LogiCORE IP AXI Block RAM (BRAM) Controller (v1.03a)

DS777 July 25, 2012 **Product Specification**

Introduction

The LogiCORE™ IP AXI Block RAM (BRAM) Controller is a soft IP core for use with the Xilinx Vivado™ Design Suite, Embedded Development Kit (EDK), and ISE™ Design Suite. The core is designed as an AXI Endpoint slave IP for integration with the AXI interconnect and system master devices to communicate to local block RAM. The core supports both single and burst transactions to the block RAM and is optimized for performance.

Features

- AXI4 (memory mapped) slave interface
- Low latency memory controller
- Separate read and write channel interfaces to utilize dual port FPGA BRAM technology
- Option to arbitrate read and write data for use with a single port of block RAM (in AXI4 and AXI4LITE modes)
- Configurable BRAM data width (32-, 64-, 128-, 256-, 512-, and 1024-bit) (equals AXI slave port data width size)
- Supports INCR burst sizes up to 256 data transfers
- Supports WRAP bursts of 2, 4, 8, and 16 data beats
- Supports AXI narrow and unaligned write burst transfers
- Compatible with Xilinx AXI Interconnect
- Supports two-deep address pipelining on each read and write channel (order must be maintained)
- Reduced footprint option for AXI4-Lite
- Optional ECC support with 32-, 64- or 128-bit BRAM data widths
  - AXI4-Lite register interface for status and control of ECC operations
  - Available interrupt output signal for ECC error detection

LogiCORE IP Facts Table	
Core Specifics	
Supported Device Family(1)	Zynq™-7000(2), Virtex®-7, Kintex™-7, Artix™-7, Virtex-6, Spartan™-6
Supported User Interfaces	AXI4, AXI4-Lite
Resources Used	
See Table 69 and Table 70.	
Provided with Core	
Documentation	Product Specification
Design Files	VHDL
Example Design	Not Provided
Test Bench	Not Provided
Constraints File	Not Provided
Simulation Model	N/A
Supported S/W Driver(3)	Standalone
Tested Design Tools(4)	
Design Entry Tools	Vivado™ Design Suite v2012.2(5) ISE™ Design Suite and EDK v14.2
Simulation	Mentor Graphics® ModelSim PE/SE
Synthesis Tools	Vivado Synthesis XST
Support	
Provided by Xilinx @ www.xilinx.com/support	

Notes:

1. For a complete listing of supported derivative devices, see 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 a listing of the supported tool versions, see the Xilinx Design Tools: Release Notes Guide.
5. Supports only 7 series devices.

Functional Description

Figure 1 and Figure 2 illustrate the top-level port connections and main modules of the AXI BRAM controller IP core. Several instantiation modes exist based on interface connection and BRAM module port utilization. The AXI BRAM controller core can be configured such that a single port to the BRAM block or both ports to the BRAM block are utilized in either an AXI4 or AXI4-Lite controller configuration. The AXI BRAM controller IP can be configured with ECC functionality on the datapath with an available external ECC register set via a second AXI4-Lite control port connection.

Figure 1 illustrates the connections when the AXI BRAM core is configured for an AXI4-Lite mode to the BRAM block. A single port utilization to the BRAM block or a dual port mode to the BRAM block can be utilized (via a parameter setting). Figure 2 illustrates the generated HDL core for supporting a AXI4 transaction interface. Single port usage to the BRAM block can be configured over an enhanced performance setting in a dual port configuration.

All communication with AXI master devices is performed through a five channel AXI interface. All write operations are initiated on the Write Address Channel (AW) of the AXI bus which specifies the type of write transaction and the corresponding address information. The Write Data Channel (W) communicates all write data for the single or burst write operations. The Write Response Channel (B) is used as the handshaking or response on the write operation.

On read operations, the Read Address Channel (AR) communicates all address and control information when the AXI master requests a read transfer. The AXI slave BRAM controller IP responds on the Read Address Channel (AR) when the read operation can be processed. When the read data is available to send back to the AXI master, the Read Data Channel (R) translates the data and status of the operation.

