IP Processor LMB BRAM Interface Controller (v3.00.b)

DS452 June 22, 2011 **Product Specification**

Introduction

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

The LMB BRAM Interface Controller connects to an lmb_v10 bus.

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

Features

- LMB v1.0 bus interfaces with byte enable support
- Used in conjunction with bram_block peripheral to provide fast BRAM memory solution for MicroBlaze™ ILMB and DLMB ports.
- Supports byte, half-word, and word transfers
- Supports optional BRAM error correction and detection.

LogiCORE™ Facts				
Core Specifics				
Supported Device Family	Spartan®-3, Spartan-3E, Spartan-6, Spartan-3A/3A DSP/3AN, Automotive Spartan-3/3A/3A DSP/3E, Automotive Spartan-6, Defence Grade Spartan-6 Q, Virtex®-4, Defence Grade Virtex-4 Q, Space-Grade Virtex-4 QV, Virtex-5, Space-Grade Virtex-5 Q, Virtex-6, Defence Grade Virtex-6 Q			
Resources Used	Slices	LUTs	FFs	Block RAMs
	N/A	6	2	0
Provided with Core				
Documentation	Product Specification			
Design File Formats	VHDL			
Constraints File	N/A			
Verification	N/A			
Instantiation Template	N/A			
Design Tool Requirements				
Xilinx Implementation Tools	ISE® 13.2			
Verification	Mentor Graphics ModelSim: v6.6a and above			
Simulation	Mentor Graphics ModelSim: v6.6a and above			
Synthesis	ISE® 13.2			
Support				
Provided by Xilinx, Inc.				

Functional Description

The LMB BRAM Interface Controller is the interface between the LMB and the bram_block peripheral. A BRAM memory subsystem consists of the controller along with the bram_block peripheral.

The input/output signals of the LMB BRAM interface controller are shown in Figure 2. The detailed list of signals are listed and described in Table 3. See the description of LMB Signals in the MicroBlaze Bus Interfaces chapter in the *MicroBlaze Processor Reference Guide*.

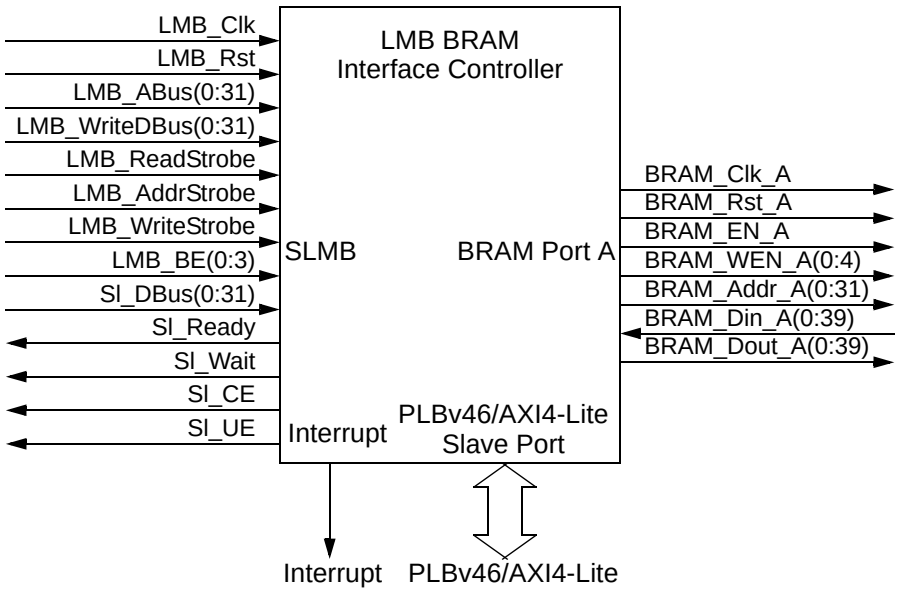

Figure 1: **LMB BRAM Interface Controller Block Diagram**

In a MicroBlaze system, without ECC protection, the LMB BRAM Interface controller would typically be connected according to Figure 2.

