sales representative.

Product Specification

Standards

The LMB BRAM Interface Controller adheres to the *ARM® AMBA® AXI4-Lite Protocol Specification IHI 0022E* [Ref 2].

Performance

The frequency and latency of the LMB BRAM Interface Controller are optimized for use with MicroBlaze™. This means that the frequency targets are aligned to MicroBlaze targets as well as the 1 cycle latency optimized for MicroBlaze instruction and data access.

Maximum Frequencies

For details about performance, visit Performance and Resource Utilization.

Latency

Data read from block RAM is available the clock cycle after the address strobe is asserted when a single port is used. This is also true when single bit errors are corrected. Data write is performed the clock cycle after the address strobe is asserted, when a single port is used.

When ECC is enabled, byte and half word data writes add a two cycle latency to the write access. This is to perform a read-modify-write cycle to generate proper ECC for the full 32-bit word stored in block RAM. When multiple ports are used, latency is increased when an access has to wait until an ongoing access on another port with higher priority is completed.

Throughput

The nominal throughput is one read or write access every clock cycle. The only exceptions are performing a byte or half word write when ECC is enabled, and when an access is ongoing on another port when using multiple ports.

Resource Utilization

For details about resource utilization, visit Performance and Resource Utilization.

Port Descriptions

The I/O ports and signals for the LMB BRAM Interface Controller are listed and described in Table 2-1.

Table 2-1: **LMB BRAM Interface Controller I/O Signals**

Port Name	MSB:LSB	I/O	Description
LMB Signals			
LMB_Clk		I	LMB Clock
LMB_Rst		I	LMB Reset (Active-High)
LMB_ABus	0:C_LMB_AWIDTH-1	I	LMB Address Bus
LMB_WriteDBus	0:C_LMB_DWIDTH-1	I	LMB Write Data Bus
LMB_ReadStrobe		I	LMB Read Strobe
LMB_AddrStrobe		I	LMB Address Strobe
LMB_WriteStrobe		I	LMB Write Strobe
LMB_BE	0:C_LMB_DWIDTH/8-1	I	LMB Byte Enable Bus
Sl_DBus	0:C_LMB_DWIDTH-1	O	LMB Read Data Bus
Sl_Ready		O	LMB Data Ready
Sl_Wait		O	LMB Wait
Sl_CE		O	LMB Correctable Error
Sl_UE		O	LMB Uncorrectable Error
LMB1_ABus	0:C_LMB_AWIDTH-1	I	LMB1 Address Bus
LMB1_WriteDBus	0:C_LMB_DWIDTH-1	I	LMB1 Write Data Bus
LMB1_ReadStrobe		I	LMB1Read Strobe
LMB1_AddrStrobe		I	LMB1 Address Strobe
LMB1_WriteStrobe		I	LMB1 Write Strobe
LMB1_BE	0:C_LMB_DWIDTH/8-1	I	LMB1 Byte Enable Bus
Sl1_DBus	0:C_LMB_DWIDTH-1	O	LMB1 Read Data Bus
Sl1_Ready		O	LMB1 Data Ready
Sl1_Wait		O	LMB1 Wait
Sl1_CE		O	LMB1 Correctable Error
Sl1_UE		O	LMB1 Uncorrectable Error

Table 2-1: **LMB BRAM Interface Controller I/O Signals *(Cont'd)***

Port Name	MSB:LSB	I/O	Description
LMB2_ABus	0:C_LMB_AWIDTH-1	I	LMB2 Address Bus
LMB2_WriteDBus	0:C_LMB_DWIDTH-1	I	LMB2 Write Data Bus
LMB2_ReadStrobe		I	LMB2 Read Strobe
LMB2_AddrStrobe		I	LMB2 Address Strobe
LMB2_WriteStrobe		I	LMB2 Write Strobe
LMB2_BE	0:C_LMB_DWIDTH/8-1	I	LMB2 Byte Enable Bus
SI2_DBus	0:C_LMB_DWIDTH-1	O	LMB2 Read Data Bus
SI2_Ready		O	LMB2 Data Ready
SI2_Wait		O	LMB2 Wait
SI2_CE		O	LMB2 Correctable Error
SI2_UE		O	LMB2 Uncorrectable Error
LMB3_ABus	0:C_LMB_AWIDTH-1	I	LMB3 Address Bus
LMB3_WriteDBus	0:C_LMB_DWIDTH-1	I	LMB3 Write Data Bus
LMB3_ReadStrobe		I	LMB3 Read Strobe
LMB3_AddrStrobe		I	LMB3 Address Strobe
LMB3_WriteStrobe		I	LMB3 Write Strobe
LMB3_BE	0:C_LMB_DWIDTH/8-1	I	LMB3 Byte Enable Bus
SI3_DBus	0:C_LMB_DWIDTH-1	O	LMB3 Read Data Bus
SI3_Ready		O	LMB3 Data Ready
SI3_Wait		O	LMB3 Wait
SI3_CE		O	LMB3 Correctable Error
SI3_UE		O	LMB3 Uncorrectable Error
Block RAM Interface Signals (Data and ECC)			
BRAM_Rst_A		O	Block RAM Reset
BRAM_Clk_A		O	Block RAM Clock
BRAM_EN_A		O	Block RAM Enable
BRAM_WEN_A	0:(C_LMB_DWIDTH+8*C_ECC)/8-1	O	Block RAM Write Enable
BRAM_Addr_A	0:C_BRAM_AWIDTH-1	O	Block RAM Address
BRAM_Din_A	0:C_LMB_DWIDTH+8*C_ECC-1	I	Block RAM Data Input
BRAM_Dout_A	0:C_LMB_DWIDTH+8*C_ECC-1	O	Block RAM Data Output
Miscellaneous Signals			
Interrupt		O	Interrupt
UE		O	One cycle pulse signalling an ECC Uncorrectable Data Error

Table 2-1: **LMB BRAM Interface Controller I/O Signals *(Cont'd)***

Port Name	MSB:LSB	I/O	Description
CE		O	One cycle pulse signalling an ECC Correctable Data Error
AXI System Signals			
S_AXI_CTRL_ACLK		I	AXI Clock
S_AXI_CTRL_ARESETN		I	AXI Reset, active-Low
AXI Write Address Channel Signals			
S_AXI_CTRL_AWADDR	C_S_AXI_CTRL_ADDR_WIDTH-1:0	I	AXI Write address. The write address bus gives the address of the write transaction.
S_AXI_CTRL_AWVALID		I	Write address valid. This signal indicates that valid write address is available.
S_AXI_CTRL_AWREADY		O	Write address ready. This signal indicates that the slave is ready to accept an address.
AXI Write Channel Signals			
S_AXI_CTRL_WDATA	C_S_AXI_CTRL_DATA_WIDTH-1: 0	I	Write data
S_AXI_CTRL_WSTB	C_S_AXI_CTRL_DATA_WIDTH/8-1:0	I	Write strobes. This signal indicates which byte lanes to update in memory.
S_AXI_CTRL_WVALID		I	Write valid. This signal indicates that valid write data and strobes are available.
S_AXI_CTRL_WREADY		O	Write ready. This signal indicates that the slave can accept the write data.
AXI Write Response Channel Signals			
S_AXI_CTRL_BRESP	1:0	O	Write response. This signal indicates the status of the write transaction. 00 - OKAY 10 - SLVERR 11 - DECERR
S_AXI_CTRL_BVALID		O	Write response valid. This signal indicates that a valid write response is available.
S_AXI_CTRL_BREADY		I	Response ready. This signal indicates that the master can accept the response information.
AXI Read Address Channel Signals			
S_AXI_CTRL_ARADDR	C_S_AXI_CTRL_ADDR_WIDTH-1:0	I	Read address. The read address bus gives the address of a read transaction.