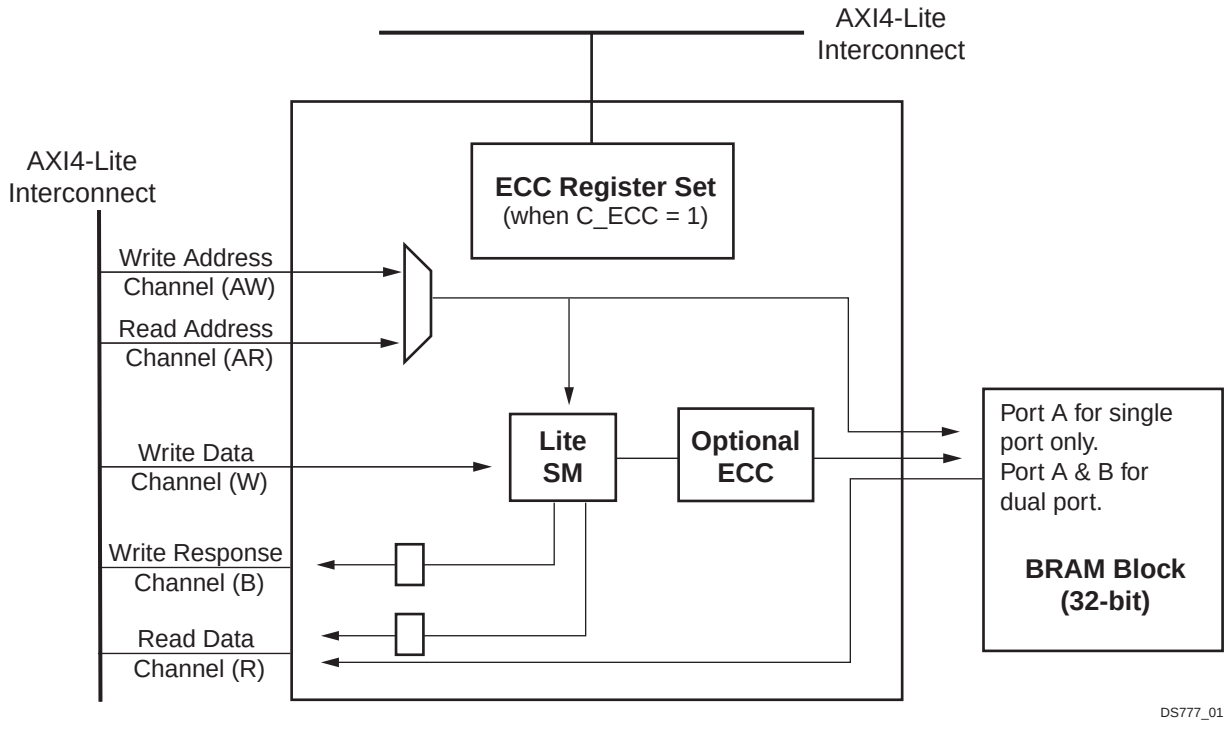

Figure 1: **AXI4-Lite BRAM Controller Block Diagram**

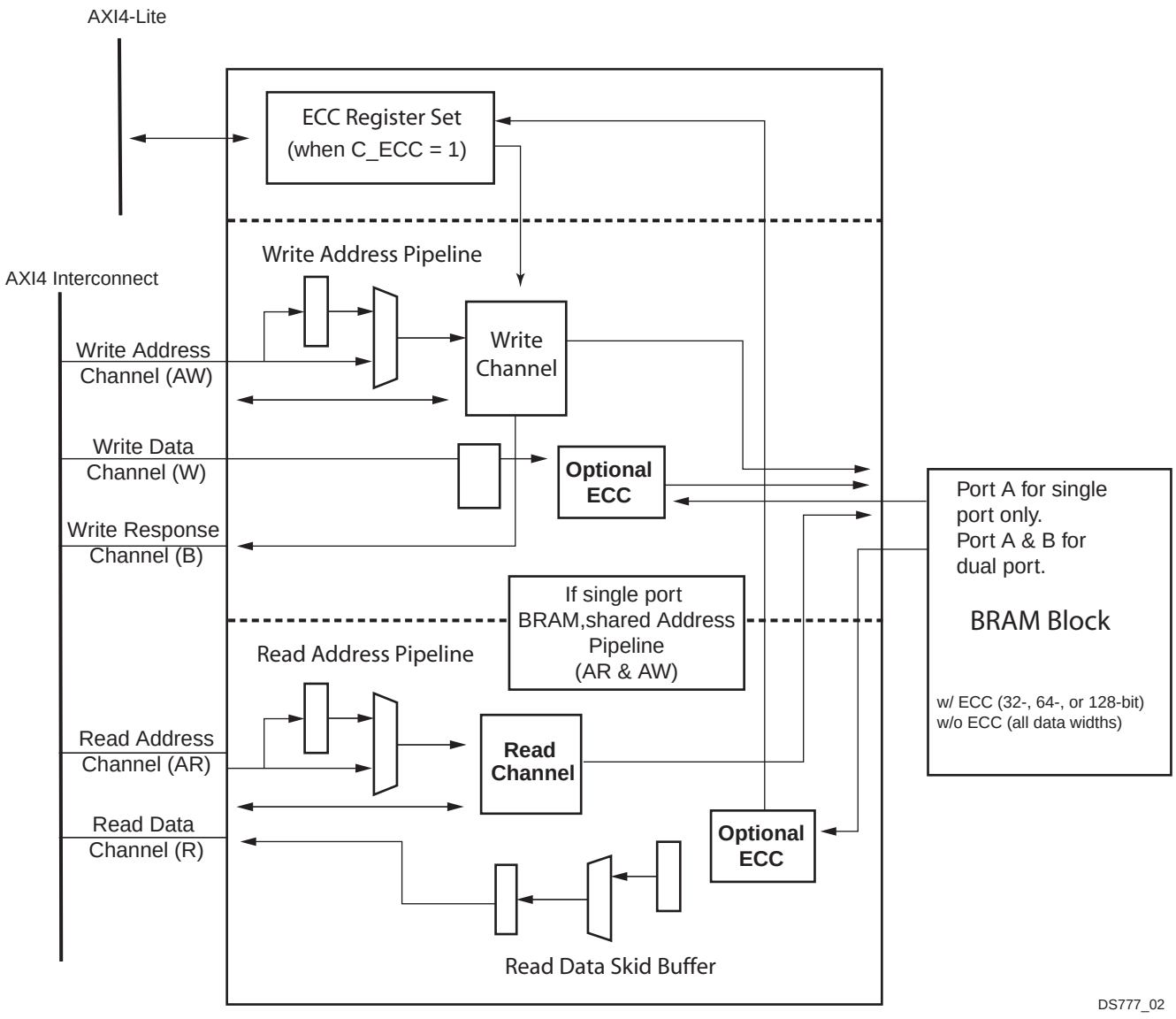

Figure 2: **AXI4 BRAM Controller Block Diagram**

ECC Description

The system designer can enable ECC functionality in the AXI BRAM controller IP core. ECC allows an AXI master to detect and correct single and detect double bit errors in the BRAM block. The ECC status and control is accessible via an additional AXI4-Lite control interface on the AXI BRAM controller core. The ECC functionality can be enabled regardless of dual or single port BRAM access. ECC is enabled by configuring the design parameter, C_ECC = 1. The ECC is available (in this release) only when the BRAM block is configured to a 32-, 64-, or 128-bit data width.

AXI4 Compatibility

The AXI BRAM controller IP core is compliant to the AMBA® AXI4 interface specifications listed in the Reference Documents section. The AXI BRAM controller core includes the subsequent features and exceptions.

- Support for 32-, 64-, 128-, 256-, 512-, and 1024-bit BRAM data widths
- Support for all AXI4 burst types and sizes
  - AXI BRAM controller handles FIXED bursts as INCR type burst operations (no FIFO burst type capability in the BRAM core)
  - 16 beats for WRAP bursts
  - 16 beats for FIXED bursts (treated as INCR burst type)
  - 256 beats for INCR burst (without exclusive access)
- Support for burst sizes that are less than the width of the block RAM, for example, *narrow* bursts. Data transfers are on different byte lanes for each beat of the burst.
- AXI user signals are not necessary or supported
- The AXI BRAM controller executes all transactions in order regardless of thread ID value. No read reordering or write reordering is implemented.
- No caching or buffering functionality is supported in the AXI BRAM controller

AXI4-Lite Support

As per the AXI specification, the AXI BRAM controller supports all requests from an AXI4-Lite master or AXI4-Lite Interconnect. The core can be configured for optimized FPGA resource usage and BRAM port utilization in this mode. The AXI4-Lite mode only supports a 32-bit AXI data bus width and single data beat transfers. No unaligned, narrow, or burst types transfers are acknowledged by the AXI BRAM controller IP core when configured in this mode.