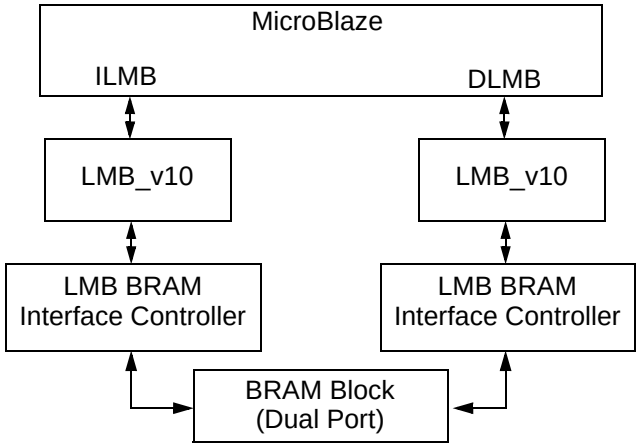

Figure 2: **Typical MicroBlaze System**

The Interrupt output and the PLBv46 or AXI4-Lite interfaces would be unconnected and the BRAM_DIn_A and BRAM_DOut_A signals would only contain 32 data bits.

LMB Controller With ECC

To mitigate the effect of BRAM Single Event Upsets, SEUs, the LMB BRAM Controller can be configured to use Error Correction Codes, ECC. When writing to the BRAM, ECC bits are generated and stored together with the written data. When reading from the BRAM 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 via the LMB to MicroBlaze or by an interrupt signal. The ECC used is a (32,7) Hamming code with the coding defined by Table 1

Table 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 XAPP645, but is shown here in its optimized form.

The need to store the ECC increases the BRAM utilization depending on BRAM data size and which FPGA family is used. The overhead is give in Table 2

Table 2: **ECC BRAM Overhead**

BRAM Data Size	ECC overhead for Spartan-3, Spartan-6 and Virtex-4 families	All other families
2kB	100%	Not Applicable(1)
4kB	50%	100%
8kB	25%	50%
16kB and larger	25%	25%

1. Minimum size is 4kB

A set of optional registers in the LMB BRAM 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.

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

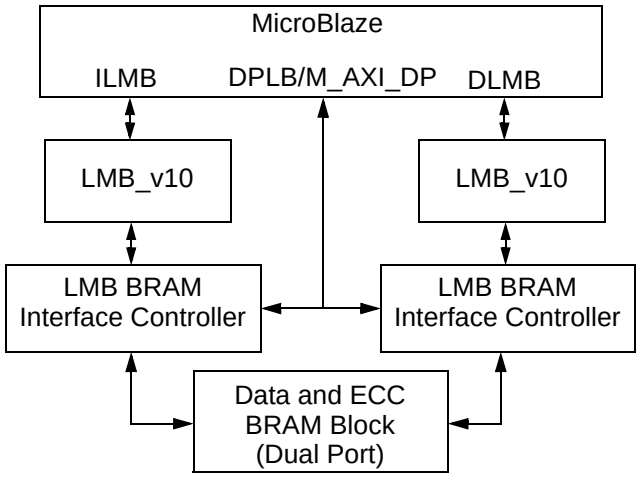

Figure 3: **Typical MicroBlaze system using ECC**

ECC Initialization

The ECC bits are normally initialized by Data2MEM. However, they can also be initialized by software. The initialization is performed by reading and writing back the whole contents of the BRAM 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, which 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 Sl_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 of the ECC function.

Small system with Correctable Error First Failing registers and Status register added. A single bit error will latch the address for the access into the Correctable Error First Failing Address Register and set the CE_STATUS bit in the ECC Status Register. An interrupt will be generated triggering MicroBlaze to read the failing address and then perform a read followed by a write on the failing address. This will remove the single bit error from the BRAM, 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.

Typical system with Uncorrectable Error First Failing registers and Fault Injection registers added. 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.