Table 2-1: **LMB BRAM Interface Controller I/O Signals *(Cont'd)***

Port Name	MSB:LSB	I/O	Description
S_AXI_CTRL_ARVALID		I	Read address valid. This signal indicates, when HIGH, that the read address is valid and remains stable until the address acknowledgment signal, S_AXI_CTRL_ARREADY, is High.
S_AXI_CTRL_ARREADY		O	Read address ready. This signal indicates that the slave is ready to accept an address.
AXI Read Data Channel Signals			
S_AXI_CTRL_RDATA	C_S_AXI_CTRL_DATA_WIDTH-1:0	O	Read data
S_AXI_CTRL_RRESP	1:0	O	Read response. This signal indicates the status of the read transfer. 00 - OKAY 10 - SLVERR 11 - DECERR
S_AXI_CTRL_RVALID		O	Read valid. This signal indicates that the required read data is available and the read transfer can complete
S_AXI_CTRL_RREADY		I	Read ready. This signal indicates that the master can accept the read data and response information

Register Space

Table 2-2 shows the Register Address Map for the LMB BRAM Interface Controller. The individual registers are described in Table 2-3 to Table 2-26.

Table 2-2: **LMB BRAM Interface Register Address Map**

Offset (hex)	Register	Access Type	Description
0x0	ECC_STATUS	R/W	ECC Status Register
0x4	ECC_EN_IRQ	R/W	ECC Enable Interrupt Register
0x8	ECC_ONOFF	R/W	ECC On/Off Register
0xC	CE_CNT	R/W	Correctable Error Counter Register
0x100	CE_FFD	R	Correctable Error First Failing Data Register
0x180	CE_FFE	R	Correctable Error First Failing ECC Register
0x1C0	CE_FFA	R	Correctable Error First Failing Address Register
0x200	UE_FFD	R	Uncorrectable Error First Failing Data Register
0x280	UE_FFE	R	Uncorrectable Error First Failing ECC Register

Table 2-2: **LMB BRAM Interface Register Address Map *(Cont'd)***

Offset (hex)	Register	Access Type	Description
0x2C0	UE_FFA	R	Uncorrectable Error First Failing Address Register
0x300	FI_D	W	Fault Inject Data Register
0x380	FI_ECC	W	Fault Inject ECC Register

ECC Status Register (ECC_STATUS)

This register holds information about correctable and uncorrectable errors. The status bits are independently set to 1 for the first occurrence of each error type. The status bits are cleared by writing a 1 to the corresponding bit position, that is, the status bits can only be cleared to 0 and not set to 1 by means of a register write. The ECC Status register operates independently of the ECC Enable Interrupt register.

The register is implemented if C_ECC_STATUS_REGISTERS is set to 1.

Table 2-3: **ECC Status Register (ECC_STATUS)**

Reserved		ECC_STATUS	
0	29	30	31

Table 2-4: **ECC Status Register Bit Definitions**

Bit(s)	Name	Core Access	Reset Value	Description
30	CE_STATUS	R/W	0	If 1 a correctable error has occurred. Cleared when 1 is written to this bit position
31	UE_STATUS	R/W	0	If 1 an uncorrectable error has occurred. Cleared when 1 is written to this bit position

ECC Interrupt Enable Register (ECC_EN_IRQ)

This register determines if the value of the CE_STATUS and UE_STATUS bits of the ECC Status Register asserts the Interrupt output signal. If both CE_EN_IRQ and UE_EN_IRQ are set to 1 (enabled), the value of the Interrupt signal is the logical OR between the CE_STATUS and UE_STATUS bits.

The register is implemented if C_ECC_STATUS_REGISTERS is set to 1.

Table 2-5: **ECC Interrupt Enable Register (ECC_EN_IRQ)**

Reserved		ECC_EN_IRQ	
0	29	30	31

Table 2-6: **ECC Interrupt Enable Register Bit Definitions**

Bit(s)	Name	Core Access	Reset Value	Description
30	CE_EN_IRQ	R/W	0	If 1, the value of the CE_STATUS bit of the ECC Status Register is propagated to the Interrupt signal. if 0, the value of the CE_STATUS bit of ECC Status Register is not propagated to the Interrupt signal.
31	UE_EN_IRQ	R/W	0	If 1, the value of the UE_STATUS bit of ECC Status Register is propagated to the Interrupt signal. if 0, the value of the UE_STATUS bit of ECC Status Register is not propagated to the Interrupt signal.

ECC On/Off Register (ECC_ONOFF)

This register determines if the ECC checking should be enabled. ECC checking should normally never be disabled. However, in the case where the block RAM ECC bits have not been initialized at startup, they must be manually initialized before enabling the ECC checking. The ECC initialization is done by performing a read followed by a write on the whole block RAM contents.

The register is implemented if C_ECC_ONOFF_REGISTER is set to 1.

Table 2-7: **ECC On/Off Register (ECC_ONOFF)**

Reserved		ECC_ONOFF
0	30	31

Table 2-8: **ECC On/Off Register Bit Definitions**

Bit(s)	Name	Core Access	Reset Value	Description
31	ECC_ONOFF	R/W	C_ECC_ONOFF_RESET_VALUE	If 1 ECC checking is enabled. if 0 ECC checking is disabled.

Correctable Error Counter Register (CE_CNT)

This registers counts the number of occurrences of correctable errors. It can be cleared or preset to any value by means of a register write. When the counter reaches its maximum value it does not wrap around, but rather stops incrementing and remains at the maximum value.

The width of the counter is defined by the value of the C_CE_COUNTER_WIDTH parameter. This register is not implemented if the value of C_CE_COUNTER_WIDTH is 0.

Table 2-9: **Correctable Error Counter Register (CE_CNT)**

Reserved		CE_CNT	
0	31-C_CE_COUNTER_WIDTH	32-C_CE_COUNTER_WIDTH	31

Table 2-10: **Correctable Error Counter Register Bit Definitions**

Bit(s)	Name	Core Access	Reset Value	Description
(32-C_CE_COUNTER_WIDTH) to 31	CE_CNT	R/W	0	Registers holds number of correctable errors encountered

Correctable Error First Failing Data Register (CE_FFD)

This register stores the (uncorrected) failing data of the first occurrence of an access with a correctable error. When the CE_STATUS bit in the ECC Status Register is cleared, this register is re-enabled to store the data of the next correctable error. Storing of failing data is enabled after reset.

