LogiCORE IP LMB BRAM Interface Controller v3.10c

Product Guide

PG061 December 18, 2012

XILINX®

XILINX

Table of Contents

SECTION I: SUMMARY

Introduction

This document provides the design specification for the LogiCORE IP Local Memory Bus (LMB) Block RAM (BRAM) Interface Controller.

The LMB BRAM Interface Controller connects to an lmb_v10 bus.

Version v3.10c of the LMB BRAM Interface Controller requires MicroBlaze™ v8.00a or higher and lmb_v10 v2.00a or higher.

Features

- LMB v1.0 bus interfaces with byte enable support
- Used in conjunction with EDK bram_block peripheral or Vivado™ 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.

LogiCORE IP Facts Table	
Core Specifics	
Supported Device Family(1)	Zynq™-7000(2), Virtex-7, Kintex™-7, Artix™-7, Virtex-6, Spartan-6, Virtex-5, Virtex®-4, Spartan®-3
Supported User Interfaces	LMB, AXI4-Lite, PLBv46.
Resources	See Table 2-2.
Provided with Core	
Design Files	ISE: VHDL Vivado: RTL
Example Design	Not Provided
Test Bench	Not Provided
Constraints File	Not Provided
Simulation Model	VHDL Behavioral
Supported S/W Driver(3)	Standalone
Tested Design Flows(4)	
Design Entry	Xilinx Platform Studio (XPS) 14.4 Vivado™ Design Suite 2012.4(5)
Simulation	Mentor Graphics ModelSim
Synthesis	Xilinx Synthesis Technology (XST) Vivado Synthesis
Support	
Provided by Xilinx @ www.xilinx.com/support	

Notes:

1. For a complete list of supported derivative devices, see the IDS Embedded Edition Derivative Device Support.
2. Supported in ISE Design Suite implementations only.
3. Standalone driver details can be found in the EDK or SDK directory (*<install_directory>*/doc/usenglish/xilinx_drivers.htm). Linux OS and driver support information is available from //wiki.xilinx.com.
4. For the supported versions of the tools, see the Xilinx Design Tools: Release Notes Guide.
5. Supports only 7 series devices.

Overview

The LMB BRAM Interface Controller is the interface between the LMB and the EDK bram_block peripheral or Vivado™ 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 are shown in Figure 1-1. The detailed list of signals are listed and described in Table 2-3. See the description of LMB Signals in the MicroBlaze™ Bus Interfaces chapter in the *MicroBlaze Processor Reference Guide* [Ref 1].

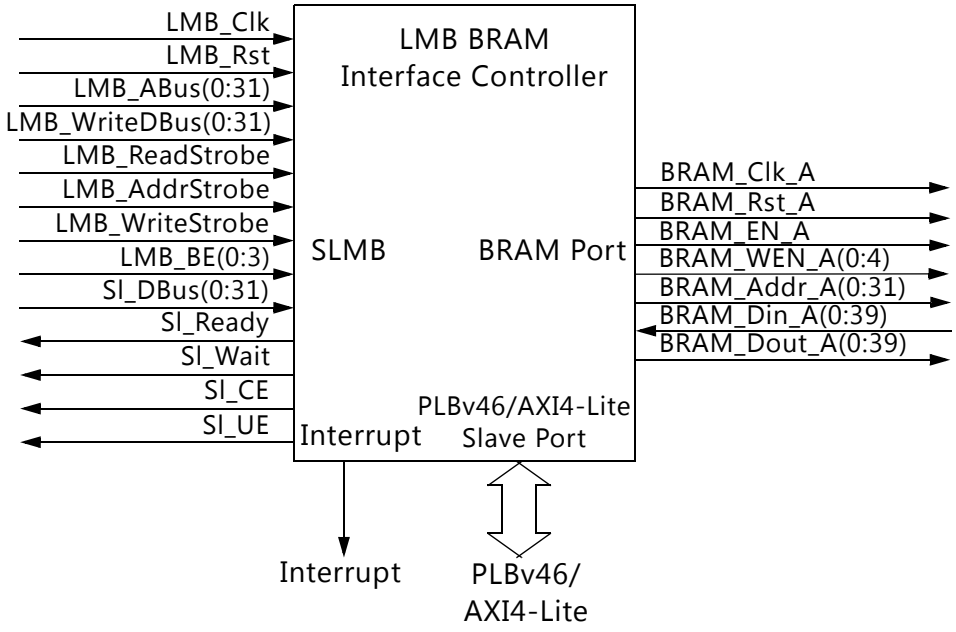

Figure 1-1: **LMB BRAM Interface Controller Block Diagram**

Feature Summary

Provides a low area, high frequency and low latency connection for the MicroBlaze DLMB and ILMB ports to FPGA block RAM. The supported block RAM sizes are 2-128 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 signalling. The optional ECC registers are connected to MicroBlaze through either an AXI4-Lite or PLBv46 interface. The PLBv46 bus is only available in EDK.

The LMB BRAM Interface Controller 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.

Licensing and Ordering Information

This Xilinx LogiCORE IP module is provided at no additional cost with the Xilinx Vivado Design Suite and ISE Design Suite Embedded Edition tools 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 on 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 Interface standard* [Ref 2].

The LMB BRAM Interface Controller implements a Processor Local Bus slave interface [Ref 3].

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

Table 2-1 lists clock frequencies for the target families. The maximum achievable clock frequency can vary. The maximum achievable clock frequency and all resource counts can be affected by the tool flow, other tool options, additional logic in the FPGA, different versions of the Xilinx tools, and other factors.