LMB BRAM Interface Controller I/O Signals

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

Table 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

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

Port Name	MSB:LSB	I/O	Description
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		0	LMB Wait
Sl_CE		O	LMB Correctable Error
Sl_UE		O	LMB Uncorrectable Error
BRAM Interface Signals (Data and ECC)			
BRAM_Rst_A		O	BRAM Reset
BRAM_Clk_A		O	BRAM Clock
BRAM_EN_A		O	BRAM Enable
BRAM_WEN_A	0:(C_LMB_DWIDTH+8*C_ECC)/8-1	O	BRAM Write Enable
BRAM_Addr_A	0:C_LMB_AWIDTH-1	O	BRAM Address
BRAM_Din_A	0:C_LMB_DWIDTH+8*C_ECC-1	I	BRAM Data Input
BRAM_Dout_A	0:C_LMB_DWIDTH+8*C_ECC-1	O	BRAM Data Output
Misc. Signals			
Interrupt		O	Interrupt
PLB Interface Signals			
SPLB_CTRL_PLB_ABus	0:31	I	PLB address bus
SPLB_CTRL_PLB_ PAValid		I	PLB primary address valid
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

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

Port Name	MSB:LSB	I/O	Description
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
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

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

Port Name	MSB:LSB	I/O	Description
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-	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		Write data
S_AXI_CTRL_WSTB	C_S_AXI_CTRL_DATA_WIDTH/8-1:0		Write strobes. This signal indicates which byte lanes to update in memory(0).
S_AXI_CTRL_WVALID			Write valid. This signal indicates that valid write data and strobes are available.
S_AXI_CTRL_WREADY			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			

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

Port Name	MSB:LSB	I/O	Description
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 will remain 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

LMB BRAM Interface Controller Parameters

To allow the user 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 the user to configure a design that only utilizes 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.

Table 4: 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(3)	0x00800000	std_logic_vector
C_LMB_AWIDTH	LMB Address Bus Width	32	32	integer
C_LMB_DWIDTH	LMB Data Bus Width	32	32	integer
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 1=PLBv46 2=AXI4-Lite	1	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
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

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

Parameter Name	Feature/Description	Allowable Values	Default Value	VHDL Type
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
PLB Parameters				
C_SPLB_CTRL_BASEADDR(5)	PLB Base Address	Valid Address(2)	None(1)	std_logic_vector
C_SPLB_CTRL_HIGHADDR(5)	PLB High Address	Valid Address(2)	None(1)	std_logic_vector
C_SPLB_CTRL_AWIDTH(5)	PLB least significant address bus width	32	32	integer
C_SPLB_CTRL_DWIDTH(5)	PLB data width	32, 64, 128	32	integer
C_SPLB_CTRL_P2P(5)	Selects point-to-point or shared bus topology	0 = Shared Bus Topology 1 = Point-to-Point Bus Topology(4)	0	integer
C_SPLB_CTRL_MID_WIDTH(5)	PLB Master ID Bus Width	$\log_2$(C_SPLB_CTRL_NUM_MASTERS) with minimum value of 1	1	integer
C_SPLB_CTRL_NUM_MASTERS(5)	Number of PLB Masters	1 - 16	1	integer

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

Parameter Name	Feature/Description	Allowable Values	Default Value	VHDL Type
C_SPLB_CTRL_SUPPORT_BURSTS(5)	Support Bursts	0	0	integer
C_SPLB_CTRL_NATIVE_DWIDTH(5)	Width of the Slave Data Bus	32	32	integer
AXI Parameters				
C_S_AXI_CTRL_BASEADDR(6)	AXI Base Address	Valid Address(2)	None(1)	std_logic_vector
C_S_AXI_CTRL_HIGHADDR(6)	AXI High Address	Valid Address(2)	None(1)	std_logic_vector
C_S_AXI_CTRL_ADDR_WIDTH(6)	AXI address bus width	32	32	integer
C_S_AXI_CTRL_DATA_WIDTH(6)	AXI data bus width	32	32	integer
C_S_AXI_CTRL_PROTOCOL(6)	AXI interface type	AXI4LITE	AXI4LITE	string

