XPS Block RAM (BRAM) Interface Controller (v1.00b)

DS596 March 1, 2011 **Product Specification**

Introduction

The XPS BRAM Interface Controller is a Xilinx IP module that incorporates a PLB V4.6 (Processor Local Bus) interface. This controller is designed to be byte accessible. Any access size (in bytes) up to the parameterized data width of the BRAM is permitted. The XPS BRAM Interface Controller is the interface between the PLBV46 and the bram_block peripheral. A BRAM memory subsystem consists of the controller along with the actual BRAM components that are included in the bram_block peripheral. If the text-based Microprocessor Hardware Specification (MHS) file is used for design entry in EDK, then the xps_bram_if_cntlr and bram_block must both be explicitly instantiated.

Features

- PLB V4.6 bus interface with byte enable support
  - Xilinx Baseline and Performance Slave operation modes
  - 32, 64, and 128-bit wide PLB interface support
  - 32, 64, and 128-bit BRAM width support
  - 32-bit addressing support
- Supports up to three transfer types
  - Single data beat
  - Cacheline 4/8
  - Fixed Length Burst 2 to 16 data beats
  - Used in conjunction with Xilinx EDK generated bram_block peripheral to provide total BRAM memory solution

LogiCORE IP Facts Table	
Core Specifics	
Supported Device Family [(1)]	Spartan-3A/3A DSP, Spartan-3, Spartan-3E, Automotive Spartan-3E/3A/3A DSP/ Spartan-6, Virtex-4/4Q/4QV, Virtex-5/5FX, Virtex-6/6CX
Supported User Interfaces	Not Applicable
Resources Used	See Table 7.
Provided with Core	
Documentation	Product Specification
Design Files	VHDL
Example Design	Not Provided
Test Bench	Not Provided
Constraints File	EDK TCL Generated
Simulation Model	Not Provided
Tested Design Tools	
Design Entry Tools	ISE® 12.1 or later
Simulation	ModelSim PE/SE 6.5c or later
Synthesis Tools	XST
Support	
Provided by Xilinx, Inc.	

Notes:

1. For a complete listing of supported devices, see the release notes for this core.

© Copyright 2006-2009, 2011 Xilinx, Inc. XILINX, the Xilinx logo, Virtex, Spartan, ISE and other designated brands included herein are trademarks of Xilinx in the United States and other countries. All other trademarks are the property of their respective owners.

Functional Description

Describe basic building blocks of core. The information in this section should answer most pre-sales FAQs.

Figure 1 illustrates the functional composition of the core. The core design is written using VHDL and is named xps_bram_if_cntlr.vhd. Because the module is designed to be connected to a separate BRAM memory module, it does not provide for the BRAM memory instantiations. There are two primary components are the Slave Interface modules — one for Baseline operation mode and one for Performance operation mode. The modules are custom designs and provide all address decoding (32-bit address) and supports 32, 64, and 128-bit data accesses of the BRAM. Depending on parameterization assignments, these accesses may be comprised of single data beat, cacheline 4 or 8, and fixed length burst transfers. The Interface modules are designed to be compatible with the Processor Local Bus Architecture Specification (v4.6).

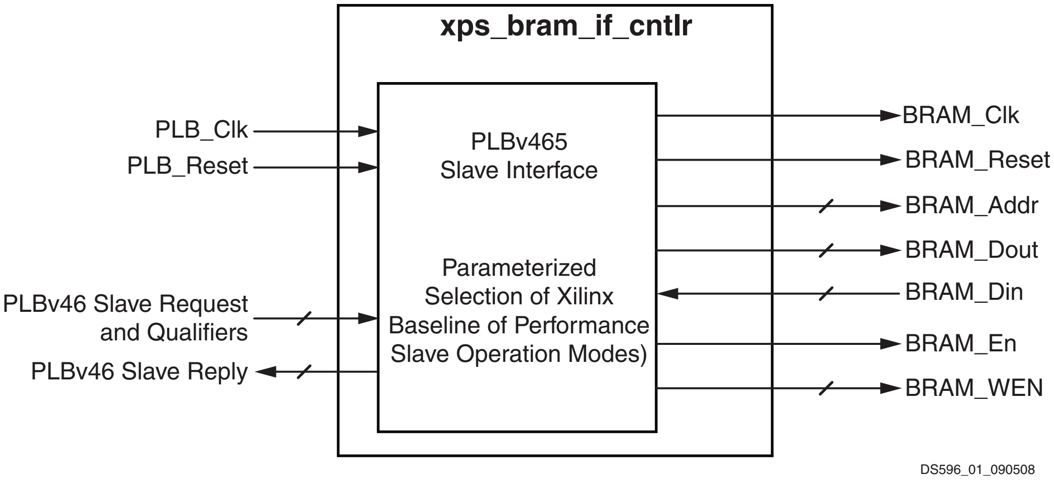


Figure 1: **XPS BRAM Interface Controller Block Diagram**

I/O Signals

The XPS BRAM Interface Controller signals are listed and described in Table 1.

Table 1: **XPS BRAM Interface Controller I/O Signal Description**

Signal Name	Interface	Signal Type	Init Status	Description
System Signals				
SPLB_Clk	System	I		PLB system synchronization clock
SPLB_Rst	System	I		PLB system PLB reset
PLB Slave Request Signals				
PLB_ABus(0:31)	PLB	I		PLB address bus
PLB_UABus(0:31)	PLB	I		PLB upper address bus
PLB_PAValid	PLB	I		PLB primary address valid indicator
PLB_SAValid	PLB	I		PLB secondary address valid indicator
PLB_rdPrim	PLB	I		PLB secondary to primary read request promotion
PLB_wrPrim	PLB	I		PLB secondary to primary write request promotion

Table 1: **XPS BRAM Interface Controller I/O Signal Description**

Signal Name	Interface	Signal Type	Init Status	Description
PLB_masterID(0:C_SPLB_MID_WIDTH-1)	PLB	I		PLB requesting master identification
PLB_abort	PLB	I		PLB bus request abort
PLB_busLock	PLB	I		PLB bus lock
PLB_RNW	PLB	I		PLB read not write
PLB_BE(0:(C_SPLB_DWIDTH / 8) -1)	PLB	I		PLB byte enables
PLB_MSize(0:1)	PLB	I		PLB master data bus size
PLB_size(0:3)	PLB	I		PLB transfer size
PLB_type(0:2)	PLB	I		PLB transfer type
PLB_lockErr	PLB	I		PLB lock error
PLB_wrDBus(0:C_SPLB_DWIDTH -1)	PLB	I		PLB write data bus
PLB_wrBurst	PLB	I		PLB burst write transfer
PLB_rdBurst	PLB	I		PLB burst read transfer
PLB_wrPendReq	PLB	I		PLB pending write request
PLB_rdPendReq	PLB	I		PLB pending read request
PLB_wrPendPri(0:1)	PLB	I		PLB pending write request priority
PLB_rdPendPri(0:1)	PLB	I		PLB pending read request priority
PLB_reqPri(0:1)	PLB	I		PLB current request priority
PLB_TAttribute(0:15)	PLB	I		PLB Transfer Attribute bus
PLB Slave Reply Signals				
Sl_addrAck	PLB	O	’0’	Slave address acknowledge
Sl_SSize(0:1)	PLB	O	Zeros	Slave data bus size
Sl_wait	PLB	O	’0’	Slave wait indicator
Sl_rearbitrate	PLB	O	’0’	Slave rearbitrate bus indicator
Sl_wrDAck	PLB	O	’0’	Slave write data acknowledge
Sl_wrComp	PLB	O	’0’	Slave write transfer complete indicator
Sl_wrBTerm	PLB	O	’0’	Slave terminate write burst transfer
Sl_rdDBus(0:C_SPLB_DWIDTH -1)	PLB	O	Zeros	Slave read bus
Sl_rdWdAddr(0:3)	PLB	O	Zeros	Slave read word address
Sl_rdDAck	PLB	O	’0’	Slave read data acknowledge
Sl_rdComp	PLB	O	’0’	Slave read transfer complete indicator
Sl_rdBTerm	PLB	O	’0’	Slave read burst terminate indicator
Sl_MBusy(0:C_SPLB_NUM_MASTERS-1)	PLB	O	Zeros	Slave busy indicator (1 bit for each PLB Master)