Table 2-1: **Maximum Frequencies**

Architecture	Speed grade	Max Frequency
Spartan®-6	-4	195
Virtex®-6	-3	300
Artix™-7	-3	225
Kintex™-7	-3	320
Virtex-7	-3	320

The configuration for reaching these numbers has a single LMB master interface and does not include ECC and interfaces to a single block RAM.

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 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

Because the LMB BRAM Interface Controller is a module that is used with other design elements in the FPGA, the utilization and timing numbers reported in Table 2-2 are estimates. The actual utilization of FPGA resources and timing of the LMB BRAM Interface Controller design can vary from the results reported here.

Table 2-2: **Performance and Resource Utilization on Virtex-6 (xc6vlx240t-ff1156-1)**

Parameter Values (other parameters have default values)									Device Resources	
C_ECC	C_INTERCONNECT	C_FAULT_INJECT	C_CE_FAILING_REGISTERS	C_UE_FAILING_REGISTERS	C_ECC_STATUS_REGISTERS	C_ECC_ONOFF_REGISTER	C_CE_COUNTER_WIDTH	C_WRITE_ACCESS	Flip-Flops	LUTs
0	0	0	0	0	0	0	0	0	2	6
1	0	0	0	0	0	0	0	0	2	109
1	0	0	0	0	0	0	0	1	2	129
1	0	0	0	0	0	0	0	2	106	196
1	2	0	0	0	1	0	0	0	50	139
1	1	0	0	0	1	0	0	0	51	129
1	1	0	0	0	1	1	0	0	52	130
1	1	1	0	0	1	0	0	2	225	290
1	1	0	1	0	1	0	0	0	152	162
1	1	0	0	1	1	0	0	0	152	232
1	1	0	0	0	1	0	10	0	69	154
1	1	1	1	1	1	1	10	2	378	465

Port Descriptions

The I/O ports and signals for the LMB BRAM Interface Controller are listed and described in Table 2-3.

Table 2-3: **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
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

Table 2-3: **LMB BRAM Interface Controller I/O Signals *(Cont'd)***

Port Name	MSB:LSB	I/O	Description
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_LMB_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
Misc. Signals			
Interrupt		O	Interrupt
UE		O	One cycle pulse signalling an ECC Uncorrectable Data Error
CE		O	One cycle pulse signalling an ECC Correctable Data Error
PLB Interface Signals			
SPLB_CTRL_PLB_ABus	0:31	I	PLB address bus
SPLB_CTRL_PLB_PAValid		I	PLB primary address valid

Table 2-3: **LMB BRAM Interface Controller I/O Signals *(Cont'd)***

Port Name	MSB:LSB	I/O	Description
SPLB_CTRL_PLB_masterID	0:C_SPLB_CTRL_MID_WIDTH-1	I	PLB current master identifier
SPLB_CTRL_PLB_RNW		I	PLB read not write
SPLB_CTRL_PLB_BE	0:C_SPLB_CTRL_DWIDTH/8-1	I	PLB byte enables
SPLB_CTRL_PLB_size	0:3	I	PLB size of requested transfer
SPLB_CTRL_PLB_type	0:2	I	PLB transfer type
SPLB_CTRL_PLB_wrDBus	0:C_SPLB_CTRL_DWIDTH-1	I	PLB write data bus
Unused PLB Interface Signals			
SPLB_CTRL_PLB_UABus	0:31	I	PLB upper address bits
SPLB_CTRL_PLB_SAValid		I	PLB secondary address valid
SPLB_CTRL_PLB_rdPrim		I	PLB secondary to primary read request indicator
SPLB_CTRL_PLB_wrPrim		I	PLB secondary to primary write request indicator
SPLB_CTRL_PLB_abort		I	PLB abort bus request
SPLB_CTRL_PLB_busLock		I	PLB bus lock
SPLB_CTRL_PLB_MSize	0:1	I	PLB data bus width indicator
SPLB_CTRL_PLB_lockErr		I	PLB lock error
SPLB_CTRL_PLB_wrBurst		I	PLB burst write transfer
SPLB_CTRL_PLB_rdBurst		I	PLB burst read transfer
SPLB_CTRL_PLB_ wrPendReq		I	PLB pending bus write request
SPLB_CTRL_PLB_ rdPendReq		I	PLB pending bus read request
SPLB_CTRL_PLB_ wrPendPri	0:1	I	PLB pending write request priority
SPLB_CTRL_PLB_ rdPendPri	0:1	I	PLB pending read request priority
SPLB_CTRL_PLB_reqPri	0:1	I	PLB current request priority
SPLB_CTRL_PLB_ TAttribute	0:15	I	PLB transfer attribute
PLB Slave Interface Signals			
SPLB_CTRL_Sl_addrAck		O	Slave address acknowledge
SPLB_CTRL_Sl_SSize	0:1	O	Slave data bus size
SPLB_CTRL_Sl_wait		O	Slave wait
SPLB_CTRL_Sl_rearbitrate		O	Slave bus rearbitrate
SPLB_CTRL_Sl_wrDAck		O	Slave write data acknowledge
SPLB_CTRL_Sl_wrComp		O	Slave write transfer complete

Table 2-3: **LMB BRAM Interface Controller I/O Signals *(Cont'd)***

Port Name	MSB:LSB	I/O	Description
SPLB_CTRL_Sl_rdDBus	0: C_SPLB_CTRL_DWIDTH-1	O	Slave read data bus
SPLB_CTRL_Sl_rdDAck		O	Slave read data acknowledge
SPLB_CTRL_Sl_rdComp		O	Slave read transfer complete
SPLB_CTRL_Sl_MBusy	0: C_SPLB_CTRL_NUM_MASTERS-1	O	Slave busy
SPLB_CTRL_Sl_MWrErr	0: C_SPLB_CTRL_NUM_MASTERS-1	O	Slave write error
SPLB_CTRL_Sl_MRdErr	0: C_SPLB_CTRL_NUM_MASTERS-1	O	Slave read error
Unused PLB Slave Interface Signals			
SPLB_CTRL_Sl_wrBTerm		O	Slave terminate write burst transfer
SPLB_CTRL_Sl_rdWdAddr	0:3	O	Slave read word address
SPLB_CTRL_Sl_rdBTerm		O	Slave terminate read burst transfer
SPLB_CTRL_Sl_MIRQ	0: C_SPLB_CTRL_NUM_MASTERS-1	O	Master interrupt request
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.