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

C_MASK

If using the Embedded Development Kit, this parameter is automatically set by the LMB BRAM Interface Controller when running the Platform Generator tool and users do not need to set the value. The address mask indicates which bits are used in the LMB decode to decode that a valid address is present on the LMB. Any bits that are set to '1' in the mask indicate that the address bit in that position is used to decode a valid LMB access. All other address bits are considered don't care for the purpose of decoding LMB accesses. The LMB BRAM Interface Controller may limit the user's choice for the address mask: the most restrictive case is that only a single bit may be set in the mask. Consult the platform generation tool informational messages for details.

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 not only enables the ECC Status Register and the ECC Interrupt Enable Register, but also the generation of the external Interrupt signal.

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 BRAM design parameters and I/O signals are shown in Table 5.

Table 5: 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

LMB BRAM Interface Controller Register Descriptions

Table 6: LMB BRAM Interface Register Address Map

Base Address + Offset (hex)	Register	Access Type	Description
C_BASEADDR + 0x0	ECC_STATUS	R/W	ECC Status Register
C_BASEADDR + 0x4	ECC_EN_IRQ	R/W	ECC Enable Interrupt Register
C_BASEADDR + 0x8	ECC_ONOFF	R/W	ECC On/Off Register
C_BASEADDR + 0xC	CE_CNT	R/W	Correctable Error Counter Register
C_BASEADDR + 0x100	CE_FFD	R	Correctable Error First Failing Data Register
C_BASEADDR + 0x180	CE_FFE	R	Correctable Error First Failing ECC Register
C_BASEADDR + 0x1C0	CE_FFA	R	Correctable Error First Failing Address Register
C_BASEADDR + 0x200	UE_FFD	R	Uncorrectable Error First Failing Data Register
C_BASEADDR + 0x280	UE_FFE	R	Uncorrectable Error First Failing ECC Register
C_BASEADDR + 0x2C0	UE_FFA	R	Uncorrectable Error First Failing Address Register
C_BASEADDR + 0x300	FI_D	W	Fault Inject Data Register
C_BASEADDR + 0x380	FI_ECC	W	Fault Inject ECC Register

ECC Status Register (ECC_STATUS)

This register holds information on the occurrence of 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, i.e. 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 7: **ECC Status Register (ECC_STATUS)**

Reserved		ECC_STATUS	
0	29	30	31

Table 8: **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 will be 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 9: **ECC Interrupt Enable Register (ECC_EN_IRQ)**

Reserved		ECC_EN_IRQ	
0	29	30	31

Table 10: **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 ECC Status Register will be propagated to the Interrupt signal. if ‘0’ the value of the CE_STATUS bit of ECC Status Register will not be 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 will be propagated to the Interrupt signal. if ‘0’ the value of the UE_STATUS bit of ECC Status Register will not be 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 BRAM ECC bits have not been initialized at startup it is necessary to initialize them manually, before enabling the ECC checking. The ECC initialization is done by performing a read followed by a write on the whole BRAM contents.

The register is implemented if C_ECC_ONOFF_REGISTER is set to 1.

Table 11: **ECC On/Off Register (ECC_ONOFF)**

Reserved		ECC_ONOFF
0	30	31

Table 12: **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 will not wrap around, but rather stop incrementing and remain 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 13: **Correctable Error Counter Register (CE_CNT)**

Reserved		CE_CNT	
0	31-C_CE_COUNTER_WIDTH	32-C_CE_COUNTER_WIDTH	31

Table 14: **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 will be re-enabled to store the data of the next correctable error. Storing of the failing data is enabled after reset.