Table 1: **XPS BRAM Interface Controller I/O Signal Description**

Signal Name	Interface	Signal Type	Init Status	Description
Sl_MWrErr(0:C_SPLB_ NUM_MASTERS-1)	PLB	O	Zeros	Slave write error indicator (1 bit for each PLB Master)
Sl_MRdErr(0:C_SPLB_ NUM_MASTERS-1)	PLB	O	Zeros	Slave read error indicator (1 bit for each PLB Master)
Sl_MIRQ(0:C_SPLB_ NUM_MASTERS-1)	PLB	O	Zeros	Slave interrupt indication (1 bit for each PLB Master)
BRAM Interface Signals				
BRAM_Rst	BRAM Block	O	Follows SPLB_Rst	Active high signal used to initialize the BRAM Memory. It is an echo of the SPLB_Rst input port
BRAM_Clk	BRAM Block	O	Follows SPLB_Clk	BRAM Clock used to synchronize the operation of the BRAM memory block. It is an Echo of the SPLB_Clk input port
BRAM_EN	BRAM Block	O	’0’	Active high signal indicating to the BRAM memory that an access is in progress.
BRAM_WEN(0:(C_SPLB_ NATIVE_DWIDTH/8)-1)	BRAM Block	O	Zeros	A bus of signals used to qualify write operation to the BRAM memory. Each bit within the bus corresponds to a byte position within the BRAM data word. The assertion of a WEN bit in conjunction with the assertion of the BRAM_EN signal and a rising edge of the BRAM_Clk constitutes a valid write condition to the address specified by the BRAM_Addr.
BRAM_Addr(0:C_ SPLB_AWIDTH-1)	BRAM Block	O	Zeros	Address bus to the BRAM memory. See Section BRAM_Addr Usage, page 7 for a discussion of the specific bits needed for the typical BRAM memory application.
BRAM_Din(0:C_SPLB_ NATIVE_DWIDTH-1)	BRAM Block	I	Zeros	Read Data bus from the BRAM memory. Data from the BRAM Memory is assumed to be valid on this bus at the rising edge of the next sequential BRAM_Clk cycle after the clock cycle in which the desired assertion of the BRAM_Addr and the BRAM_EN signals are made.
BRAM_Dout(0:C_SPLB_ NATIVE_DWIDTH-1)	BRAM Block	O	Zeros	Output data bus used to transfer Write data to the BRAM Memory. Write data is written when the BRAM_EN and the BRAM_WEN signals are asserted and a rising edge of BRAM_Clk occurs.

XPS BRAM Interface Controller Parameters

To allow you to obtain an XPS BRAM Interface Controller that is uniquely tailored for your system, certain features can be parameterized in the XPS BRAM Interface Controller design. This allows you to configure a design that only utilizes the resources required by your system, and operates with the best possible performance. The features that can be parameterized in Xilinx XPS BRAM Interface Controller design are shown in Table 2.

Table 2: **XPS BRAM Interface Controller Design Parameters**

Feature/ Description	Parameter Name	Allowable Values	Default Values	VHDL Type
User Specified Features				
BRAM Base Address	C_BASEADDR	System Address value of C_SPLB_AWIDTH bits wide [2]	FFFF_FFFF [1]	std_logic_ vector
BRAM HIGH Address	C_HIGHADDR	System Address value of C_SPLB_AWIDTH bits wide [2]	0000_0000 [1]	std_logic_ vector
Specify the required BRAM Interface Data Width	C_SPLB_NATIVE_ DWIDTH	32, 64, or 128	32	integer
Tool Specified Features				
PLB Address Bus Width	C_SPLB_AWIDTH [3]	32, 36 [4]	32	integer
PLB Data Bus Width	C_SPLB_DWIDTH[3]	32, 64, or 128	32	integer
Number of Masters	C_SPLB_NUM_ MASTERS[3]	1 - 16	2	integer
Width of Master ID Bus	C_SPLB_MID_ WIDTH[3]	roundup(log$_2$(C_SPLB_NUM_ MASTERS))	1	integer
Optimization mode for PLB burst and cacheline transfers	C_SPLB_SUPPORT_ BURSTS[3]	0 = Optimized for resource savings at the price of only supporting single data beat transfers 1 = Transfer Optimized Mode for Single Data Beat, Fixed Length Burst, and Cacheline 4/8 up to 128- bits wide at the price of higher resource utilization of the core	1	integer
Enable Support for Point to Point interconnect configuration	C_SPLB_P2P[3]	0 = Normal Mode 1 = Point to Point Optimizations Enabled [5]	0	Integer
Indicates the Smallest Width Master that may access the XPS BRAM Interface Controller	C_SPLB_SMALLEST _MASTER[3]	32, 64, or 128	32	Integer
Target FPGA device family	C_FAMILY[3]	spartan3a, spartan3adsp, spartan3e, aspartan3, aspartan3e, aspartan3adsp Spartan-6, virtex-4, virtex4q, virtex5, virtex5fx, virtex6, virtex6cx	virtex5	String

Notes:

1. Default values are specified for C_BASEADDR and C_HIGHADDR to insure that they are set by the User. If the value is not set, an implementation error will be generated.
2. C_BASEADDR value must be a power of 2 and a multiple of the desired address range, where the address range is (C_HIGHADDR+1) - C_BASEADDR. Example base address settings are shown in Table 4 in the Setting the C_BASEADDR and C_HIGHADDR Parameters, page 6 section.
3. These parameters are calculated and automatically assigned by the EDK XPS tools during the system creation process
4. Xilinx EDK limits addressing to 32-bits.
5. Point to Point optimizations include removal of address decoding. This mode is not usable in a shared bus interconnect environment.