The register is implemented if the C_CE_FAILING_REGISTERS is set to 1.

Table 2-11: **Correctable Error First Failing Data Register (CE_FFD)**

CE_FFD	
0	31

Table 2-12: **Correctable Error First Failing Data Register Bit Definitions**

Bit(s)	Name	Core Access	Reset Value	Description
0 to 31	CE_FFD	R	0	Data of the first occurrence of a correctable error

Correctable Error First Failing ECC Register (CE_FFE)

This register stores the ECC of the first occurrence of an access with a correctable error. When the CE_STATUS bit in the ECC Status Register is cleared, this register is re-enabled to store the ECC of the next correctable error. Storing of the failing ECC is enabled after reset.

The register is implemented if C_CE_FAILING_REGISTERS is set to 1.

Table 2-13: **Correctable Error First Failing ECC Register (CE_FFE)**

Reserved		CE_FFE	
0	24	25	31

Table 2-14: **Correctable Error First Failing ECC Register Bit Definitions**

Bit(s)	Name	Core Access	Reset Value	Description
25 to 31	CE_FFE	R	0	ECC of the first occurrence of a correctable error

Correctable Error First Failing Address Register (CE_FFA)

This register stores the address of the first occurrence of an access with a correctable error. When the CE_STATUS bit in the ECC Status Register is cleared, this register is re-enabled to store the address of the next correctable error. Storing of the failing address is enabled after reset.

The register is implemented if C_CE_FAILING_REGISTERS is set to 1.

Table 2-15: **Correctable Error First Failing Address Register (CE_FFA)**

CE_FFA	
0	31

Table 2-16: **Correctable Error First Failing Address Register Bit Definitions**

Bit(s)	Name	Core Access	Reset Value	Description
0 to 31	CE_FFA	R	0	Address of the first occurrence of a correctable error

Uncorrectable Error First Failing Data Register (UE_FFD)

This register stores the failing data of the first occurrence of an access with an uncorrectable error. When the UE_STATUS bit in the ECC Status Register is cleared, this register is re-enabled to store the data of the next uncorrectable error. Storing of failing data is enabled after reset.

The register is implemented if C_UE_FAILING_REGISTERS is set to 1.

Table 2-17: **Uncorrectable Error First Failing Data Register (UE_FFD)**

UE_FFD	
0	31

Table 2-18: **Uncorrectable Error First Failing Data Register Bit Definitions**

Bit(s)	Name	Core Access	Reset Value	Description
0 to 31	UE_FFD	R	0	Data of the first occurrence of an uncorrectable error

Uncorrectable Error First Failing ECC Register (UE_FFE)

This register stores the ECC of the first occurrence of an access with a uncorrectable error. When the UE_STATUS bit in the ECC Status Register is cleared, this register is re-enabled to store the ECC of the next uncorrectable error. Storing of the failing ECC is enabled after reset.

The register is implemented if C_UE_FAILING_REGISTERS is set to 1.

Table 2-19: **Uncorrectable Error First Failing ECC Register (UE_FFE)**

Reserved		UE_FFE	
0	24	25	31

Table 2-20: **Uncorrectable Error First Failing ECC Register Bit Definitions**

Bit(s)	Name	Core Access	Reset Value	Description
25 to 31	UE_FFE	R	0	ECC of the first occurrence of an uncorrectable error

Uncorrectable Error First Failing Address Register (UE_FFA)

This register stores the address of the first occurrence of an access with an uncorrectable error. When the UE_STATUS bit in the ECC Status Register is cleared, this register is re-enabled to store the address of the next uncorrectable error. Storing of the failing address is enabled after reset.

The register is implemented if C_UE_FAILING_REGISTERS is set to 1.

Table 2-21: **Uncorrectable Error First Failing Address Register (UE_FFA)**

UE_FFA	
0	31

Table 2-22: **Uncorrectable Error First Failing Address Register Bit Definitions**

Bit(s)	Name	Core Access	Reset Value	Description
0 to 31	UE_FFA	R	0	Address of the first occurrence of an uncorrectable error

Fault Injection Data Register (FI_D)

This register is used to inject errors in data written to the block RAM and can be used to test the error correction and error signalling. The bits set in the register toggle the corresponding data bits of the subsequent data written to the block RAM without affecting the ECC bits written. After the fault has been injected, the Fault Injection Data Register is cleared automatically.

The register is implemented if C_FAULT_INJECT is set to 1.

IMPORTANT: *Injecting faults should be performed in a critical region in software; that is, writing to this register and the subsequent write to the LMB BRAM must not be interrupted.*

Table 2-23: **Fault Injection Data Register (FI_D)**

FI_D	
0	31

Table 2-24: **Fault Injection Data Register Bit Definitions**

Bit(s)	Name	Core Access	Reset Value	Description
0 to 31	FI_D	W	0	Bit positions set to 1 toggle the corresponding bits of the next data word written to the LMB BRAM. The register is automatically cleared after the fault has been injected.

Fault Injection ECC Register (FI_ECC)

This register is used to inject errors in the generated ECC written to the block RAM and can be used to test the error correction and error signalling. The bits set in the register toggle the corresponding ECC bits of the next data written to block RAM. After the fault has been injected, the Fault Injection ECC Register is cleared automatically.

The register is implemented if C_FAULT_INJECT is set to 1.

IMPORTANT: *Injecting faults should be performed in a critical region in software, that is, writing to this register and the subsequent write to LMB BRAM must not be interrupted.*

Table 2-25: **Fault Injection ECC Register (FI_ECC)**

Reserved		FI_ECC	
0	24	25	31

Table 2-26: **Fault Injection ECC Register Bit Definitions**

Bit(s)	Name	Core Access	Reset Value	Description
25 to 31	FI_ECC	R	0	Bit positions set to 1 toggle the corresponding bit of the next ECC written to the LMB BRAM. The register is automatically cleared after the fault has been injected.

Designing with the Core

This chapter includes guidelines and additional information to facilitate designing with the core.

General Design Guidelines

In a MicroBlaze™ system without Error Correction Codes (ECC) protection, the LMB BRAM Interface Controller is typically connected as in Figure 3-1.