The AXI4-Lite IP core can be configured as dual port to the block RAM or configured to arbitrate to a single port of block RAM at the AR and AW AXI interfaces. In this mode (C_SINGLE_PORT_BRAM = 1), only one active operation is allowed at a time in the BRAM controller core.

BRAM Interface

The BRAM interface on the AXI BRAM controller IP core is optimized to provide the highest performance interface to the FPGA BRAM module.

The dual-port capability of the Xilinx BRAM technology is utilized in this design (when configured in dual-port mode). Port A of the BRAM module is designated as the write port, while Port B of the BRAM module is designated as the read port.

For decreased size, the BRAM controller can be configured in a single port utilization to the BRAM module. In this case, Port A is used for both read and write operations to block RAM.

To reduce resource utilization and limit any potential impacts to latency, any collisions between the read and write ports of the block RAM are not detected by the BRAM controller.

Systems using 7 series and Virtex-6 FPGAs that are generated in EDK have a bank of block RAM based on incorporating the RAMB18E1 or the RAMB36E1. The block RAM is configured based on the selected data width desired for the block RAM.

For Spartan®-6 FPGA designs, the BRAM block incorporates the RAMB16BWER and the RAMB8BWER as the basic building blocks for all BRAM configurations. The optional internal pipeline register on the output datapath is utilized to increase the performance in Spartan-6 FPGA system topologies.

The BRAM block instantiation is NOT included in the AXI BRAM controller IP in an EDK system topology. The BRAM block is generated as part of the EDK XPS tools when the embedded system is created. The address parameters for the block RAM are analyzed and the appropriate BRAM components are instantiated by the EDK tools.

Instantiating the BRAM controller from the CORE Generator tool as a stand-alone IP core requires the BRAM components be instantiated separately with the Block Memory Generator tool.