Table 2-3: **LMB BRAM Interface Controller I/O Signals *(Cont'd)***

Port Name	MSB:LSB	I/O	Description
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.
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 acknowledgement 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 2-4 shows the Register Address Map for the LMB BRAM Interface Controller. The individual registers are described in Table 2-5 to Table 2-28.

Table 2-4: **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
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 2-5: **ECC Status Register (ECC_STATUS)**

Reserved		ECC_STATUS	
0	29	30	31

Table 2-6: **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 2-7: **ECC Interrupt Enable Register (ECC_EN_IRQ)**

Reserved		ECC_EN_IRQ	
0	29	30	31

Table 2-8: **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 2-9: **ECC On/Off Register (ECC_ONOFF)**

Reserved		ECC_ONOFF
0	30	31

Table 2-10: **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 2-11: **Correctable Error Counter Register (CE_CNT)**

Reserved		CE_CNT	
0	31-C_CE_COUNTER_WIDTH	32-C_CE_COUNTER_WIDTH	31

Table 2-12: **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 2-13: **Correctable Error First Failing Data Register (CE_FFD)**

CE_FFD	
0	31

Table 2-14: **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 2-15: **Correctable Error First Failing ECC Register (CE_FFE)**

Reserved		CE_FFE	
0	24	25	31

Table 2-16: **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 2-17: **Correctable Error First Failing Address Register (CE_FFA)**

CE_FFA	
0	31

Table 2-18: **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 2-19: **Uncorrectable Error First Failing Data Register (UE_FFD)**

UE_FFD	
0	31

Table 2-20: **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 2-21: **Uncorrectable Error First Failing ECC Register (UE_FFE)**

Reserved		UE_FFE	
0	24	25	31

Table 2-22: **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 2-23: **Uncorrectable Error First Failing Address Register (UE_FFA)**

UE_FFA	
0	31

Table 2-24: **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 2-25: **Fault Injection Data Register (FI_D)**

FI_D	
0	31

Table 2-26: **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 2-27: **Fault Injection ECC Register (FI_ECC)**

Reserved		FI_ECC	
0	24	25	31

Table 2-28: **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.

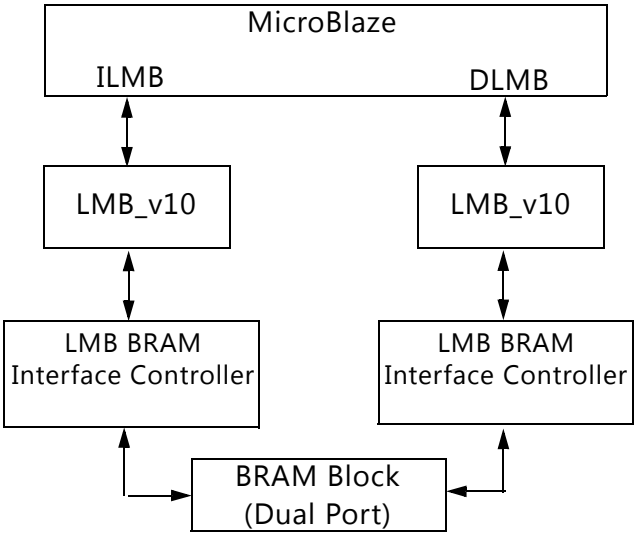

Figure 3-1: **Typical MicroBlaze System**

The Interrupt output and the PLBv46 or 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.

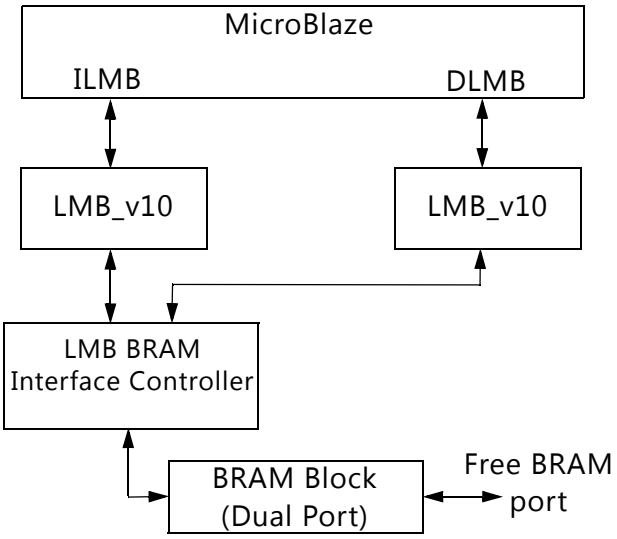

Figure 3-2: **MicroBlaze System with Multiplexed ILMB and DLMB**

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* [Ref 4], 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 and which FPGA family is used. The overhead is listed in Table 3-2.