Parameter - Port Dependencies

Table 3: **XPS BRAM Interface Controller Parameter-Port Dependencies**

Generic or Port	Name	Affects Port	Depends on Parameter	Relationship Description
		Design Parameters		
G1	C_SPLB_NATIVE_DWIDTH	P1, P2, P3		Port widths are set directly or derived from the parameter value
G2	C_SPLB_NUM_MASTERS	P5, P6, P7, P8		Port widths are set directly or derived from the parameter value
G3	C_SPLB_MID_WIDTH	P12		Port width is set directly by the parameter value
G4	C_SPLB_DWIDTH	P9, P10, P11		Port widths are set directly or derived from the parameter value
G5	C_SPLB_AWDITH	P4		Port width is set directly by the parameter value
		I/O Signals		
P1	BRAM_Din		G1	Port width is set directly by the parameter value
P2	BRAM_Dout		G1	Port width is set directly by the parameter value
P3	BRAM_WEN		G1	Port width is derived from the parameter value
P4	BRAM_ABus		G5	Port width is set directly by the parameter value
P5	Sl_MBusy		G2	Port width is set directly by the parameter value
P6	Sl_MWrErr		G2	Port width is set directly by the parameter value
P7	Sl_MRdErr		G2	Port width is set directly by the parameter value
P8	Sl_MIRQ		G2	Port width is set directly by the parameter value
P9	PLB_BE		G4	Port width is derived from the parameter value
P10	PLB_wrDBus		G4	Port width is set directly by the parameter value
P11	Sl_rdDBus		G4	Port width is set directly by the parameter value
P12	PLB_masterID		G3	Port width is set directly by the parameter value

Register Descriptions

There are no User programmable registers for this IP core.

Application Information

Setting the C_BASEADDR and C_HIGHADDR Parameters

The base address (C_BASEADDR) and high address (C_HIGHADDR) parameters must specify a valid range for the BRAM memory configuration that is attached to the xps_bram_if_cntlr. The range (C_HIGHADDR – C_BASEADDR) specified by the high address and base address must be equal to 2^n bytes minus 1, where n is a positive integer and 2^n is a valid memory size as shown in Table 5 or Table 6 depending on the target device family.

In addition, the C_BASEADDR value must be a multiple of the desired memory size (or address range). Examples of valid C_BASEADDR and C_HIGHADDR settings for various memory sizes are shown in Table 4.

Table 4: **Example Address Range Specifications for C_BASEADDR and C_HIGHADDR**

Memory Size (Bytes)	Basic Address Range Required	C_BASEADDR	C_HIGHADDR
8K	0x0000_0000 to 0x0000_1FFF	0xE00A0000	0xE00A1FFF
16K	0x0000_0000 to 0x0000_3FFF	0x3FF00000	0x3FF03FFF
32K	0x0000_0000 to 0x0000_7FFF	0x82000000	0x82007FFF
64K	0x0000_0000 to 0x0000_FFFF	0xB0010000	0xB001FFFF
128K	0x0000_0000 to 0x0001_FFFF	0x00820000	0x0083FFFF
256K	0x0000_0000 to 0x0003_FFFF	0xFFFC0000	0xFFFFFFFF

Point to Point vs. Shared Bus Configuration Differences

The input parameter C_SPLB_P2P configures PLB interface logic for the type of interconnect topology used to connect the XPS BRAM Interface Controller in the Host system. When.C_SPLB_P2P is set to 0, the interface is configured for shared bus. In this mode, the interface incorporates the normal address decoding logic. When.C_SPLB_P2P is set to 1, the interface is configured for point to point type connection with a PLB Master. Address decoding is eliminated and the interface responds to all valid requests from the Master regardless of address.

Special Resource Optimized Mode

The XPS BRAM Interface Controller provides a special resource optimized mode for resource critical applications not requiring high data transfer performance. This mode is activated when the C_NATIVE_DWIDTH parameter is set to 32 and the C_SUPPORT_BURSTS parameter is set to 0. In this mode, only 32-bit single data beat transfers are supported. This is the Xilinx Baseline operation mode.

Backend BRAM Block Interface

There are 7 ports comprising the backend interface of the XPS BRAM Interface Controller. These ports are used to interface directly with a User's BRAM memory. The ports are:

- BRAM_Rst
- BRAM_Clk
- BRAM_EN
- BRAM_WEN (bus)
- BRAM_Addr (bus)
- BRAM_Din (bus)
- BRAM_Dout (bus)

Port properties and a brief description of each can be found in Table 1, page 2.

BRAM_Addr Usage

The address bus output port BRAM_Addr is sized according to the C_SPLB_AWIDTH parameter value. Currently, this is fixed by the EDK tools at 32 bits. This is obviously more addressing capability than the usual BRAM memory requires. In addition, the address provided is a byte address that is derived from the PLB start address provided during the Address Phase of the associated PLB request. Generally, a BRAM memory array configurations do not use the address bits that delineate byte positions within the base data width of the memory. Instead, individual byte write qualifier signals are used. These are provided by the BRAM_WEN output port. Thus, the lower order address bits of the BRAM_Addr bus will be unused. The User is required to rip the appropriate address signals from the BRAM_Addr bus for use by the BRAM memory configuration being implemented. The number of address bits required and used address bits for Spartan®--3 and Virtex®--4 devices are shown in Table 5. and in Table 6 for Virtex-5 devices.

Supported BRAM Memory Configurations

Typical BRAM memory configurations that are supported by this BRAM Controller are shown in Table 5 for Virtex-4, Spartan-3, and Spartan-6 devices and in Table 6 for Virtex-5 and Virtex-6 devices. **The BRAM instantiations are not provided by this core. They are part of the bram_block module generated by the EDK XPS tools during embedded system creation.**

Table 5: **Supported BRAM Memory sizes for Virtex-4, Spartan-3, and Spartan-6 FPGAs**

Native Data Width Size (bits)	Supported Memory Sizes (Bytes) / BRAM Memory Configuration (Depth x Width)	Number of BRAM primitives (18Kbit ea.) required	Number of BRAM_Addr bits required	Typical BRAM_Addr(0:31) bit usage for BRAM width
C_PORT_DWIDTH = 32				
32	8K / (2,048x32)	4 [(1)]	11	BRAM_Addr(17:29)
32	16K / (4,096x32)	8	12	BRAM_Addr(16:29)
32	32K / (8,192x32)	16	13	BRAM_Addr(15:29)
32	64K / (16,384x32)	32	14	BRAM_Addr(14:29)
C_PORT_DWIDTH = 64				
64	16K / (2,048x64)	8 [(2)]	11	BRAM_Addr(18:28)
64	32K / (4,096x64)	16	12	BRAM_Addr(17:28)
64	64K / (8,192x64)	32	13	BRAM_Addr(16:28)
64	128K / (16,384x64)	64	14	BRAM_Addr(15:28)
C_PORT_DWIDTH = 128				
128	32K / (2,048x128)	16 [(3)]	11	BRAM_Addr(16:27)
128	64K / (4,096x128)	32	12	BRAM_Addr(15:27)
128	128K / (8,192x128)	64	13	BRAM_Addr(14:27)
128	256K / (16,384x128)	128	14	BRAM_Addr(13:27)

Notes:

1. A minimum of 4 BRAM primitives are required to maintain byte write capability for a 32-bit native data width BRAM array.
2. A minimum of 8 BRAM primitives are required to maintain byte write capability for a 64-bit native data width BRAM array.
3. A minimum of 16 BRAM primitives are required to maintain byte write capability for a 128-bit native data width BRAM array.