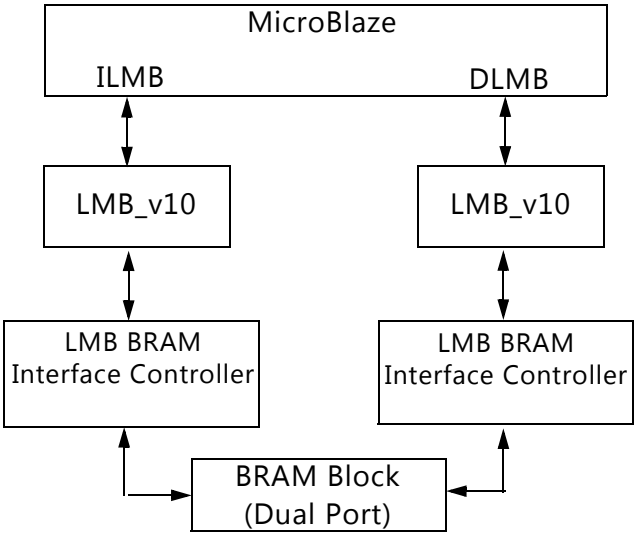

Figure 3-1: **Typical MicroBlaze System**

The Interrupt output and the AXI4-Lite interfaces are unconnected, and the `BRAM_DIn_A` and `BRAM_DOut_A` signals only contain 32 data bits.

The LMB BRAM Interface Controller supports multiple LMB masters, making it possible to use the second block RAM port for low latency data communication with MicroBlaze. The LMB Interface Controller would in this case be connected, as in Figure 3-2.

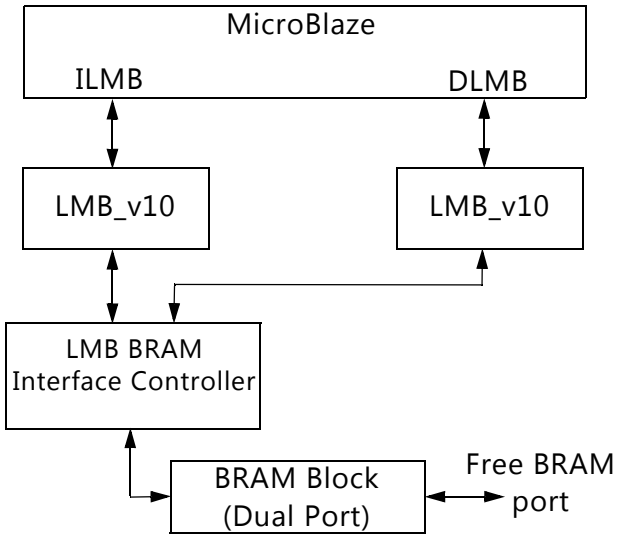

Figure 3-2: **MicroBlaze System with Multiplexed ILMB and DLMB**

Note that MicroBlaze performance will drop somewhat, since the DLMB and ILMB accesses now cannot be performed concurrently. The performance reduction is application dependent, but can be expected to be 10–20%.

When the LMB BRAM Interface Controller supports more than one master, there is a fixed priority order between the LMB ports. SLMB has the highest priority and in decreasing priority order, SLMB1, SLMB2 and SLMB3. To minimize the negative performance impact MicroBlaze DLMB should be given the highest priority, which means that it should be connected to SLMB and MicroBlaze ILMB to SLMB1.

LMB Controller With ECC

To mitigate the effect of block RAM Single Event Upsets (SEU), the LMB BRAM Interface Controller can be configured to use Error Correction Codes (ECC). When writing to the block RAM, ECC bits are generated and stored together with the written data. When reading from the block RAM, the ECC bits are used to correct all single bit errors and detect all double bit errors in the data read. Errors are either signalled by the LMB to MicroBlaze or by an interrupt signal. The ECC used is a (32,7) Hamming code, as defined in Table 3-1.

Table 3-1: **ECC Coding**

Participating Data Bits	ECC0	ECC1	ECC2	ECC3	ECC4	ECC5	ECC6
0	*	*					*
1	*		*				*
2		*	*				*
3	*	*	*				
4	*			*			*
5		*		*			*
6	*	*		*			
7			*	*			*
8	*		*	*			
9		*	*	*			
10	*	*	*	*			*
11	*				*		*
12		*			*		*
13	*	*			*		
14			*		*		*
15	*		*		*		
16		*	*		*		
17	*	*	*		*		*
18				*	*		*
19	*			*	*		
20		*		*	*		
21	*	*		*	*		*
22			*	*	*		
23	*		*	*	*		*
24		*	*	*	*		*
25	*	*	*	*	*		
26	*					*	*
27		*				*	*
28	*	*				*	
29			*			*	*
30	*		*			*	
31		*	*			*	

The ECC encoding corresponds to that shown in the Xilinx Application Note, *Single Error Correction and Double Error Detection* (XAPP645) [Ref 3], but is shown here in its optimized form.

The need to store the ECC increases the block RAM utilization depending on block RAM data size. The overhead is listed in Table 3-2.

Table 3-2: **ECC Block RAM Overhead**

Block RAM Data Size	ECC Overhead
4 kB	100%
8 kB	50%
16 kB and larger	25%

A set of optional registers in the LMB BRAM Interface Controller controls the operation of the ECC logic. The registers are accessed through an AXI4-Lite slave interface. The slave interface is connected to MicroBlaze M_AXI_DP ports in a typical system, according to Figure 3-3.

The LMB BRAM Interface Controller requires that the AXI4-Lite bus is synchronous to `LMB_Clk`.

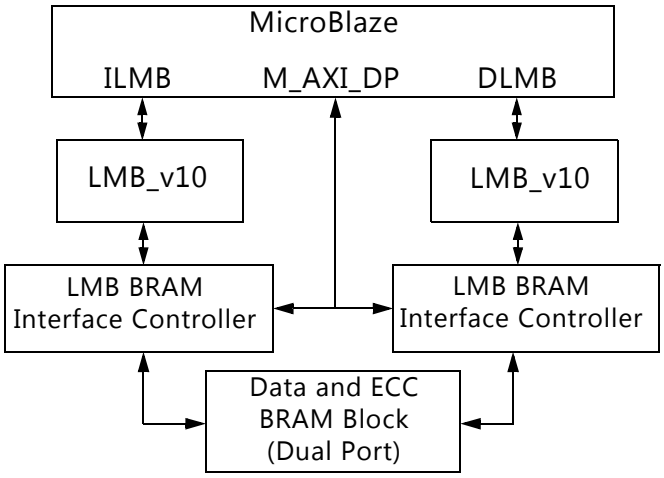

Figure 3-3: **Typical MicroBlaze System Using ECC**

ECC Initialization

The ECC bits are normally initialized by the Vivado Design Suite or the standalone `updatemem` tool (see the *Vivado Design Suite User Guide: Embedded Processor Hardware Design* (UG898) [Ref 4]). However, they can also be initialized by software running on MicroBlaze. The initialization is performed by reading and writing back the complete contents of the block RAM data while ECC checking is suppressed, and then enabling it by writing 1 to the ECC On/Off Register. The ECC checking is disabled when the parameter C_ECC_ONOFF_REGISTER = 1 and the parameter C_ECC_ONOFF_RESET_VALUE = 0, which causes the initial value in the ECC On/Off Register to be 0.

ECC Use Cases

The use cases below represent possible system configuration scenarios that the LMB BRAM Interface Controller supports. However, other configurations are possible, since the parameters are individually configurable.