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

Block RAM Data Size	ECC Overhead for Spartan-3, Spartan-6 and Virtex-4 Families	All Other Families
2 kB	100%	Not Applicable(1)
4 kB	50%	100%
8 kB	25%	50%
16 kB and larger	25%	25%

1. Minimum size is 4K

A set of optional registers in the LMB BRAM Interface Controller controls the operation of the ECC logic. The registers are accessed through either a PLBv46 or an AXI4-Lite slave interfaces. The slave interfaces are connected to MicroBlaze DPLB or M_AXI_DP ports in a typical system, according to Figure 3-3.

The LMB BRAM Interface Controller requires that the PLBv46/AXI4-Lite bus is synchronous to `LMB_Clk`.

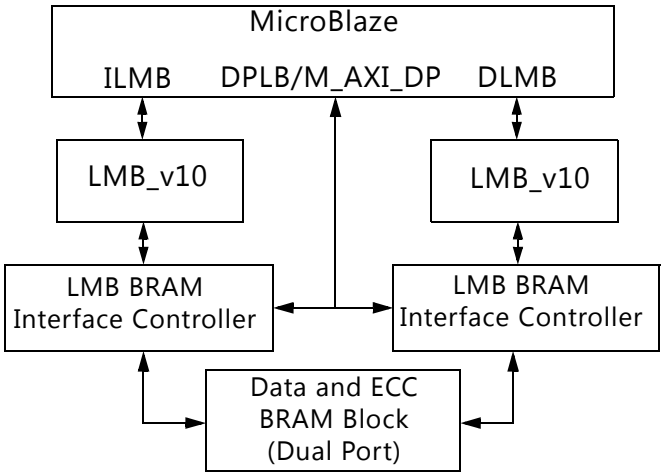

Figure 3-3: **Typical MicroBlaze System Using ECC**

ECC Initialization

The ECC bits are normally initialized by Data2MEM (See *Data2MEM User Guide* [Ref 6]). 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 and 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* [Ref 1].

SECTION II: VIVADO DESIGN SUITE

Customizing and Generating the Core

Constraining the Core

Customizing and Generating the Core

This chapter includes information on using Xilinx tools to customize and generate the core using the Vivado™ Design Suite.

GUI

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.

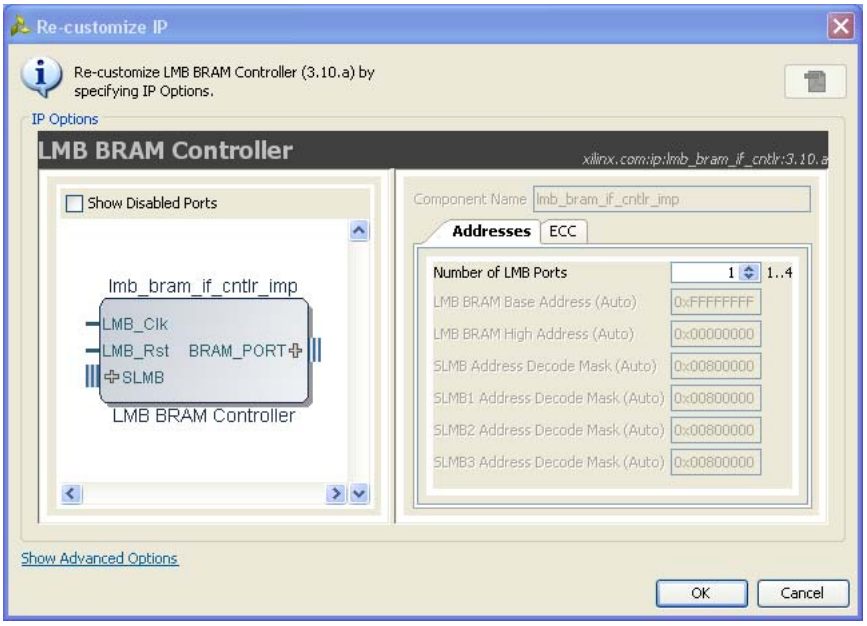

Figure 4-1: **Addresses Parameter Tab**

- **Number of LMB Ports** - Sets the number of ports available to connect to MicroBlaze™.
- **LMB BRAM Base Address** - Base address of the local memory.
- **LMB BRAM High Address** - High address of the local memory.

- **SLMB/SLMB1/SLMB2/SLM3 Address Decode Mask** - A mask indicating which address bits the LMB BRAM Interface Controller takes into account when decoding an access.