Table 6: **Supported BRAM Memory sizes for Virtex-5 and Virtex-6 FPGAs**

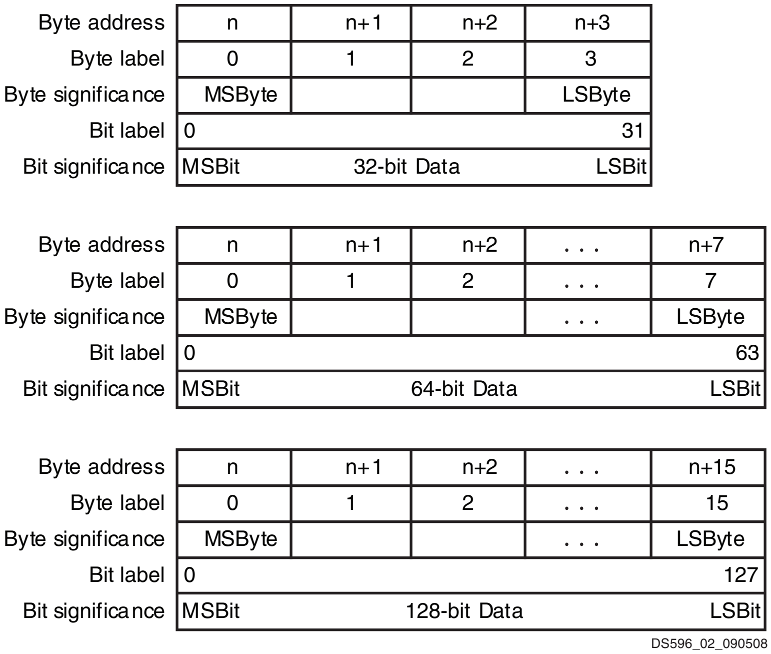
Native Data Width Size (bits)	Supported Memory Sizes (Bytes) / BRAM Memory Configuration (Depth x Width)	Number of BRAM primitives (36Kbit ea.) required	Number of BRAM_Addr bits required	Typical BRAM_Addr(0:31) bit usage for 64-bit wide Memory (8 byte lanes)
C_NATIVE_DWIDTH = 32				
32	4K / (1,024x32)	1 [(1)]	10	BRAM_Addr(20:29)
32	8K / (2,048x32)	2	11	BRAM_Addr(19:29)
32	16K / (4,096x32)	4	12	BRAM_Addr(18:29)
32	32K / (8,192x32)	8	13	BRAM_Addr(17:29)
32	64K / (16,384x32)	16	14	BRAM_Addr(16:29)
32	128K / (32,768x32)	32	15	BRAM_Addr(15:29)
32	256K / (65,536x32)	64	16	BRAM_Addr(14:29)
C_NATIVE_DWIDTH = 64				
64	8K / (1,024x64)	2 [(1)]	10	BRAM_Addr(19:28)
64	16K / (2,048x64)	4	11	BRAM_Addr(18:28)
64	32K / (4,096x64)	8	12	BRAM_Addr(17:28)
64	64K / (8,192x64)	16	13	BRAM_Addr(16:28)
64	128K / (16,384x64)	32	14	BRAM_Addr(15:28)
64	256K / (32,768x64)	64	15	BRAM_Addr(14:28)
64	512K / (65,536x64)	128	16	BRAM_Addr(13:28)
C_NATIVE_DWIDTH = 128				
128	16K / (1,024x128)	4 [(1)]	10	BRAM_Addr(18:27)
128	32K / (2,048x128)	8	11	BRAM_Addr(17:27)
128	64K / (4,096x128)	16	12	BRAM_Addr(16:27)
128	128K / (8,192x128)	32	13	BRAM_Addr(15:27)
128	256K / (16,384x128)	64	14	BRAM_Addr(14:27)
128	512K / (32,768x128)	128	15	BRAM_Addr(13:27)
128	1024K / (65,536x128)	256	16	BRAM_Addr(12:27)

Notes:

1. Virtex-5 and Virtex-6 BRAM primitives have up to 4 byte enables per primitive.

Data Types and Organization

The BRAM interface ports are designed to interface with a BRAM Memory via a big-endian data structure. The byte ordering and bit numbering for this structure is shown in Figure 2.



Byte address	n	n+1	n+2	n+3
Byte label	0	1	2	3
Byte significance	MSByte			LSByte
Bit label	0			31
Bit significance	MSBit	32-bit Data		LSBit

Byte address	n	n+1	n+2	. . .	n+7
Byte label	0	1	2	. . .	7
Byte significance	MSByte			. . .	LSByte
Bit label	0				63
Bit significance	MSBit		64-bit Data		LSBit

Byte address	n	n+1	n+2	. . .	n+15
Byte label	0	1	2	. . .	15
Byte significance	MSByte			. . .	LSByte
Bit label	0				127
Bit significance	MSBit		128-bit Data		LSBit

DS596_02_090508

Figure 2: **Big-Endian Byte Ordering and Left to Right Bit Numbering**

Typical PLB Timing

The following text and sequence of figures describes and shows typical timing relationships of the Slave reply signals of this core. All PLB transactions are made up of two time phases known as the Address Phase and the Data Phase. The Address Phase occurs first and is the period of time where the PLB arbiter gates the PLB Master's request address and qualifiers to the PLB and indicates these signals are valid via the assertion of the PLB_PAValid signal. It is during this phase that the PLB Slave device is required to decode the incoming address and respond if the address is within the Slave's assigned address space. The address phase is completed when the addressed Slave device responds with an assertion of the Sl_addrAck or Sl_rearbitrate. When an Address Phase is completed with a Sl_addrAck assertion, then the PLB Data Phase is initiated. This is the period where the actual data transfer occurs. It is during the Data Phase that the BRAM interface signals are active. The XPS BRAM Interface Controller supports three PLB transfer types for Data Phase operations. These are:

- Single Data Beat Read and Write
  - 1-4 bytes (32-bit Native Data Width)
  - 1-8 bytes (64-bit Native Data Width)
  - 1-16 bytes (128-bit Native Data Width)
- Cacheline Read and Write (4 word and 8 word; Note that 16 word cacheline requests are not supported)
- Fixed Length Burst Read and Write (2 to 16 Data Beats per request)
  - Transfer size of word supported for 32-bit Native Data Width
  - Transfer size of word and double word supported for 64-bit Native Data Width
  - Transfer size of word, double word, and quad word for 128-bit Native Data Width

A Data Phase is generally signaled to be completed with the assertion of the Sl_rdComp or Sl_wrComp signal by the responding Slave device. At that time, the arbiter may arbitrate and drive the next pending request onto the PLB. For information on these and other PLB protocol and signaling scenarios, see *IBM 128-bit Processor Local Bus, Architecture Specification, Version 4.6*.

Single Data Beat Read Transfer

A Single Data Beat Read is used to transfer 1 to 4 bytes (32-bit Native Data Width), 1 to 8 bytes (64-bit Native DWidth), or 1 to 16 bytes (128-bit Native DWidth) of data from the target Slave to the requesting PLB Master. The requested byte lanes are asserted by the Master via the PLB_BE input bus. However, in the case of the BRAM memory, a read is not destructive so the byte enable selections are not echoed to the BRAM interface ports. The XPS BRAM Interface Controller expects the BRAM memory to provide a full bus width of data for all read data beats. During a Single Data Beat read, the data is transferred during a single PLB clock cycle that is denoted by the assertion of the Sl_rdDAck and Sl_rdComp by the Slave. Figure 3 shows the timing of a Single Data Beat Read transfer.

This diagram is for a 64-bit wide PLB and a BRAM Native DWIDTH of 64 bits.