Minimal

This system is suitable when area constraints are high, and there is no need for testing of the ECC function, or analysis of error frequency and location. No ECC registers are implemented. Single bit errors are corrected by the ECC logic before being passed to MicroBlaze. Uncorrectable errors are signalled by asserting the LMB SI_UE signal, which generates an exception in MicroBlaze. Parameter set is C_ECC = 1.

Small

This system should be used when it is required to monitor error frequency, but there is no need for testing of the ECC function. Minimal system with Correctable Error Counter Register added to monitor single bit error rates. If the error rate is too high, the scrubbing rate should be increased to minimize the risk of a single bit error becoming an uncorrectable double bit error. Parameters set are C_ECC = 1 and C_CE_COUNTER_WIDTH = 10.

Typical

This system represents a typical use case, where it is required to monitor error frequency, as well as generating an interrupt to immediately correct a single bit error through software. It does not provide support for testing the ECC function.

This is a small system with the addition of Correctable Error First Failing registers and a Status register. A single bit error latches the address for the access into the Correctable Error First Failing Address Register and sets the CE_STATUS bit in the ECC Status Register. An interrupt is generated, triggering MicroBlaze to read the failing address and then perform a read followed by a write on the failing address. This removes the single bit error from the block RAM, thus reducing the risk of the single bit error becoming a uncorrectable double bit error. Parameters set are C_ECC = 1, C_CE_COUNTER_WIDTH = 10, C_ECC_STATUS_REGISTER = 1 and C_CE_FAILING_REGISTERS = 1.

Full

This system uses all of the features provided by the LMB BRAM Interface Controller, to enable full error injection capability, as well as error monitoring and interrupt generation. This is a typical system with the addition of Uncorrectable Error First Failing registers and Fault Injection registers. All features switched on for full control of ECC functionality for system debug or systems with high fault tolerance requirements. Parameters set are C_ECC = 1, C_CE_COUNTER_WIDTH = 10, C_ECC_STATUS_REGISTER = 1, C_CE_FAILING_REGISTERS = 1, C_UE_FAILING_REGISTERS = 1 and C_FAULT_INJECT = 1.

Clocking

The LMB BRAM Interface Controller is fully synchronous with all clocked elements clocked with the `LMB_Clk`.

The `S_AXI_CTRL_ACLK` input is not used and should be left unconnected.

The `BRAM_Clk_A` is an output clock used for clocking the LMB BRAM Interface Controller.

Resets

The `LMB_Rst` is the master reset input signal for the LMB BRAM Interface Controller.

The `BRAM_Rst_A` output signal is tied to 0 and could be left unconnected.

The `S_AXI_CTRL_ARESETN` input is not used and should be left unconnected.

Protocol Description

See the LMB Interface Description timing diagrams in the *MicroBlaze Processor Reference Guide* (UG984) [Ref 1].

Design Flow Steps

This chapter describes customizing and generating the core, constraining the core, and the simulation, synthesis and implementation steps that are specific to this IP core. More detailed information about the standard Vivado® design flows and the IP integrator can be found in the following Vivado Design Suite user guides:

- *Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator* (UG994) [Ref 5]
- *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 6]
- *Vivado Design Suite User Guide: Getting Started* (UG910) [Ref 7]
- *Vivado Design Suite User Guide: Logic Simulation* (UG900) [Ref 8]

Customizing and Generating the Core

This section includes information about using Xilinx tools to customize and generate the core in the Vivado Design Suite.

If you are customizing and generating the core in the Vivado IP integrator, see the *Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator* (UG994) [Ref 5] for detailed information. IP integrator might auto-compute certain configuration values when validating or generating the design. To check whether the values do change, see the description of the parameter in this chapter. To view the parameter value, run the `validate_bd_design` command in the Tcl console.

You can customize the IP for use in your design by specifying values for the various parameters associated with the IP core using the following steps:

1. Select the IP from the Vivado IP catalog.
2. Double-click the selected IP or select the **Customize IP** command from the toolbar or right-click menu.

For details, see the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 6] and the *Vivado Design Suite User Guide: Getting Started* (UG910) [Ref 7].

Note: Figures in this chapter are illustrations of the Vivado Integrated Design Environment (IDE). The layout depicted here might vary from the current version.

The LMB BRAM Interface Controller parameters are divided in two categories: Addresses and ECC. When using Vivado IP integrator feature, the addresses and masks are auto-generated.

The Addresses parameter configuration screen is shown in Figure 4-1.