ECC Support

ECC support can be enabled in the AXI BRAM Controller IP core by setting the design parameter, C_ECC = 1. ECC is only supported for BRAM data widths of 32-, 64-, or 128-bit configurations. More information is described in ECC, page 19.

AXI System Topologies

The AXI BRAM controller core is designed to integrate in an AXI system via the AXI Interconnect topology to provide multiple masters access to block RAM. The AXI BRAM controller is an endpoint slave IP core to be attached as a slave AXI device on the AXI Interconnect. The AXI Interconnect allows the BRAM controller to be optimized for latency and reduce the resources utilized. The AXI Interconnect performs any mismatching of master data widths to the size of the block RAM and the BRAM controller core. It is recommended that the AXI BRAM Controller, AXI4 interface data width be configured to match the native width of the AXI Interconnect to which it is attached. The AXI Interconnect performs the decode of the master address prior to presenting any operation to the BRAM controller core. Additional information on the Xilinx AXI Interconnect is available in Reference Documents, page 66.

The connection of the AXI BRAM controller core into an example system topology is shown in Figure 3 for both single and dual port BRAM module configurations.

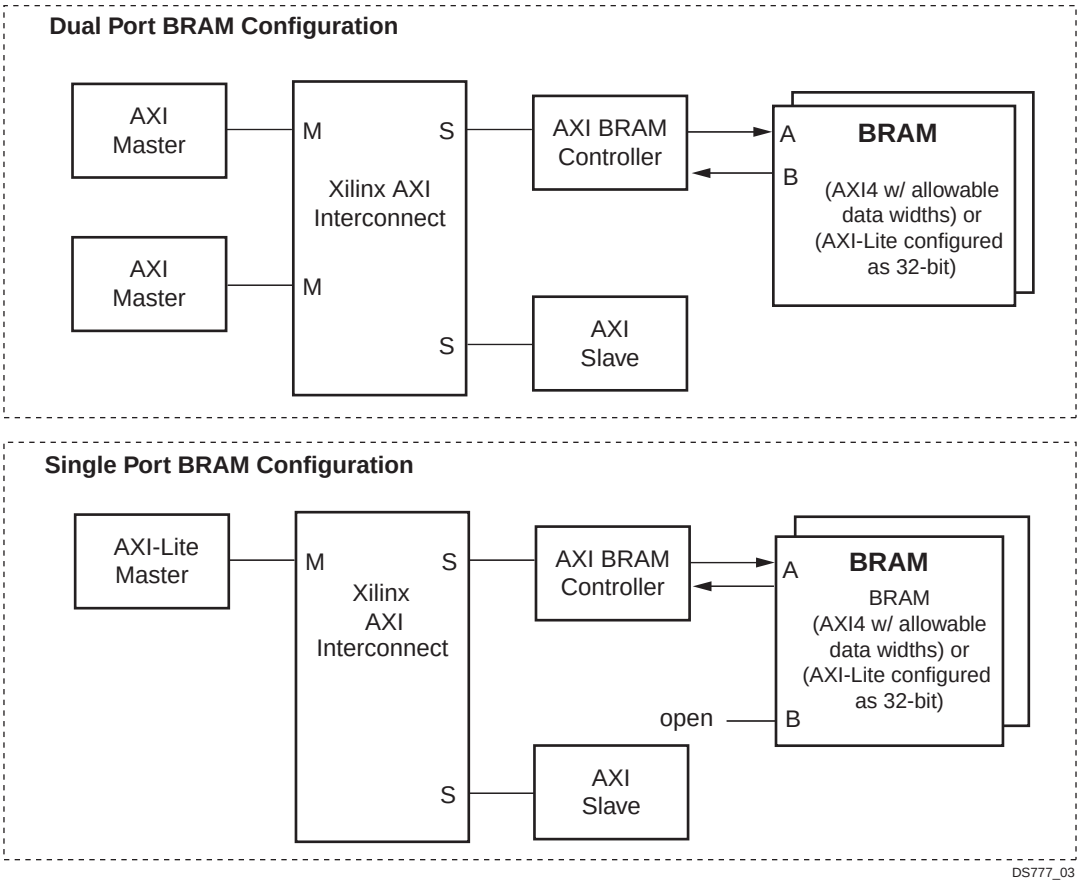

Figure 3: **AXI System Configuration**

I/O Signals

The AXI BRAM Controller I/O signals are listed and described in Table 1.

Table 1: **I/O Signal Description**

Signal Name	Interface	Signal Type	Init Status	Description
Global Signals				
S_AXI_ACLK	S_AXI, S_AXI_CTRL	I		AXI Bus Clock.
S_AXI_ARESETN	S_AXI, S_AXI_CTRL	I		AXI active low reset.
ECC_Interrupt		O	'0'	Interrupt to signal ECC error condition: Signal unused when C_ECC = 0.
ECC_UE		O	'0'	ECC uncorrectable error output flag: The behavior of this flag is described in section, ECC, page 19. Signal unused when C_ECC = 0.
AXI Write Address Channel Signals (AW)				
S_AXI_AWID [C_S_AXI_ID_WIDTH-1:0]	S_AXI (AW)	I		AXI address write ID.