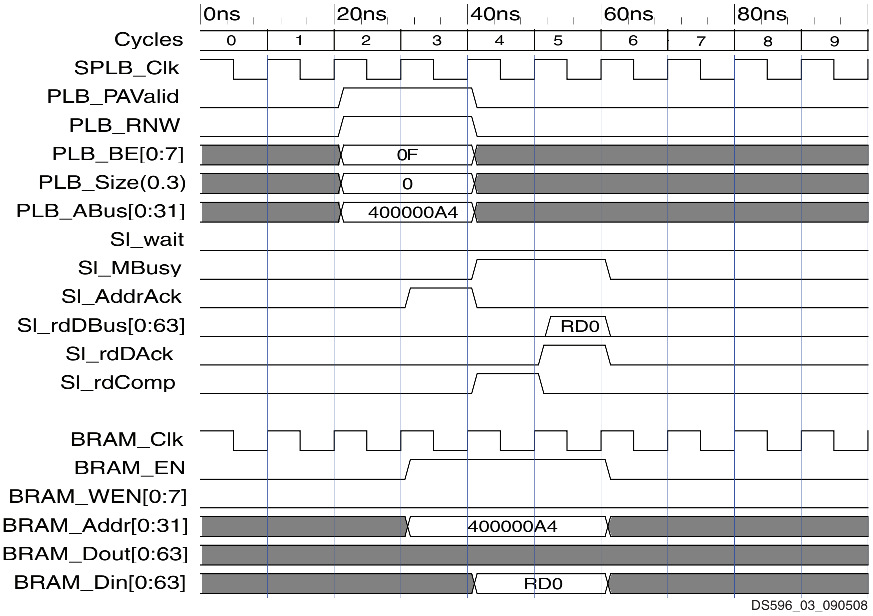


Figure 3: **Single Data Beat Read**

Single Data Beat Write Transfer

A Single Data Beat Write is used to transfer 1 to 4 bytes (32-bit Native Data Width), 1 to 8 bytes (64-bit Native DWidth), or 1 to 16 bytes (128-bit Native DWidth) of data from the requesting PLB Master to the target Slave. The data is transferred during a single PLB clock cycle that is denoted by the assertion of the Sl_wrDAck by the Slave. The Master specifies which byte positions are to be modified via the PLB_BE qualifier bus. The bit assertions within the PLB_BE bus are translated to the generation of the appropriate BRAM_WEN bit assertions that are sent to the User BRAM memory block. The signal timing of a Single Data Beat Write operation is shown in Figure 4.

This diagram is for a 64-bit wide PLB and a BRAM Native DWIDTH of 64 bits.

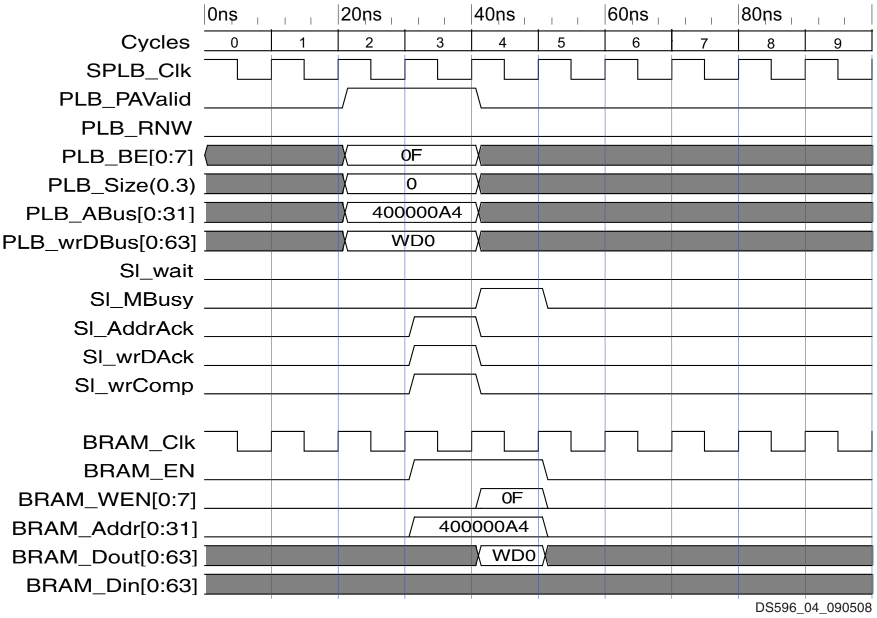
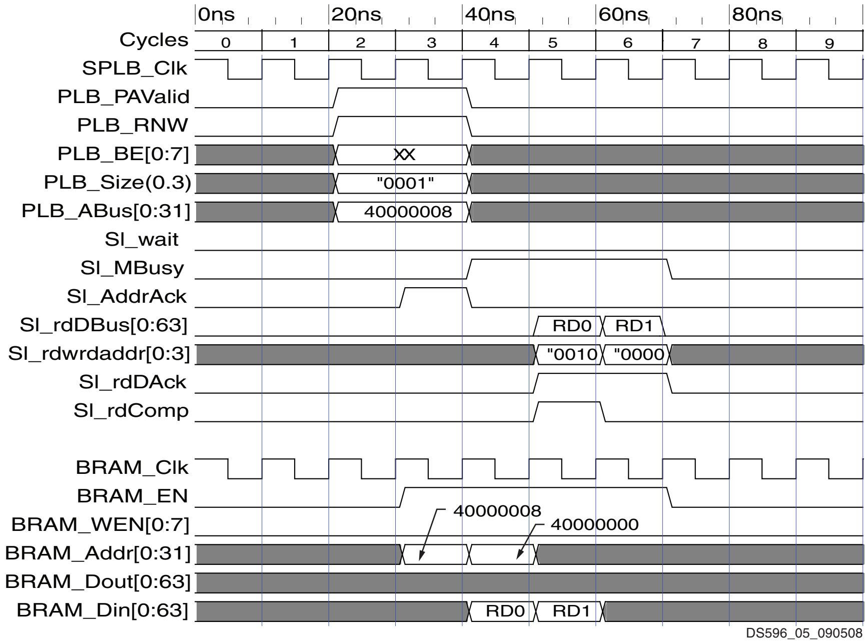


Figure 4: **Single Data Beat Write**

4-Word Cacheline Read (C_SPLB_SUPPORT_BURSTS = 1)

PLB V4.6 Cacheline 4-word read transactions are supported by the BRAM Interface Controller. A 4-word Cacheline read is completed in 2 data beats during the Data Phase on the PLB V4.6 bus as shown in Figure 5. All byte lanes are used during Cacheline reads.

This diagram is for a 64-bit wide PLB and a BRAM Native DWIDTH of 64 bits. The actual data width of the requesting Master and the actual Native Data Width of the BRAM Interface Controller can change the number of data beats required to complete the requested transfer.



DS596_05_090508

Figure 5: **4-Word Cacheline Read (C_SPLB_SUPPORT_BURSTS = 1)**

8-Word Cacheline Read (C_SPLB_SUPPORT_BURSTS = 1)

PLB V4.6 Cacheline 8-word read transactions are supported by the BRAM Interface Controller. An 8-word Cacheline read is completed in 4 data beats during the Data Phase on the PLB V4.6 bus as shown in Figure 6. All byte lanes are used during Cacheline reads.

This diagram is for a 64-bit wide PLB and a BRAM Native DWIDTH of 64 bits. The actual data width of the requesting Master and the actual Native Data Width of the BRAM Interface Controller can change the number of data beats required to complete the requested transfer.