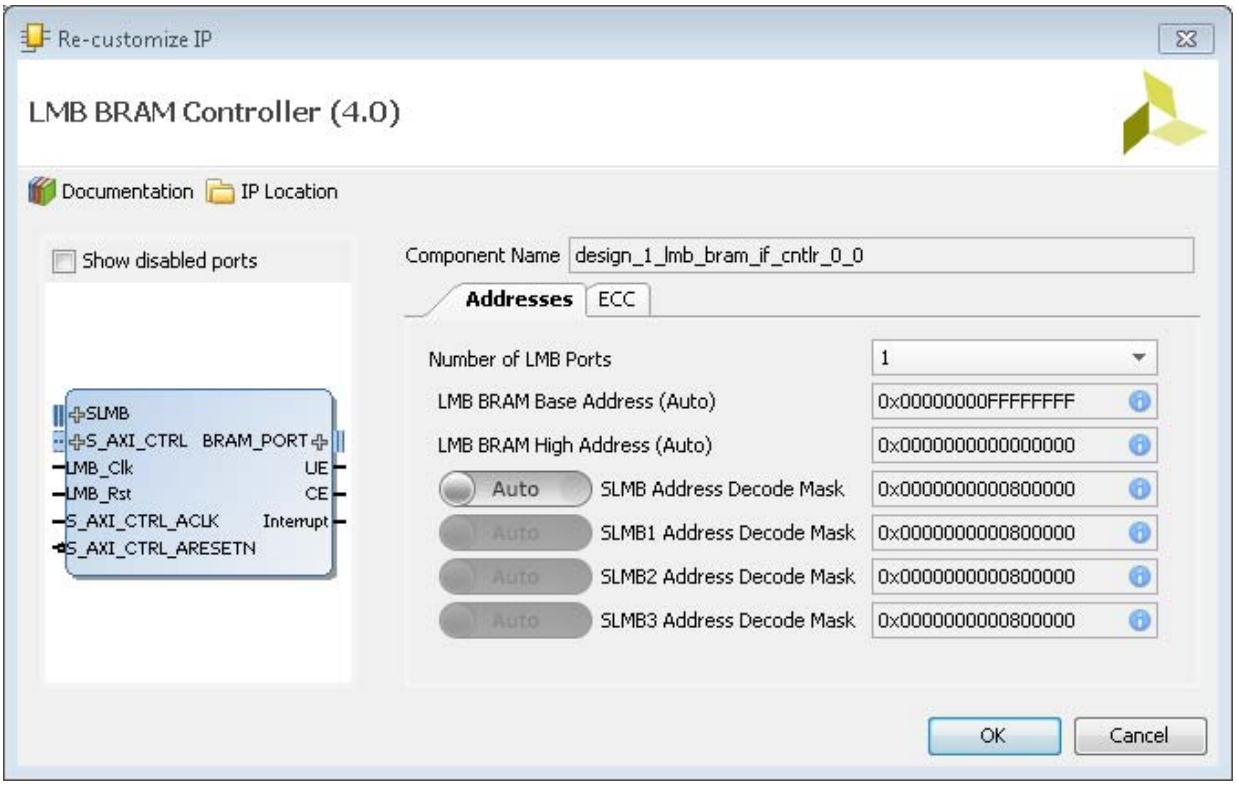

Figure 4-1: **Addresses Parameter Tab**

- **Number of LMB Ports** - Sets the number of ports available to connect to MicroBlaze™.
- **LMB BRAM Base Address** - Base address of the local memory with up to 64 bits.
- **LMB BRAM High Address** - High address of the local memory with up to 64 bits.
- **SLMB/SLMB1/SLMB2/SLMB3 Address Decode Mask** - A mask indicating which address bits the LMB BRAM Interface Controller takes into account when decoding an access with up to 64 bits.

The ECC parameter configuration screen is shown in Figure 4-2.

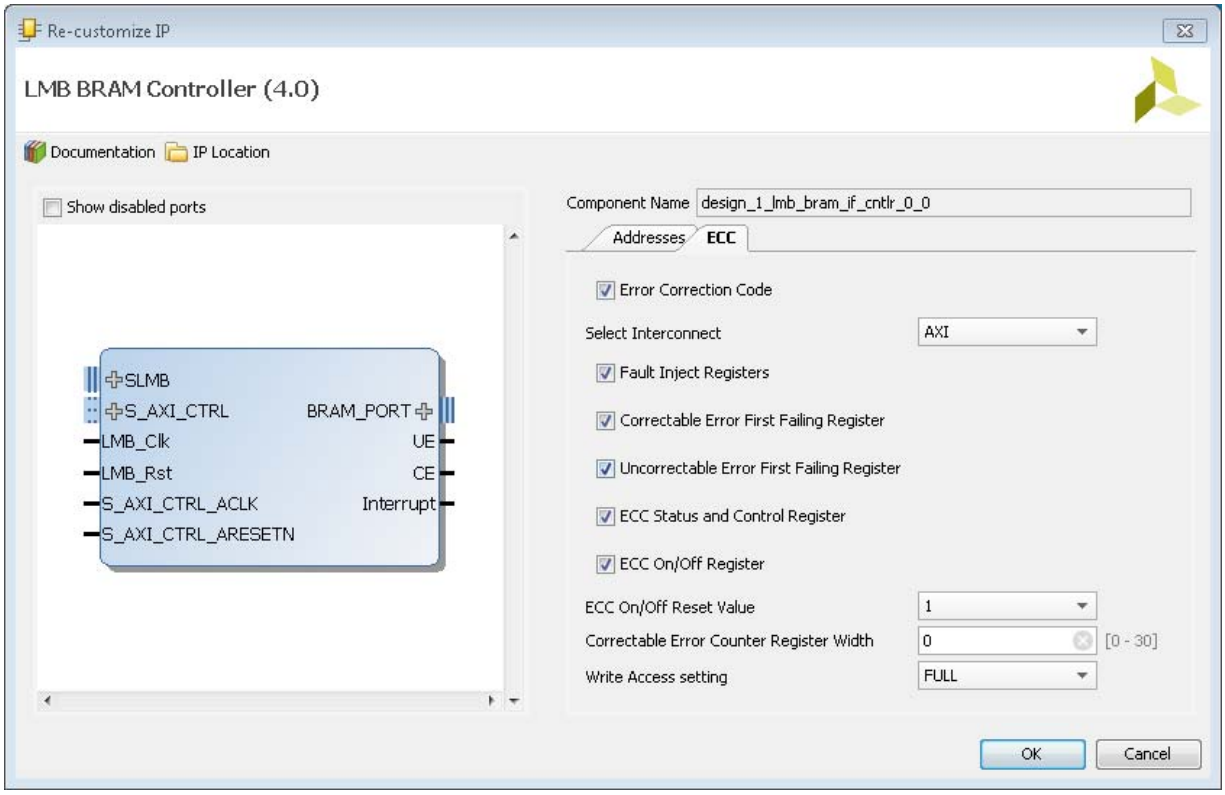

Figure 4-2: **ECC Parameter Tab**

- **Error Correction Code** - Enables Error Correction Code, to correct single bit errors and detect double bit errors.
- **Select Interconnect** - Can be set to *None* for basic functionality, or *AXI* to access ECC registers.
- **Fault Inject Registers** - Enable fault inject registers to allow testing of the ECC functionality.
- **Correctable Error First Failing Register** - Enable this register to store the first failing address of a correctable error.
- **Uncorrectable Error First Failing Register** - Enable this register to store the first failing address of an uncorrectable error.
- **ECC Status and Control Register** - Enable these registers to read ECC status and control ECC generation.
- **ECC On/Off Register** - Enable this register to be able to toggle ECC functionality.
- **ECC On/Off Reset Value** - Set to 1 to enable ECC or 0 to disable ECC after reset.
- **Correctable Error Counter Register Width** - Determines how many correctable errors can be counted. The value 0 means that the register is not implemented.

- **Write Access Setting** - Can be set to *Full*, *Word only* or *None*. Should normally be set to *Full* for Data LMB, and *None* for Instruction LMB.

Parameter Values

To obtain an LMB BRAM Interface Controller that is uniquely tailored a specific system, certain features can be parameterized in the LMB BRAM Interface Controller design. This allows you to configure a design that only uses the resources required by the system, and operates with the best possible performance. The features that can be parameterized in Xilinx LMB BRAM Interface Controller designs are shown in Table 4-1.