Table 1: **I/O Signal Description *(Cont'd)***

Signal Name	Interface	Signal Type	Init Status	Description
S_AXI_AWADDR [C_S_AXI_ADDR_WIDTH-1:0]	S_AXI (AW)	I		AXI write address
S_AXI_AWLEN [7:0]	S_AXI (AW)	I		AXI address write burst length: The burst length gives the exact number of transfers in a burst. Determines the number of data transfers associated with the write address.
S_AXI_AWSIZE [2:0]	S_AXI (AW)	I		AXI address write burst size: Indicates the size of each transfer in the burst. Byte lane strobes determine exactly which byte lanes to update.
S_AXI_AWBURST [1:0]	S_AXI (AW)	I		AXI address write burst type: Indicates the burst type, coupled with the size information, details how the address for each transfer within the burst is calculated.
S_AXI_AWLOCK [2:0]	S_AXI (AW)	I		AXI write address lock signal: Provides information about atomic operation transfers and barrier transactions. Unused at this time, but listed here for future implementation and support.
S_AXI_AWCACHE [4:0]	S_AXI (AW)	I		AXI write address cache control signal: Provides information about bufferable, cacheable, and allocation attributes. Unused at this time, but listed here for future implementation and support.
S_AXI_AWPROT [3:0]	S_AXI (AW)	I		AXI write address protection signal. Unused at this time, but listed here for future implementation and support.
S_AXI_AWVALID	S_AXI (AW)	I		AXI write address valid. Indicates that the valid write address and control information is available. '0' = address and control NOT available '1' = address and control data is available The address and control information remains stable until the acknowledge signal (S_AXI_AWREADY) is asserted (active high).
S_AXI_AWREADY	S_AXI (AW)	O	'0'	AXI write address ready. Indicates that the BRAM controller has accepted the write channel address and associated control signals. '0' = BRAM controller not ready '1' = BRAM controller is ready
AXI Write Data Channel Signals (W)				
S_AXI_WDATA [C_S_AXI_DATA_WIDTH-1:0]	S_AXI (W)	I		AXI write data.
S_AXI_WSTRB [C_S_AXI_DATA_WIDTH/8-1:0]	S_AXI (W)	I		AXI write data strobes. Indicates which bytes lanes to update in block RAM. Each byte strobe correlates to a byte other write data bus. WSTRB[n] corresponds to WDATA[(8*n) + 7:(8*n)].
S_AXI_WLAST	S_AXI (W)	I		AXI write data last signal. Indicates the last transfer in a write burst.

Table 1: **I/O Signal Description *(Cont'd)***

Signal Name	Interface	Signal Type	Init Status	Description
S_AXI_WVALID	S_AXI (W)	I		AXI write data valid. Indicates that valid write data and strobes are available on the write data channel. '0' = write data and strobes unavailable '1' = write data and strobes available and valid
S_AXI_WREADY	S_AXI (W)	O	'0'	AXI write data ready. indicates that the BRAM controller is ready to accept the write data and strobes. '0' = BRAM controller not ready '1' = BRAM controller is ready
AXI Write Response Channel Signals (B)				
S_AXI_BID [C_S_AXI_ID_WIDTH-1:0]	S_AXI (B)	O	Zeros	AXI write data response ID. Identification tab of the write response. BID matches the AWID value of the write transaction to which the BRAM controller is responding.
S_AXI_BRESP [1:0]	S_AXI (B)	O	Zeros	AXI write response. Indicates the status of the write transaction. The supported responses of the BRAM controller are OKAY (SLVERR and EXOKAY to be supported in a future release).
S_AXI_BVALID	S_AXI (B)	O	'0'	AXI write response valid. Asserted by the BRAM controller to indicate a valid write response is available. '0' = write response not available '1' = write response is available
S_AXI_BREADY	S_AXI (B)	I		Write response ready. Indicates the master requesting the write operation can accept the response information. '0' = AXI master not ready '1' = AXI master is ready
AXI Read Address Channel Signals (AR)				
S_AXI_ARID [C_S_AXI_ID_WIDTH-1:0]	S_AXI (AR)	I		AXI address read ID. Identification tag for the read address group of signals.
S_AXI_ARADDR [C_S_AXI_ADDR_WIDTH-1:0]	S_AXI (AR)	I		AXI read address. The address provides the initial address of the read burst transaction. Only the start address of the burst is provided and the control signals indicate the address detail and calculation for each transfer in the burst.
S_AXI_ARLEN [7:0]	S_AXI (AR)	I		AXI address read burst length. The burst length gives the exact number of transfers in a burst. Determines the number of data transfers associated with the read address.
S_AXI_ARSIZE [2:0]	S_AXI (AR)	I		AXI address read burst size. Indicates the size of each transfer in the burst.
S_AXI_ARBURST [1:0]	S_AXI (AR)	I		AXI address read burst type. Indicates the burst type, coupled with the size information, details how the address for each transfer within the burst is calculated.
S_AXI_ARLOCK [2:0]	S_AXI (AR)	I		AXI read address lock signal. Provides information about atomic operation transfers. Unused at this time, but listed here for future implementation and support.