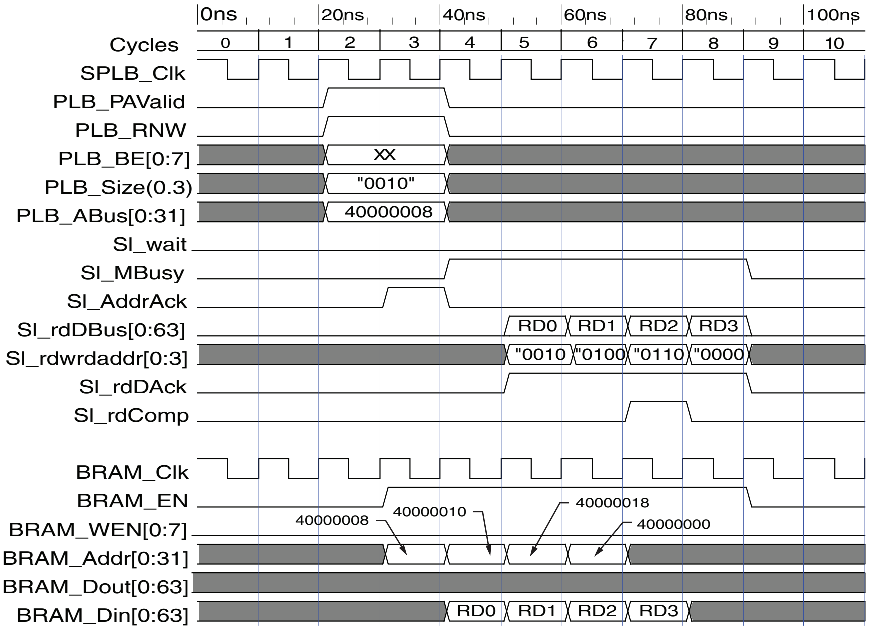


Figure 6: **8-Word Cacheline Read (C_SPLB_SUPPORT_BURSTS = 1)**

4-Word Cacheline Write (C_SPLB_SUPPORT_BURSTS = 1)

PLB V4.6 Cacheline 4-word write transactions are supported by the BRAM Interface Controller. A 4-word Cacheline write is completed in 2 data beats during the Data Phase on the PLB V4.6 bus as shown in Figure 7. All byte lanes are used during Cacheline writes and the starting address must be aligned to the start of the cacheline.

This diagram is for a 64-bit wide PLB and a BRAM Native DWIDTH of 64 bits. The actual data width of the requesting Master and the actual Native Data Width of the BRAM Interface Controller can change the number of data beats required to complete the requested transfer.

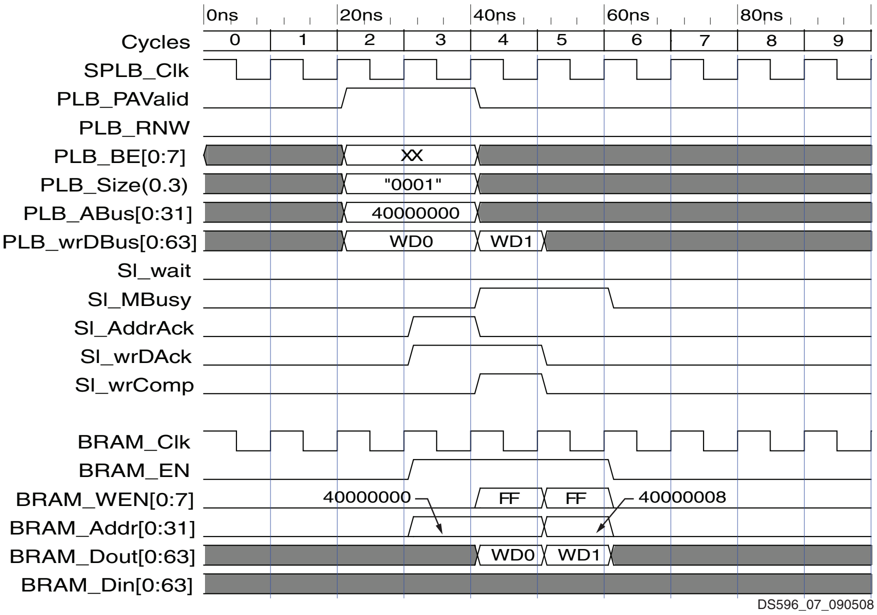


Figure 7: **4-Word Cacheline Write (C_SPLB_SUPPORT_BURSTS = 1)**

8-Word Cacheline Write (C_SPLB_SUPPORT_BURSTS = 1)

PLB V4.6 Cacheline 8-word write transactions are supported by the BRAM Interface Controller. An 8-word Cacheline write is completed in 4 data beats during the Data Phase on the PLB V4.6 bus as shown in Figure 8. All byte lanes are used during Cacheline writes and the starting address must be aligned to the start of the Cacheline.

This diagram is for a 64-bit wide PLB and a BRAM Native DWIDTH of 64 bits. The actual data width of the requesting Master and the actual Native Data Width of the BRAM Interface Controller can change the number of data beats required to complete the requested transfer.

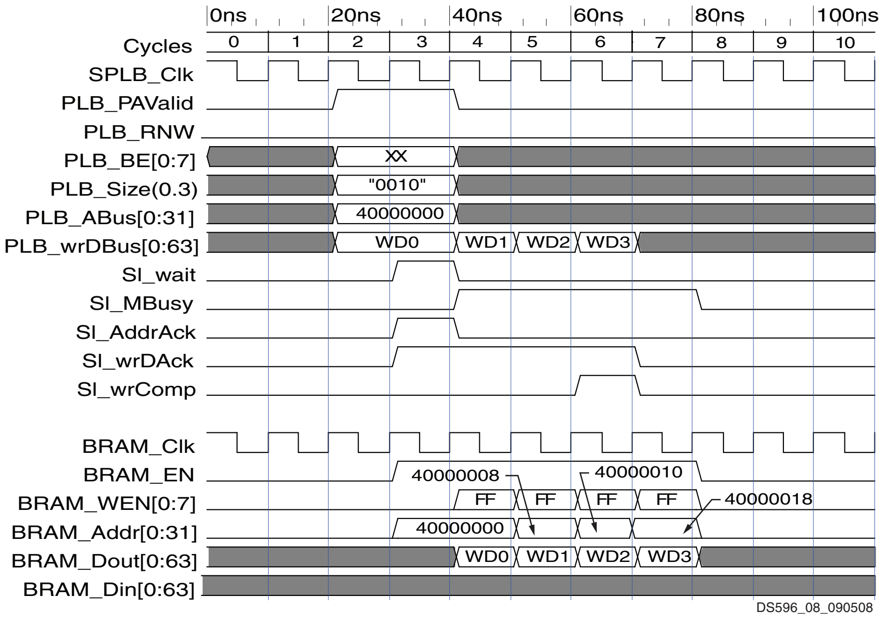


Figure 8: **8-Word Cacheline Write (C_SPLB_SUPPORT_BURSTS = 1)**

Fixed Length Burst Read (C_SPLB_SUPPORT_BURSTS = 1)

PLB V4.6 Fixed Length Burst Read transactions are supported by the BRAM Interface Controller. The transfer length may be 2 to 16 data beats with specified data widths of words, double-words, and quad-words. A 10 Double Word Fixed Length Burst read is shown in Figure 9.