Table 4-1: **LMB BRAM Interface Controller Parameters**

Parameter Name	Feature/Description	Allowable Values	Default Value	VHDL Type
Basic Parameters				
C_BASEADDR	LMB BRAM Base Address	Valid Address Range(2)	None(1)	std_logic_vector
C_HIGHADDR	LMB BRAM HIGH Address	Valid Address Range(2)	None(1)	std_logic_vector
C_MASK	LMB Decode Mask	Valid decode mask for SLMB(3)	0x00000000 00800000	std_logic_vector
C_MASK1	LMB Decode Mask	Valid decode mask for SLMB1(3)	0x00000000 00800000	std_logic_vector
C_MASK2	LMB Decode Mask	Valid decode mask for SLMB2(3)	0x00000000 00800000	std_logic_vector
C_MASK3	LMB Decode Mask	Valid decode mask for SLMB3(3)	0x00000000 00800000	std_logic_vector
ECC Parameters				
C_ECC	Implement Error Correction and Detection	0=No ECC 1=ECC	0	integer
C_INTERCONNECT(4)	Select type of register access interface	0=No interface 2=AXI4-Lite	0	integer
C_FAULT_INJECT(4)	Implement Fault Injection registers	0=No fault inject register 1=Fault inject registers	0	integer
C_CE_FAILING_REGISTERS(4)	Implement First Failing Address, Data and ECC registers for correctable error	0=No CE failing registers 1=CE failing registers	0	integer

Table 4-1: **LMB BRAM Interface Controller Parameters *(Cont'd)***

Parameter Name	Feature/Description	Allowable Values	Default Value	VHDL Type
C_UE_FAILING_REGISTERS[4]	Implement First Failing Address, Data and ECC registers for uncorrectable error	0=No UE failing registers 1=UE failing registers	0	integer
C_ECC_STATUS_REGISTERS[4]	Implement status and interrupt registers	0=Interrupt not generated and no status register 1=Interrupt available and status register	0	integer
C_ECC_ONOFF_REGISTER[4]	Implement register to enable/disable ECC checking	0=ECC checking is always enabled 1=ECC checking is controlled by the value in this register	0	integer
C_ECC_ONOFF_RESET_VALUE[4]	Selects reset value for ECC On/Off Register	0=ECC On/Off Register is initialized to 0 at reset 1= ECC On/Off Register is initialized to 1 at reset	1	integer
C_CE_COUNTER_WIDTH[4]	Correctable Error Counter width	0=No CE Counter 1-31=Width of CE Counter	0	integer
C_WRITE_ACCESS[4]	LMB access types	0=No LMB write 1=Only 32-bit word write 2=8-, 16- and 32 bit writes	2	integer

Notes:

1. No default value is specified for BASEADDRand HIGHADDR to ensure that the actual value is set; if the value is not set, a compiler error is generated. These generics must be a power of 2. BASEADDR must be a multiple of the range, where the range is HIGHADDR - BASEADDR +1.
2. The range specified by BASEADDR and HIGHADDR must comprise a complete, contiguous power-of-two range, such that range = 2^n, and the n least significant bits of BASEADDR must be zero.
3. The decode mask determines which bits are used by the LMB decode logic to decode a valid access to LMB.
4. Parameter value is don't care unless parameter C_ECC = 1

C_ECC

Unless error correction and detection is enabled, all ECC related parameters are "don't care".

C_INTERCONNECT

When error correction and detection is enabled (C_ECC = 1) and any register parameters are enabled, an interface to access the registers is needed. The register access interface can be of AXI4-Lite type. The parameters related to AXI4-Lite are 'don't care' unless enabled by the value of C_INTERCONNECT.

C_ECC_STATUS_REGISTERS

This parameter enables the ECC Status Register and the ECC Interrupt Enable Register and the generation of the external Interrupt signal.

Parameter - Port Dependencies

The width of many of the LMB BRAM Interface Controller signals depends on the number of memories in the system and the width of the various data and address buses. The dependencies between the LMB BRAM Interface Controller design parameters and I/O signals are shown in Table 4-2.

Table 4-2: **Parameter-Port Dependencies**

Parameter Name	Ports (Port width depends on parameter)
C_ECC	BRAM_WEN_A, BRAM_Din_A, BRAM_Dout_A

Programming Model

Supported Memory Sizes

For supported block RAM memory sizes, see the *Block Memory Generator, LogiCORE IP Product Guide* (PG058) [Ref 9].

Example Base Address, High Address Specifications

The base address (C_BASEADDR) and high address (C_HIGHADDR) must specify a valid range for the block RAM that is attached to the LMB BRAM Interface Controller. The range (C_HIGHADDR–C_BASEADDR) specified by the Offset Address and Range in Vivado IP integrator must be equal to 2^n bytes, where *n* is a positive integer and 2^n is a valid memory size as shown above. In addition, the *n* least significant bits of C_BASEADDR must be equal to 0.

LMB Timing

See the MicroBlaze Bus Interfaces chapter in the *MicroBlaze Processor Reference Guide* (UG984) [Ref 1] for details on the transaction signaling.

User Parameters

Table 4-3 shows the relationship between the fields in the Vivado IDE and the User Parameters (which can be viewed in the Tcl console).

Table 4-3: **Vivado IDE Parameter to User Parameter Relationship**

Vivado IDE Parameter	User Parameter	Default Value
Number of LMB Ports	C_NUM_LMB	1
LMB BRAM Base Address	C_BASEADDR	0xFFFFFFFFFFFFFFFF
LMB BRAM High Address	C_HIGHADDR	0x0000000000000000
SLMB Address Decode Mask	C_MASK	0x0000000000800000
SLMB1 Address Decode Mask	C_MASK1	0x0000000000800000
SLMB2 Address Decode Mask	C_MASK2	0x0000000000800000
SLMB3 Address Decode Mask	C_MASK3	0x0000000000800000
Error Correction Code	C_ECC	0
Select Interconnect	C_INTERCONNECT	None
Fault Inject Registers	C_FAULT_INJECT	0
Correctable Error First Failing Register	C_CE_FAILING_REGISTERS	0
Uncorrectable Error First Failing Register	C_UE_FAILING_REGISTERS	0
ECC Status and Control Register	C_ECC_STATUS_REGISTERS	0
ECC On/Off Register	C_ECC_ONOFF_REGISTER	0
ECC On/Off Reset Value	C_ECC_ONOFF_RESET_VALUE	1
Correctable Error Counter Register Width	C_CE_COUNTER_WIDTH	0
Write Access setting	C_WRITE_ACCESS	FULL

Output Generation

The following files are generated by the IP core in Vivado IP integrator.

- Verilog/VHDL template,
- VHDL source files
- VHDL wrapper file in the library work

For details, see the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 6].

Constraining the Core

This section contains information about constraining the core in the Vivado Design Suite.

Required Constraints

This section is not applicable for this IP core.

Device, Package, and Speed Grade Selections

This section is not applicable for this IP core.

Clock Frequencies

This section is not applicable for this IP core.

Clock Management

The LMB BRAM Interface Controller is fully synchronous with all clocked elements clocked by the LMB_Clk input.

To operate properly when connected to MicroBlaze, the LMB_Clk must be the same as MicroBlaze Clk.

Clock Placement

This section is not applicable for this IP core.

Banking

This section is not applicable for this IP core.

Transceiver Placement

This section is not applicable for this IP core.

I/O Standard and Placement

This section is not applicable for this IP core.

Simulation

For comprehensive information about Vivado simulation components, as well as information about using supported third-party tools, see the *Vivado Design Suite User Guide: Logic Simulation* (UG900) [Ref 8].

IMPORTANT: *For cores targeting 7 series or Zynq-7000 devices, UNIFAST libraries are not supported. Xilinx IP is tested and qualified with UNISIM libraries only.*

Synthesis and Implementation

For details about synthesis and implementation, see the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 6].