Table 1: **I/O Signal Description *(Cont'd)***

Signal Name	Interface	Signal Type	Init Status	Description
S_AXI_ARCACHE [4:0]	S_AXI (AR)	I		AXI read address cache control signal. Provides information about bufferable, cacheable, and allocation attributes. Unused at this time, but listed here for future implementation and support.
S_AXI_ARPROT [3:0]	S_AXI (AR)	I		AXI read address protection signal. Unused at this time, but listed here for future implementation and support.
S_AXI_ARVALID	S_AXI (AR)	I		AXI read address valid. Indicates that the valid read address and control information is available. '0' = address and control NOT available '1' = address and control data is available The address and control information remains stable until the acknowledge signal (S_AXI_ARREADY) is asserted (active high).
S_AXI_ARREADY	S_AXI (AR)	O	'0'	AXI read address ready. Indicates that the BRAM controller has accepted the read channel address and associated control signals. '0' = BRAM controller not ready '1' = BRAM controller is ready
AXI Read Data Channel Signals (R)				
S_AXI_RID [C_S_AXI_ID_WIDTH-1:0]	S_AXI (R)	O	Zeros	AXI read data response ID. Identification tab of the read response. RID matches the ARID value of the read address transaction to which the BRAM controller is responding.
S_AXI_RDATA [C_S_AXI_DATA_WIDTH-1:0]	S_AXI (R)	O	Zeros	AXI read data.
S_AXI_RRESP [1:0]	S_AXI (R)	O	Zeros	***Note:*** AXI read response. Indicates the status of the read transfer. AXI BRAM controller supports the OKAY and SLVERR responses.
S_AXI_RLAST	S_AXI (R)	O	'0'	AXI read data last signal. Indicates the last transfer in a read burst.
S_AXI_RVALID	S_AXI (R)	O	'0'	AXI read valid. Asserted by the BRAM controller to indicate the required read data is available and the read transfer can complete. '0' = read data not available '1' = read data is available
S_AXI_RREADY	S_AXI (R)	I		Read ready. Indicates the requesting master can accept the read data and response information. '0' = AXI master not ready '1' = AXI master is ready
AXI4-Lite Control Signals (only available when C_ECC = 1)				
S_AXI_CTRL_AWADDR [C_S_AXI_CTRL_ADDR_ WIDTH-1:0]	S_AXI_CTRL (AW)	I		AXI4-Lite control write address

Table 1: **I/O Signal Description *(Cont'd)***

Signal Name	Interface	Signal Type	Init Status	Description
S_AXI_CTRL_AWVALID	S_AXI_CTRL (AW)	I		AXI4-Lite control write address valid. Indicates that the valid write address and control information is available. '0' = address and control NOT available '1' = address and control data is available The address and control information remains stable until the acknowledge signal (S_AXI_LITE_AWREADY) is asserted (active high).
S_AXI_CTRL_AWREADY	S_AXI_CTRL (AW)	O	'0'	AXI4-Lite control write address ready. Indicates that the BRAM controller has accepted the write channel address and associated control signals. '0' = BRAM controller not ready '1' = BRAM controller is ready
S_AXI_CTRL_WDATA [C_S_AXI_CTRL_DATA_ WIDTH-1:0]	S_AXI_CTRL (W)	I		AXI4-Lite control write data.
S_AXI_CTRL_WVALID	S_AXI_CTRL (W)	I		AXI4-Lite write data valid. Indicates that valid write data and strobes are available on the write data channel. '0' = write data and strobes unavailable '1' = write data and strobes available and valid
S_AXI_CTRL_WREADY	S_AXI_CTRL (W)	O	'0'	AXI4-Lite control write data ready. indicates that the BRAM controller is ready to accept the write data and strobes. '0' = BRAM controller not ready '1' = BRAM controller is ready
S_AXI_CTRL_BRESP [1:0]	S_AXI_CTRL (B)	O	Zeros	AXI4-Lite control response. Indicates the status of the write transaction. The supported responses of the BRAM controller are OKAY & SLVERR.
S_AXI_CTRL_BVALID	S_AXI_CTRL (B)	O	'0'	AXI4-Lite control write response valid. Asserted by the BRAM controller to indicate a valid write response is available. '0' = write response not available '1' = write response is available
S_AXI_CTRL_BREADY	S_AXI_CTRL (B)	I		AXI4-Lite control write response ready. Indicates the master requesting the write operation can accept the response information. '0' = AXI master not ready '1' = AXI master is ready
S_AXI_CTRL_ARADDR [C_S_AXI_CTRL_ADDR_ WIDTH-1:0]	S_AXI_CTRL (AR)	I		AXI4-Lite control read address information.
S_AXI_CTRL_ARVALID	S_AXI_CTRL (AR)	I		AXI4-Lite control read address valid. Indicates that the valid read address and control information is available. '0' = address and control NOT available '1' = address and control data is available The address and control information remains stable until the acknowledge signal (S_AXI_LITE_ARREADY) is asserted (active high).