This diagram is for a 64-bit wide PLB and a BRAM Native DWIDTH of 64 bits. The actual data width of the requesting Master and the actual Native Data Width of the BRAM Interface Controller can change the number of data beats required to complete the requested transfer.

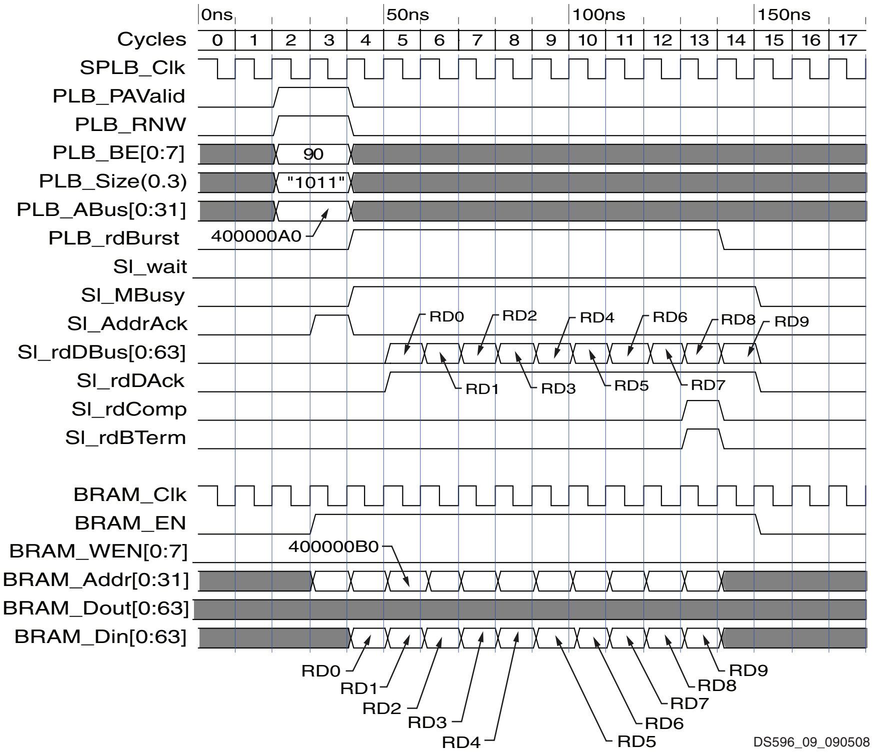


Figure 9: **Fixed Length Burst Read (C_SPLB_SUPPORT_BURSTS = 1)**

Fixed Length Burst Write (C_SPLB_SUPPORT_BURSTS = 1)

PLB V4.6 Fixed Length Burst Write transactions are supported by the BRAM Interface Controller. The transfer length may be 2 to 16 data beats with specified data widths of bytes, half-words, words, and double-words. A 12 double word Fixed Length Burst Write is shown in Figure 10.

This diagram is for a 64-bit wide PLB and a BRAM Native DWIDTH of 64 bits. The actual data width of the requesting Master and the actual Native Data Width of the BRAM Interface Controller can change the number of data beats required to complete the requested transfer.

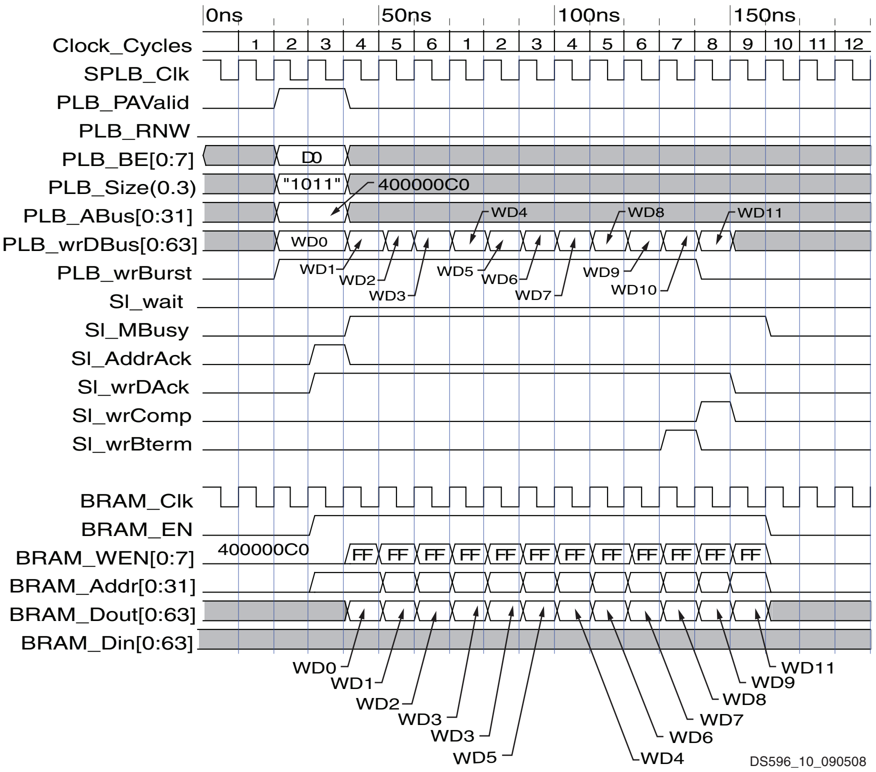


Figure 10: **Fixed Length Burst Write (C_SPLB_SUPPORT_BURSTS = 1)**

Design Implementation

Target Technology

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

Device Utilization and Performance Benchmarks

Core Performance

To analyze the XPS BRAM Interface Controller timing within the FPGA, a design was generated that enclosed the Core in a wrapper. For sizing estimates, a simple wrapper that connected all I/O to the ports of the wrapper was utilized. For determine the F_{Max} value, the wrapper was modified to incorporate input and output registers on all input and output ports respectively. FPGA performance and resource utilization benchmarks from the synthesis and mapping of the wrappers hosted in a xc5vlx110t-3 device are shown in Table 7.

Table 7: **XPS BRAM Interface Controller FPGA Performance and Resource Utilization Benchmarks**

Target FPGA			Device Resources			f_{MAX} (MHz)
C_SPLB_ SUPPORT_BURSTS	C_NATIVE_ DWIDTH	C_SPLB_ DWIDTH	Slices	Slice Flip- Flops	4-input LUTs	f_{MAX}
0	32	32		9	25	176
1	32	32		78	131	180
1	64	64		81	184	165
1	128	128		95	248	150

System Performance

To measure the system performance (F_{Max}) of this core, it was added to a Virtex-4 device system, a Virtex-5 device system, and a Spartan-3A device system as the Device Under Test (DUT) as shown in Figure 11, Figure 12, and Figure 13.

Because the XPS BRAM Interface Controller core will be used with other design modules in the FPGA, the utilization and timing numbers reported in this section are estimates only. When this core is combined with other designs in the system, the utilization of FPGA resources and timing of the core design will vary from the results reported here.