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

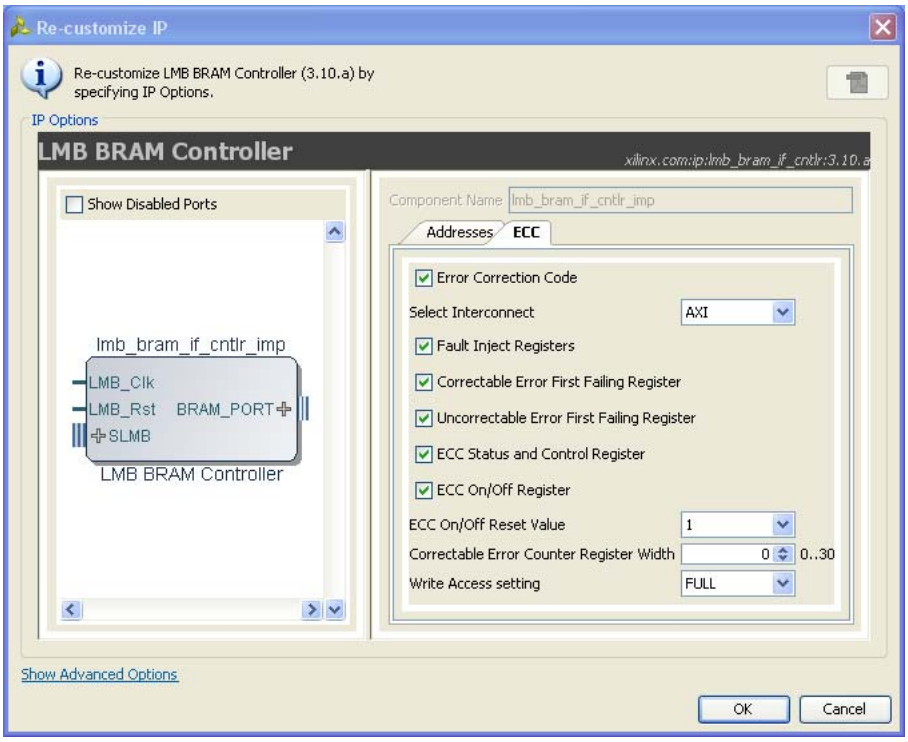

Figure 4-2: **ECC Parameter Tab**

- **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)	0x00800000	std_logic_vector
C_MASK1	LMB Decode Mask	Valid decode mask for SLMB1(3)	0x00800000	std_logic_vector
C_MASK2	LMB Decode Mask	Valid decode mask for SLMB2(3)	0x00800000	std_logic_vector
C_MASK3	LMB Decode Mask	Valid decode mask for SLMB3(3)	0x00800000	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 BASEADDR and 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 may be either of PLBv46 or AXI4-Lite type. The parameters related to PLBv46 and 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 *IP Processor Block RAM (BRAM) Block (v1.00a) data sheet* [Ref 5].

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* [Ref 1] for details on the transaction signaling.

Output Generation

The following files are generated by the IP in Vivado IP Integrator.

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

Constraining the Core

Required Constraints

There are no required constraints for this core.

Device, Package, and Speed Grade Selections

There are no Device, Package or Speed Grade requirements for this core.

Clock Frequencies

There are no specific clock frequency requirements for this 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

There are no specific Clock placement requirements for this core.

Banking

There are no specific Banking rules for this core.

Transceiver Placement

There are no Transceiver Placement requirements for this core.

I/O Standard and Placement

There are no specific I/O standards and placement requirements for this core.

SECTION III: ISE DESIGN SUITE

Customizing and Generating the Core

Constraining the Core

Customizing and Generating the Core

This chapter includes information on using Xilinx tools to customize and generate the core using the ISE® Design Suite.

GUI

The LMB BRAM Interface Controller parameters are divided in five categories: Addresses, LMB, ECC, SPLB_CTRL and S_AXI_CTRL. The Addresses parameter category is shown in Figure 6-1.

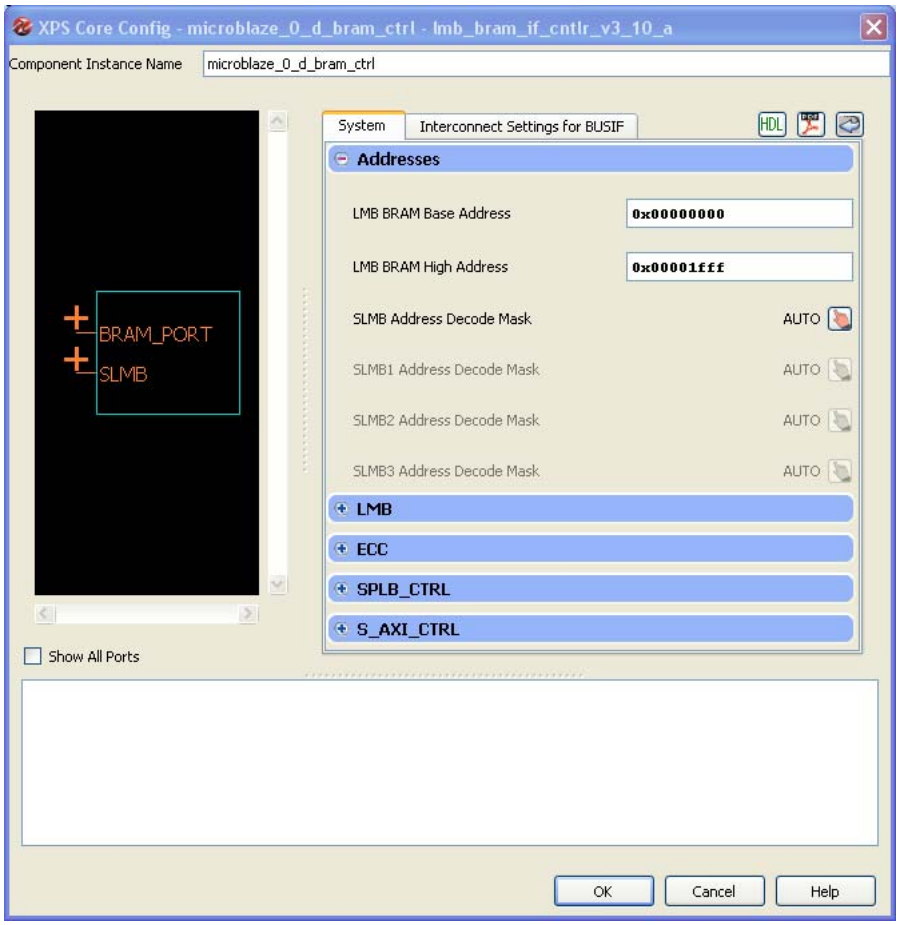

Figure 6-1: **Addresses Parameter Category**

- **LMB BRAM Base Address** - Base address of the local memory.
- **LMB BRAM High Address** - High address of the local memory.
- **SLMB/SLMB1/SLMB2/SLM3 Address Decode Mask** - A mask indicating which address bits the LMB BRAM Interface Controller takes into account when decoding an access.