Table 1: **I/O Signal Description *(Cont'd)***

Signal Name	Interface	Signal Type	Init Status	Description
S_AXI_CTRL_ARREADY	S_AXI_CTRL (AR)	O	'0'	AXI4-Lite control read address ready. Indicates that the BRAM controller has accepted the read channel address and associated control signals. '0' = BRAM controller not ready '1' = BRAM controller is ready
S_AXI_CTRL_RDATA [C_S_AXI_CTRL_DATA_ WIDTH-1:0]	S_AXI_CTRL (R)	O	Zeros	AXI4-Lite control read data.
S_AXI_CTRL_RRESP [1:0]	S_AXI_CTRL (R)	O	Zeros	AXI4-Lite control read response. Indicates the status of the read transfer. AXI BRAM controller supports the OKAY and SLVERR responses.
S_AXI_CTRL_RVALID	S_AXI_CTRL (R)	O	'0'	AXI4-Lite control read valid. Asserted by the BRAM controller to indicate the required read data is available and the read transfer can complete. '0' = read data not available '1' = read data is available
S_AXI_CTRL_RREADY	S_AXI_CTRL (R)	I		AXI4-Lite control read ready. Indicates the requesting master can accept the read data and response information. '0' = AXI master not ready '1' = AXI master is ready
BRAM Signals				
BRAM_Rst_A	BRAM	O	'0'	Port A BRAM active High reset.
BRAM_Clk_A	BRAM	O	'0'	Port A BRAM clock. Connected to AClk with same frequency, same phase alignment.
BRAM_En_A	BRAM	O	'0'	BRAM Port A (write port) enable signal. Active High.
BRAM_WE_A [(C_S_AXI_DATA_WIDTH/8) + C_ECC - 1:0]	BRAM	O	Zeros	BRAM Port A (write port) write enable signal. Active High. Byte wise signal to width of block RAM. If ECC is enabled, the BRAM_WE_A width increases to hold the ECC bits as shown in ECC Data Sizes, page 20.
BRAM_Addr_A [C_BRAM_ADDR_WIDTH-1:0]	BRAM	O	Zeros	BRAM Port A (write port) address bus. Bus is sized according to data width and based on C_S_AXI_BASEADDR and C_S_AXI_HIGHADDR settings.
BRAM_WrData_A [(8 * C_ECC + C_S_AXI_DATA_WIDTH)-1:0]	BRAM	O	Zeros	BRAM Port A (write port) write data bus. Size of BRAM write data width is equal to size of AXI slave port connection to the BRAM controller. If ECC is enabled, the BRAM_WrData_A width increases to hold the ECC bits as shown in Table 10, page 20.

Table 1: **I/O Signal Description *(Cont'd)***

Signal Name	Interface	Signal Type	Init Status	Description
BRAM_RdData_A [(8 * C_ECC + C_S_AXI_DATA_WIDTH)-1:0]	BRAM	I		BRAM Port A (read port) read data bus. Size of BRAM read data width is equal to size of AXI slave port connection to the BRAM controller. If ECC is enabled, the BRAM_RdData_A width increases to hold the ECC bits as shown in Table 10, page 20. BRAM_RdData_A port used during the read-modify-write (if needed) when C_ECC = 1. BRAM_RdData_A port also used for read operations when C_SINGLE_PORT_BRAM = 1.
BRAM_Rst_B	BRAM	O	'0'	Port B BRAM active High reset.
BRAM_Clk_B	BRAM	O	'0'	Port B BRAM clock. Connected to AClk with same frequency, same phase alignment.
BRAM_En_B	BRAM	O	Zeros	BRAM Port B (read port) enable signal. Active High. Only used when C_SINGLE_PORT_BRAM = 0.
BRAM_WE_B [(C_S_AXI_DATA_WIDTH/8) + C_ECC - 1:0] (2)	BRAM	O	Zeros	BRAM Port B (read port) write enable signal. Active High. *Fixed to default zeros value (port B is a read-only port to block RAM).* Size is byte-wise to width of block RAM. If ECC is enabled, the BRAM_WE_A width increases to hold the ECC bits as shown in Table 10, page 20.
BRAM_Addr_B [C_BRAM_ADDR_WIDTH-1:0]	BRAM	O	Zeros	BRAM Port B (read port) address bus. Bus is sized according to data width and based on C_S_AXI_BASEADDR and C_S_AXI_HIGHADDR settings. Only used when C_SINGLE_PORT_BRAM = 0.
BRAM_RdData_B [(8 * C_ECC + C_S_AXI_DATA_WIDTH)-1:0]	BRAM	I		BRAM Port B (read port) read data bus. Size of BRAM read data width is equal to size of AXI slave port connection to the BRAM controller. If ECC is enabled, the BRAM_RdData_B width increases to hold the ECC bits as shown in Table 10, page 20. Only used when C_SINGLE_PORT_BRAM = 0.
BRAM_WrData_B [(8 * C_ECC + C_S_AXI_DATA_WIDTH-1:0] (2)	BRAM	O	Zeros	BRAM Port B (read port) write data bus. *Fixed to default zeros value (port B is a read-only port to block RAM).* Size of BRAM write data width is equal to size of AXI slave port connection to the AXI BRAM Controller. If ECC is enabled, the BRAM_WrData_B width increases to match the data width with ECC bits as shown in Table 10, page 20.