Appendix A

Migrating

This appendix contains information about upgrading to a more recent version of the IP core.

Migrating to the Vivado Design Suite

For information about migrating to the Vivado® Design Suite, see the *ISE to Vivado Design Suite Migration Guide* (UG911) [Ref 10].

Debugging

This appendix includes details about resources available on the Xilinx Support website and debugging tools.

Finding Help on Xilinx.com

To help in the design and debug process when using the LMB BRAM Interface Controller, the Xilinx Support web page contains key resources such as product documentation, release notes, answer records, information about known issues, and links for obtaining further product support.

Documentation

This product guide is the main document associated with the LMB BRAM Interface Controller. This guide, along with documentation related to all products that aid in the design process, can be found on the Xilinx Support web page or by using the Xilinx Documentation Navigator.

Download the Xilinx Documentation Navigator from the Downloads page. For more information about this tool and the features available, open the online help after installation.

Answer Records

Answer Records include information about commonly encountered problems, helpful information on how to resolve these problems, and any known issues with a Xilinx product. Answer Records are created and maintained daily ensuring that users have access to the most accurate information available.

Answer Records for this core can be located by using the Search Support box on the main Xilinx support web page. To maximize your search results, use proper keywords such as

- Product name
- Tool message(s)
- Summary of the issue encountered

A filter search is available after results are returned to further target the results.

Answer Records for theLMB BRAM Interface Controller Core

- AR54407

Technical Support

Xilinx provides technical support at the Xilinx Support web page for this LogiCORE™ IP product when used as described in the product documentation. Xilinx cannot guarantee timing, functionality, or support if you do any of the following:

- Implement the solution in devices that are not defined in the documentation.
- Customize the solution beyond that allowed in the product documentation.
- Change any section of the design labeled DO NOT MODIFY.

To contact Xilinx Technical Support, navigate to the Xilinx Support web page.

Debug Tools

The main tool available to address LMB BRAM Interface Controller design issues is the Vivado® Design Suite debug feature.

Vivado Design Suite Debug Feature

The Vivado® Design Suite debug feature inserts logic analyzer and virtual I/O cores directly into your design. The debug feature also allows you to set trigger conditions to capture application and integrated block port signals in hardware. Captured signals can then be analyzed. This feature in the Vivado IDE is used for logic debugging and validation of a design running in Xilinx devices.

The Vivado logic analyzer is used to interact with the logic debug IP cores, including:

- ILA 2.0 (and later versions)
- VIO 2.0 (and later versions)

See the *Vivado Design Suite User Guide: Programming and Debugging* (UG908) [Ref 11].

Reference Boards

All 7 series Xilinx development boards support the LMB BRAM Interface Controller. These boards can be used to prototype designs and establish that the core can communicate with the system.

Simulation Debug

The simulation debug flow for the Mentor Graphics Questa Simulator (QuestaSim) is described below. A similar approach can be used with other simulators.

- Check for the latest supported versions of QuestaSim in the Xilinx Design Tools: Release Notes Guide. Is this version being used? If not, update to this version.
- If using Verilog, do you have a mixed mode simulation license? If not, obtain a mixed-mode license.
- Ensure that the proper libraries are compiled and mapped. In the Vivado Design Suite this can be done using **Flow > Simulation Settings.**
- Have you associated the intended software program for the MicroBlaze™ processor with the simulation? Use the command **Tools > Associate ELF Files** in the Vivado Design Suite.
- When observing the traffic on the LMB interface connected to the LMB BRAM I/F Controller, see the *MicroBlaze Processor Reference Guide* (UG984) [Ref 1] for the LMB timing.

Hardware Debug

This section provides debug steps for common issues. The Vivado Design Suite debug feature is a valuable resource to use in hardware debug. The signal names mentioned in the following individual sections can be probed using the debug feature to debug specific problems.

Many of these common issues can also be applied to debugging design simulations.

General Checks

Ensure that all the timing constraints were met during implementation.

- Does it work in post-place and route timing simulation? If problems are seen in hardware but not in timing simulation, this could indicate a PCB issue. Ensure that all clock sources are active and clean.
- If using MMCMs in the design, ensure that all MMCMs have obtained lock by monitoring the `LOCKED` port.

LMB Checks

To monitor the LMB interface, the signals `LMB_ABus`, `LMB_WriteDBus`, `LMB_ReadStrobe`, `LMB_AddrStrobe`, `LMB_WriteStrobe`, `LMB_BE`, `Sl_DBus`, and `Sl_Ready` can be connected to the Vivado debug feature. When Error Correction Codes are used, the signals `Sl_Wait`, `Sl_CE`, and `Sl_UE` can also be added.

To sample the interface signals, the Vivado debug feature should use the `LMB_Clk` clock signal.

AXI4-Lite Interface Debug

Read from a register that does not have all 0s as a default to verify that the interface is functional. Output `S_AXI_CTRL_ARREADY` asserts when the read address is valid, and output `S_AXI_CTRL_RVALID` asserts when the read data/response is valid. If the interface is unresponsive, ensure that the following conditions are met:

- The `S_AXI_CTRL_ACLK` input is connected and toggling.
- The interface is not being held in reset, and `S_AXI_CTRL_ARESETN` is an active-Low reset.
- The main core clock `LMB_Clk` is toggling and that the enables are also asserted.
- If the simulation has been run, verify in simulation and/or a Vivado debug capture that the waveform is correct for accessing the AXI4-Lite interface.

Appendix C

Application Software Development

Device Drivers

The LMB BRAM Interface Controller is supported by the block RAM (BRAM) driver, included with Xilinx Software Development Kit.

Additional Resources and Legal Notices

Xilinx Resources

For support resources such as Answers, Documentation, Downloads, and Forums, see Xilinx Support.

References

These documents provide supplemental material useful with this user guide:

1. *MicroBlaze Processor Reference Guide* (UG984)
2. *ARM AMBA AXI4-Lite Protocol Specification* ARM IHI 0022E, registration required
3. *Xilinx Application Note, Single Error Correction and Double Error Detection* (XAPP645)
4. *Vivado Design Suite User Guide: Embedded Processor Hardware Design* (UG898)
5. *Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator* (UG994)
6. *Vivado Design Suite User Guide: Designing with IP* (UG896)
7. *Vivado Design Suite User Guide: Getting Started* (UG910)
8. *Vivado Design Suite User Guide: Logic Simulation* (UG900)
9. *Block Memory Generator, LogiCORE IP Product Guide* (PG058)
10. *ISE to Vivado Design Suite Migration Guide* (UG911)
11. *Vivado Design Suite User Guide: Programming and Debugging* (UG908)

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
04/06/2016	4.0	Updated with description of extended addressing.
11/18/2015	4.0	Added support for UltraScale+ families.
06/24/2015	4.0	Moved performance and resource utilization data to the web.
03/20/2013	1.0	This Product Guide replaces PG061. There are no documentation changes for this release.

Please Read: Important Legal Notices