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

Table 15: **Correctable Error First Failing Data Register (CE_FFD)**

CE_FFD	
0	31

Table 16: **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 will be 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 17: **Correctable Error First Failing ECC Register (CE_FFE)**

Reserved		CE_FFE	
0	24	25	31

Table 18: **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 will be 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 19: **Correctable Error First Failing Address Register (CE_FFA)**

CE_FFA	
0	31

Table 20: **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 will be re-enabled to store the data of the next uncorrectable error. Storing of the failing data is enabled after reset.

The register is implemented if C_UE_FAILING_REGISTERS is set to 1.

Table 21: **Uncorrectable Error First Failing Data Register (UE_FFD)**

UE_FFD	
0	31

Table 22: **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 will be 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 23: **Uncorrectable Error First Failing ECC Register (UE_FFE)**

Reserved		UE_FFE	
0	24	25	31

Table 24: **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 will be 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 25: **Uncorrectable Error First Failing Address Register (UE_FFA)**

UE_FFA	
0	31

Table 26: **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 BRAM 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 BRAM 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.

Note that injecting faults should be performed in a critical region in software, i.e. writing this register and the subsequent write to LMB BRAM must not be interrupted.

Table 27: **Fault Injection Data Register (FI_D)**

FI_D	
0	31

Table 28: **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 BRAM 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 BRAM. 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.

Note that injecting faults should be performed in a critical region in software, i.e. writing this register and the subsequent write to LMB BRAM must not be interrupted.

Table 29: **Fault Injection ECC Register (FI_ECC)**

Reserved		FI_ECC	
0	24	25	31

Table 30: **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.

Design Implementation

Design Tools

The LMB BRAM interface controller design is hand written. The NGC netlist output from XST is then input to the Xilinx Alliance tool suite for actual device implementation.

Target Technology

The target technology is an FPGA listed in the Supported Device Family field of the LogiCORE Facts table.

Device Utilization and Performance Benchmarks

Because the BRAM interface controller is a module that is used with other design pieces in the FPGA, the utilization and timing numbers reported in this section are just estimates, and the actual utilization of FPGA resources and timing of the BRAM interface controller design will vary from the results reported here.

Table 31: **Performance and Resource Utilization Benchmarks on Virtex-6 (xc6vlx240t-1-ff1156)**

Parameter Values (other parameters at default value)									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

Programming Model

Supported Memory Sizes

For supported BRAM memory sizes, see DS444 IP Processor Block RAM (BRAM) Block (v1.00.a).

Example Base Address, High Address Specifications

The base address (C_BASEADDR) and high address (C_HIGHADDR) must specify a valid range for the BRAM that is attached to the BRAM Controller. The range (C_HIGHADDR – C_BASEADDR) specified by the high address and 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 above. In addition, the *n* least significant bits of C_BASEADDR must be equal to 0.:

Table 32: 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* for details on the transaction signaling.

Support

Xilinx provides technical support for this LogiCORE 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.*

Reference Documents

See Xilinx Application note 645 [XAPP645], *http://www.xilinx.com/support/documentation/application_notes/xapp645.pdf*, for detailed information about ECC and the Hamming code used by the LMB BRAM Interface Controller.

Revision History

The following table shows the revision history for this document:

Date	Version	Description of Revisions
1/12/06	1.0	Initial release.
7/28/08	1.2	Added QPro Virtex-4 Hi-Rel and QPro Virtex-4 Rad Tolerant FPGA support.
10/1/08	1.3	Initial release of v2.10.b
12/15/08	1.4	In LogiCORE IP Facts Table, replaced device family listing and tool name(s) with link to PDF file; added link to special disclaimer on first page.
4/24/09	1.5	Replaced references to supported device families and tool name(s) with hyperlink to PDF file.
3/2/10	1.6	Incorporated CR543063; inserted link to Block BRAM DS for supported BRAM sizes data, converted to current DS template.
3/1/11	1.7	Initial release of v3.00.a. ECC capability added.
6/22/11	1.8	Changed maximum value for parameter CE_COUNTER_WIDTH to 31, CR595448. Corrected Register Address Map in Table 6.
6/22/11	1.9	Initial release of v3.00.b. Incorporated CR606663.

Notice of Disclaimer