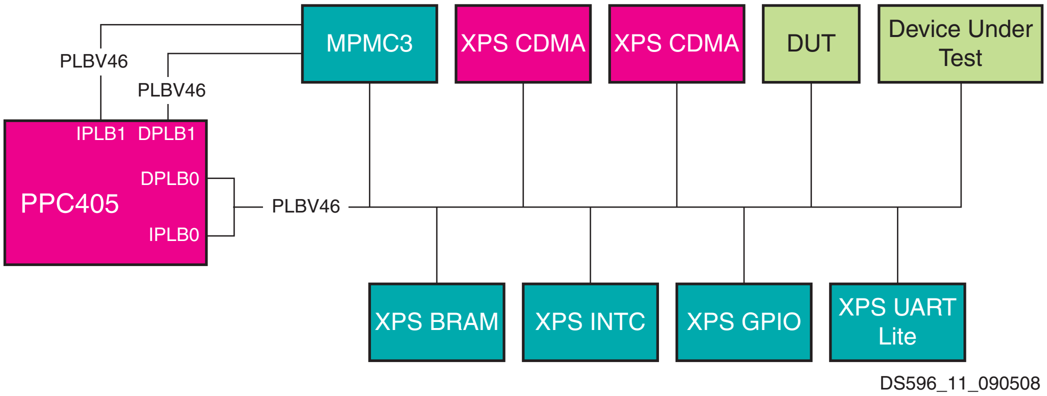


Figure 11: **Virtex-4 FX FPGA System**

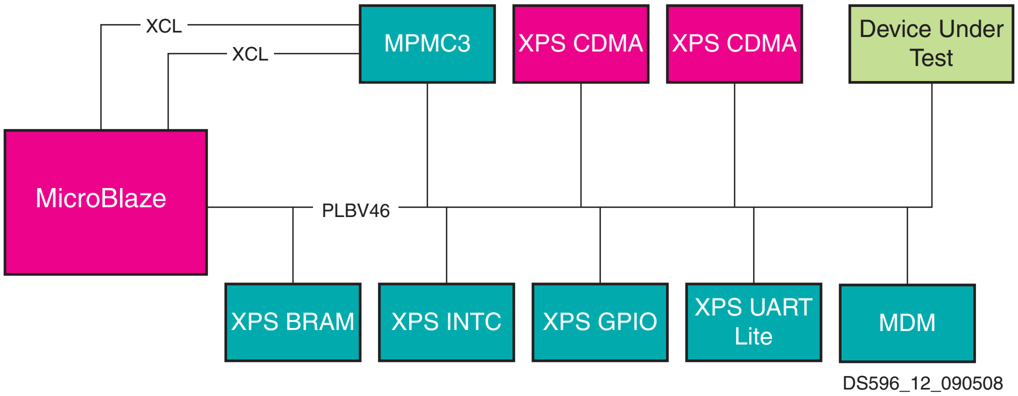


Figure 12: **Virtex-5 LX FPGA System**

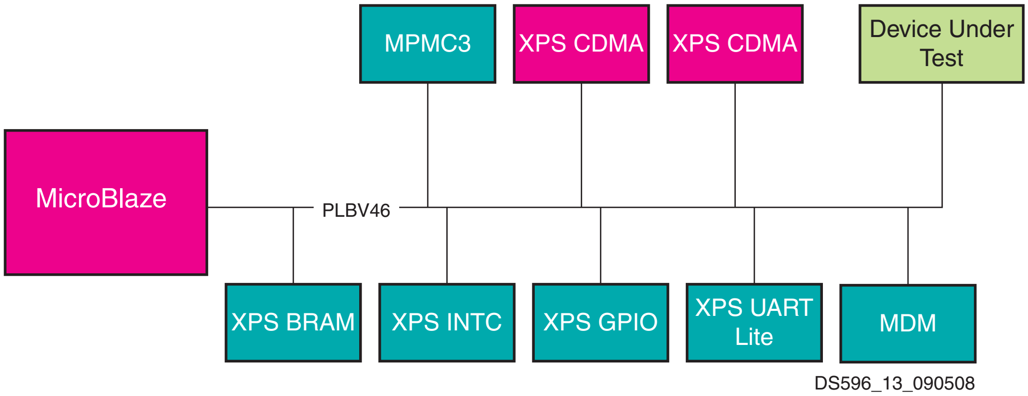


Figure 13: **Spartan-3A FPGA System**

The target FPGA was then filled with logic to drive the LUT and BRAM utilization to approximately 70% and the I/O utilization to approximately 80%. Using the default tool options and the slowest speed grade for the target FPGA, the resulting target F_{Max} numbers are shown in Table 8.

Table 8: **XPS BRAM Interface Controller System Performance**

Target FPGA	Target F_{MAX} (MHz)
S3A700 -4	90
V4FX60 -10	100
V5LXT50 -1	120

The target F_{Max} is influenced by the exact system and is provided for guidance. It is not a guaranteed value across all systems.

Specification Exceptions

This design does not support the following PLB V4.6 specification features for slave devices:

- Address pipelining
- Parity
- Indeterminate length bursts
- Cacheline 16 requests
- Fixed length burst requests longer than 16 data beats

Reference Documents

1. IBM 128-bit Processor Local Bus, Architecture Specification, Version 4.6

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.*

Ordering Information

This Xilinx LogiCORE IP module is provided at no additional cost with the Xilinx ISE Design Suite Embedded Edition software under the terms of the Xilinx End User License. The core is generated using the Xilinx ISE Embedded Edition software (EDK).

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 modules and software, please contact your local Xilinx sales representative.

Revision History

Date	Version	Revision
11/16/06	1.0	Initial Xilinx release.
02/28/07	1.1	Updated timing diagrams for reduced latency operation, removed references to Cacheline 16 support, added resource utilization numbers, Pt to Pt Mode operation description updated, added IBM spec reference.
8/09/07	1.2	In Table 2, C_SPLB_SUPPORT_BURSTS Default Value was 0.
9/26/07	1.3	Added F_{Max} Margin <RD Red>System Performance section.
4/17/08	1.4	Added Automotive Spartan-3A FPGA support.
7/25/08	1.5	Added QPro Virtex-4 Hi-Rel and QPro Virtex-4 Rad Tolerant FPGA support.
9/8/08	1.6	Updated core version to v1_00_b
11/25/08	1.7	Converted to current DS template; added PDF properties; updated supported device families list; updated images 11,12, and 13; updated legal matter.
4/24/09	1.8	Replaced references to supported device families and tool name(s) with hyperlink to PDF file.
3/1/11	1.9	Updated to for 13.1 release; updated Supported BRAM Memory tables to include data for Spartan-6 and Virtex-6 FPGAs; updated Design Tools versions.

Notice of Disclaimer

Xilinx is providing this design, code, or information (collectively, the “Information”) to you “**AS-IS**” with no warranty of any kind, express or implied. Xilinx makes no representation that the Information, or any particular implementation thereof, is free from any claims of infringement. You are responsible for obtaining any rights you may require for any implementation based on the Information. All specifications are subject to change without notice. XILINX EXPRESSLY DISCLAIMS ANY WARRANTY WHATSOEVER WITH RESPECT TO THE ADEQUACY OF THE INFORMATION OR ANY IMPLEMENTATION BASED THEREON, INCLUDING BUT NOT LIMITED TO ANY WARRANTIES OR REPRESENTATIONS THAT THIS IMPLEMENTATION IS FREE FROM CLAIMS OF INFRINGEMENT AND ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Except as stated herein, none of the Information may be copied, reproduced, distributed, republished, downloaded, displayed, posted, or transmitted in any form or by any means including, but not limited to, electronic, mechanical, photocopying, recording, or otherwise, without the prior written consent of Xilinx.