The LMB parameter category is shown in Figure 6-2.

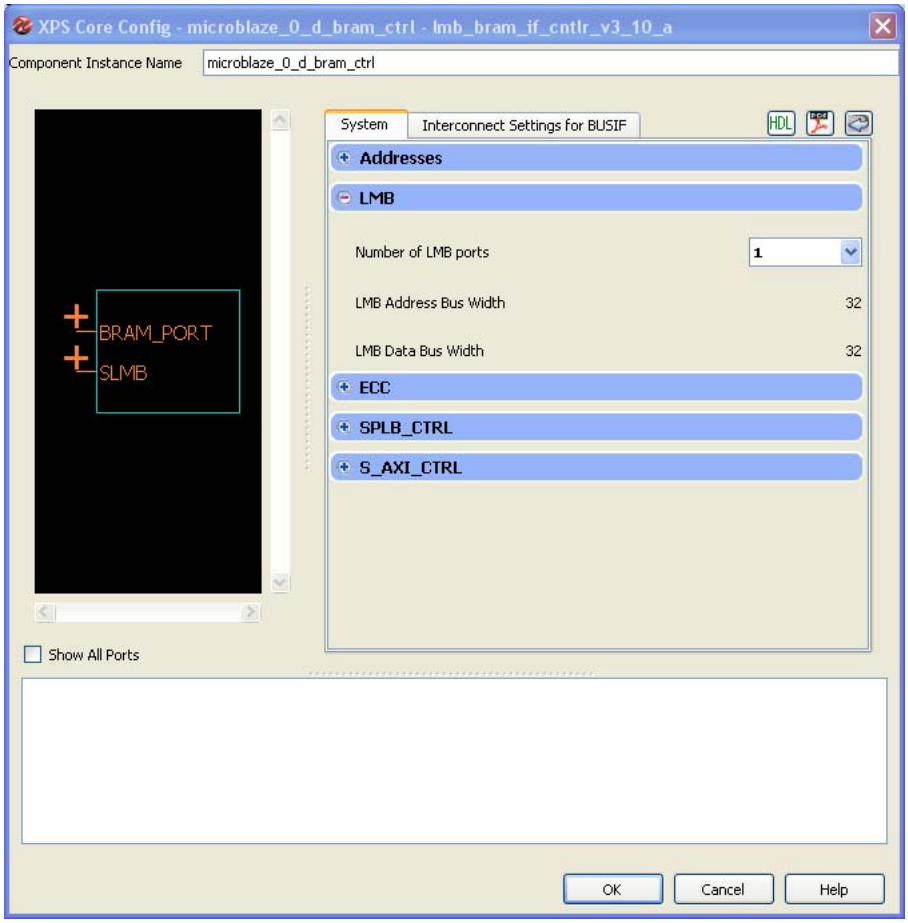

Figure 6-2: **LMB Parameter Category**

- **Number of LMB Ports** - Sets the number of ports available to connect to MicroBlaze™.
- **LMB Address Bus Width** - The number of bits in the address bus. Fixed to 32.
- **LMB Data Bus Width** - The number of bits in the data bus. Fixed to 32.

The ECC parameter category is shown in Figure 6-3.

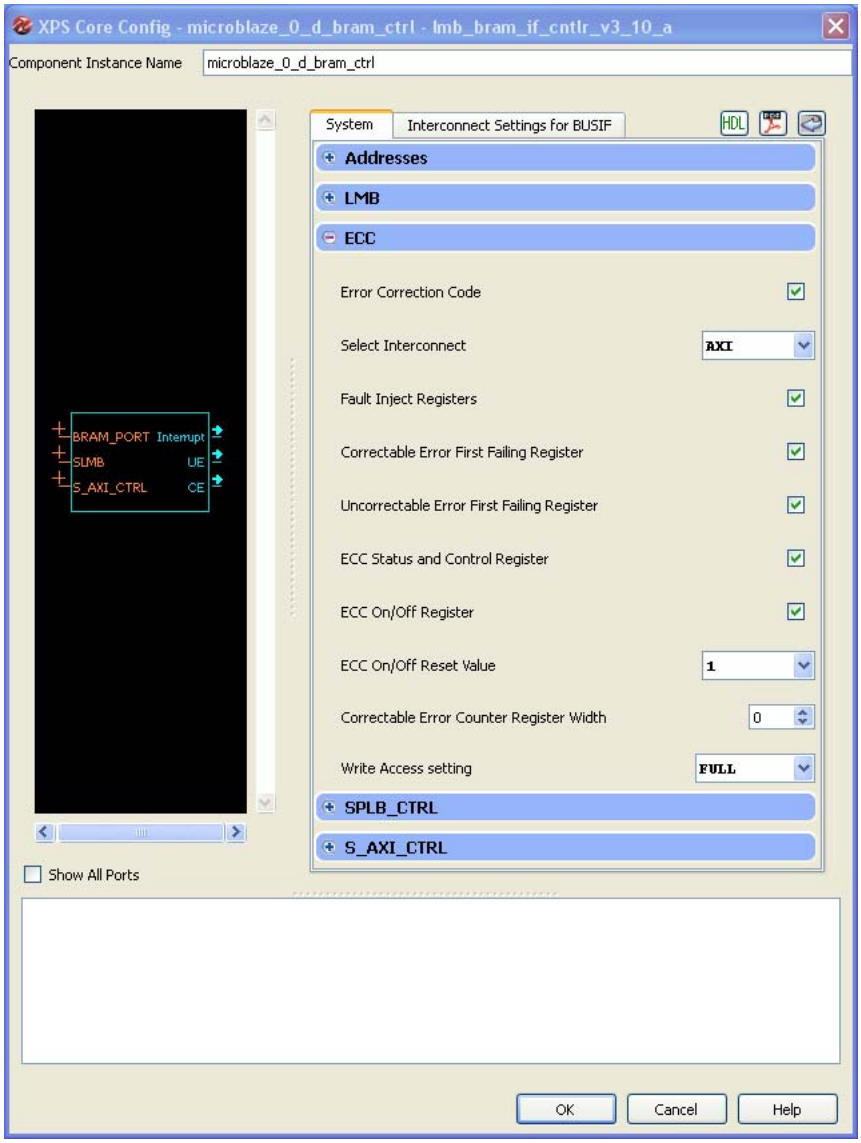

Figure 6-3: **ECC Parameter Category**

- **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.

The SPLB_CTRL and S_AXI_CTRL categories allow setting bus interface specific parameters, in particular the base address and high address of the ECC registers.

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 specific features that can be parameterized in Xilinx LMB BRAM Interface Controller EDK designs are shown in Table 6-1. See SECTION II: VIVADO DESIGN SUITE, Chapter 4, Customizing and Generating the Core for additional parameters.

Table 6-1: **EDK LMB BRAM Interface Controller Parameters**

Parameter Name	Feature/Description	Allowable Values	Default Value	VHDL Type
Basic Parameters				
C_LMB_AWIDTH	LMB Address Bus Width	32	32	integer
C_LMB_DWIDTH	LMB Data Bus Width	32	32	integer
PLB Parameters				
C_SPLB_CTRL_BASEADDR(3)	PLB Base Address	Valid Address(2)	None(1)	std_logic_vector
C_SPLB_CTRL_HIGHADDR(3)	PLB High Address	Valid Address(2)	None(1)	std_logic_vector
C_SPLB_CTRL_AWIDTH(3)	PLB least significant address bus width	32	32	integer
C_SPLB_CTRL_DWIDTH(3)	PLB data width	32, 64, 128	32	integer
C_SPLB_CTRL_P2P(3)	Selects point-to-point or shared bus topology	0 = Shared Bus Topology 1 = Point-to-Point Bus Topology	0	integer

Table 6-1: **EDK LMB BRAM Interface Controller Parameters *(Cont'd)***

Parameter Name	Feature/Description	Allowable Values	Default Value	VHDL Type
C_SPLB_CTRL_MID_WIDTH[3]	PLB Master ID Bus Width	$\log_2$(C_SPLB_CTRL_NUM_MASTERS) with minimum value of 1	1	integer
C_SPLB_CTRL_NUM_MASTERS[3]	Number of PLB Masters	1 - 16	1	integer
C_SPLB_CTRL_SUPPORT_BURSTS[3]	Support Bursts	0	0	integer
C_SPLB_CTRL_NATIVE_DWIDTH[3]	Width of the Slave Data Bus	32	32	integer
AXI Parameters				
C_S_AXI_CTRL_BASEADDR[4]	AXI Base Address	Valid Address[2]	None[1]	std_logic_vector
C_S_AXI_CTRL_HIGHADDR[4]	AXI High Address	Valid Address[2]	None[1]	std_logic_vector
C_S_AXI_CTRL_ADDR_WIDTH[4]	AXI address bus width	32	32	integer
C_S_AXI_CTRL_DATA_WIDTH[4]	AXI data bus width	32	32	integer
C_S_AXI_CTRL_PROTOCOL[4]	AXI interface type	AXI4LITE	AXI4LITE	string

Notes:

1. No default value is specified for BASEADDR and 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. Parameter value is don't care unless parameter C_INTERCONNECT = 1 (PLBv46)
4. Parameter value is don't care unless parameter C_INTERCONNECT = 2 (AXI4-Lite)

Parameter - Port Dependencies

The width of many of the 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 6-2.

Table 6-2: **Parameter-Port Dependencies**

Parameter Name	Ports (Port Width Depends on Parameter)
C_LMB_AWIDTH	LMB_ABus
C_LMB_DWIDTH	LMB_BE, LMB_WriteDBus, Sl_DBus, BRAM_WEN_A, BRAM_Din_A, BRAM_Dout_A
C_SPLB_CTRL_MID_WIDTH	SPLB_CTRL_PLB_masterID
C_SPLB_CTRL_DWIDTH	SPLB_CTRL_PLB_BE, SPLB_CTRL_PLB_wrDBus, SPLB_CTRL_Sl_rdDBus
C_SPLB_CTRL_NUM_MASTERS	SPLB_CTRL_Sl_MBusy, SPLB_CTRL_Sl_MWrErr, SPLB_CTRL_Sl_MRdErr, SPLB_CTRL_Sl_MIRQ
C_ECC	BRAM_WEN_A, BRAM_Din_A, BRAM_Dout_A

Programming Model

Supported Memory Sizes

For supported block RAM memory sizes, see IP Processor Block RAM (BRAM) Block (v1.00a) [Ref 5].

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 high address and the base address must be equal to 2^n bytes, where *n* is a positive integer and 2^n is a valid memory size as shown in Table 6-3. In addition, the *n* least significant bits of C_BASEADDR must be equal to 0.

Table 6-3: **Example Address Range Specifications**

Memory Size (Bytes)	C_BASEADDR	C_HIGHADDR
8 K	0x24000000	0x24001FFF
16 K	0xE0000000	0xE0003FFF
32 K	0x3FF00000	0x3FF07FFF
64 K	0x82000000	0x8200FFFF
128 K	0xB0000000	0xB001FFFF
256 K	0xC0000000	0xC003FFFF

LMB Timing

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

Constraining the Core

See SECTION II: VIVADO DESIGN SUITE,Chapter 5, Constraining the Core.

SECTION IV: APPENDICES

Migrating

Debugging

Application Software Development

Additional Resources

Appendix A

Migrating

This appendix describes migrating from older versions of the IP to the current IP release.

For information on migrating to the Vivado™ Design Suite, see the *Vivado Design Suite Migration Methodology Guide* [Ref 7].

Debugging

This appendix includes details about resources available on the Xilinx Support website and debugging tools. In addition, this appendix provides a step-by-step debugging process and a flow diagram to guide you through debugging the LMB BRAM Interface Controller core.

The following topics are included in this appendix:

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 (www.xilinx.com/support) contains key resources such as product documentation, release notes, answer records, information about known issues, and links for opening a Technical Support WebCase.

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 (www.xilinx.com/support) or by using the Xilinx Documentation Navigator.

Download the Xilinx Documentation Navigator from the Design Tools tab on the Downloads page (www.xilinx.com/download). For more information about this tool and the features available, open the online help after installation.

Release Notes

Known issues for all cores, including the LMB BRAM Interface Controller are described in the IP Release Notes Guide (XTP025).

Contacting Technical Support

Xilinx provides premier technical support for customers encountering issues that require additional assistance.

To contact Xilinx Technical Support:

1. Navigate to www.xilinx.com/support.
2. Open a WebCase by selecting the WebCase link located under Support Quick Links.

When opening a WebCase, include:

- Target FPGA including package and speed grade.
- All applicable Xilinx Design Tools and simulator software versions.
- Additional files based on the specific issue might also be required. See the relevant sections in this debug guide for guidelines about which file(s) to include with the WebCase.

Xilinx provides technical support at www.xilinx.com/support for this LogiCORE™ IP product when used as described in the product documentation. Xilinx cannot guarantee timing, functionality, or support of product if implemented in devices that are not defined in the documentation, if customized beyond that allowed in the product documentation, or if changes are made to any section of the design labeled DO NOT MODIFY.

Debug Tools

The main tool available to address LMB BRAM Interface Controller design issues is the ChipScope™ Pro tool.

ChipScope Pro Tool

The ChipScope Pro debugging tool inserts logic analyzer, bus analyzer, and virtual I/O cores directly into your design. The ChipScope Pro debugging tool allows you to set trigger conditions to capture application and integrated block port signals in hardware. Captured signals can then be analyzed through the ChipScope Pro logic analyzer tool. For detailed information for using the ChipScope Pro debugging tool, see www.xilinx.com/tools/cspro.htm.

Reference Boards

All 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 ModelSim is described below. A similar approach can be used with other simulators.

- Check for the latest supported versions of ModelSim 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 Xilinx Platform Studio this is done within the tool using **Edit > Preferences > Simulation**, and in the Vivado Design Suite using **Flow > Simulation Settings.**
- Have you associated the intended software program for the MicroBlaze™ processor with the simulation? Use **Project > Select Elf File** in Xilinx Platform Studio to do this. Make sure to regenerate the simulation files with **Simulation > Generate Simulation HDL Files** afterwards. The equivalent command in the Vivado Design Suite is **Tools > Associate ELF Files**.
- When observing the traffic on the LMB interface connected to the LMB BRAM I/F Controller, see the *MicroBlaze Processor Reference Guide* [Ref 1] for the LMB timing.

Hardware Debug

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

Many of these common issues can also be applied to debugging design simulations. Details are provided on:

- General Checks
- LMB Checks

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 ChipScope, When Error Correction Codes are used, the signals `Sl_Wait`, `Sl_CE`, and `Sl_UE` can also be added.

To sample the interface signals, ChipScope should use the `LMB_Clk` clock signal.

Interface Debug

AXI4-Lite Interface

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 ChipScope debugging tool capture that the waveform is correct for accessing the AXI4-Lite interface.

PLBv46 Interface

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

- The interface is not being held in reset, and `LMB_Rst` is an active-High 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 ChipScope debugging tool capture that the waveform is correct for accessing the PLBv46 interface.

XILINX®

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.

Appendix D

Additional Resources

Xilinx Resources

For support resources such as Answers, Documentation, Downloads, and Forums, see the Xilinx Support website at:

www.xilinx.com/support.

For a glossary of technical terms used in Xilinx documentation, see:

www.xilinx.com/company/terms.htm.

References

These documents provide supplemental material useful with this user guide:

1. MicroBlaze Processor Reference Guide (PG081)
2. ARM® AMBA AXI4-Lite Protocol Specification, Version 2.0 ARM IHI 0022D
3. IBM 128-Bit Processor Local Bus Architectural Specification (v4.6)
4. Xilinx Application Note, Single Error Correction and Double Error Detection (XAPP645)
5. IP Processor Block RAM (BRAM) Block (v1.00a) (DS444)
6. Data2MEM User Guide (UG658)
7. Vivado™ Design Suite Migration Methodology Guide (UG911)
8. Vivado™ Design Suite user documentation

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
07/25/12	1.0	Initial Xilinx release. This Product Guide is derived from DS452.
10/16/12	1.1	Xilinx release 14.3/2012.3. Updated core version.
12/18/12	1.2	Xilinx release 14.4/2012.4. Updated core version. Debug chapter updated.

Notice of Disclaimer