Notes:

1. This AXI IP slave core does not include support for the low power interface signals of the AXI defined specification.
2. BRAM_WE_B and BRAM_WrData_B are unused port signals in the AXI BRAM controller IP core. These signals are included in the BRAM interface for tool compatibility (and the default values are driven properly).

Parameters

To allow the user to create an AXI BRAM controller that is uniquely tailored for their system, certain features can be parameterized in the AXI BRAM controller design. This allows the configuration of a design that utilizes only the resources required by the system and that operates with the best possible performance. The features that can be parameterized in the AXI BRAM controller design are shown in Table 2.

Inferred Parameters

In addition to the parameters listed in Table 2, there are also parameters that are inferred for each AXI interface in the EDK tools. Through the design, these EDK-inferred parameters control the behavior of the AXI Interconnect. For a complete list of the interconnect settings related to the AXI interface, see DS768, AXI Interconnect IP data sheet (Reference Documents, page 66). Some parameters are described in Table 2.

Table 2: **Design Parameters**

Feature/Description	Parameter Name	Allowable Values	Default Values	VHDL Type
AXI IF Parameters				
AXI protocol connection type	C_S_AXI_PROTOCOL	AXI4, AXI4LITE	AXI4	string
Width of ID vectors in AXI system (includes all master and slave devices)	C_S_AXI_ID_WIDTH	1-16	4	integer
Data width of AXI slave (BRAM data width)	C_S_AXI_DATA_WIDTH	32, 64, 128, 256, 512, or 1024	32	Integer
Width of AXI address bus	C_S_AXI_ADDR_WIDTH	32	32	integer
Support of AXI narrow write or read operations. Valid only when connected to the AXI4 interface.	C_S_AXI_SUPPORTS_NARROW_BURST	0 = No support for narrow AXI transactions 1 = Core includes logic to support narrow transfers	1	integer
BRAM Parameters				
Selects BRAM port utilization (dual or single)	C_SINGLE_PORT_BRAM	0 = Dual port BRAM 1 = Single port BRAM	0	integer
ECC Parameters				
Enable ECC functionality with single bit error correction and double bit error detection	C_ECC	0 = No ECC 1 = ECC logic enabled in datapath + ECC AXI-Lite register interface is accessible	0	integer
AXI4-Lite ECC control register address width	C_S_AXI_CTRL_ADDR_WIDTH	32	32	integer
AXI4-Lite ECC control data width	C_S_AXI_CTRL_DATA_WIDTH	32	32	integer
Implement ECC fault injection registers (4)	C_FAULT_INJECT (4)	0 = No fault injection registers 1 = Fault injection registers enable	0	integer

Table 2: **Design Parameters *(Cont'd)***

Feature/Description	Parameter Name	Allowable Values	Default Values	VHDL Type
Controls ECC on/off value at startup/reset (4)	C_ECC_ONOFF_RESET_VALUE (4)	0 = ECC is not enabled at reset (the block RAM is not initialized with ECC) 1 = ECC is enabled at reset; the ECC is initialized upon start-up	1	integer
Non HDL Parameters (Not included in RTL, but integrated as a part of the core during the EDK build process to allow for AXI system optimizations)				
AXI BRAM base address assignment	C_S_AXI_ BASEADDR (See Note 2)	System address value of C_AWIDTH bits wide	FFFF_FFFF	std_logic _vector
AXI BRAM high address assignment	C_S_AXI_ HIGHADDR (2)	System address value of C_AWIDTH bits wide	0000_0000	std_logic _vector
AXI4-Lite ECC control register interface protocol	C_S_AXI_CTRL_PROTOCOL	AXI4LITE	AXI4LITE	string
AXI4-Lite ECC control base address assignment	C_S_AXI_CTRL_ BASEADDR (2)(4)	System address value of C_AWIDTH bits wide	FFFF_FFFF	std_logic _vector
AXI4-Lite ECC control register high address assignment	C_S_AXI_CTRL_ HIGHADDR (2)(4)	System address value of C_AWIDTH bits wide	0000_0000	std_logic _vector
AXI4-Lite control register interface supports read operations	C_S_AXI_CTRL_SUPPORTS_REA D	1 (constant in MPD for AXI Interconnect)	1	integer
AXI4-Lite control register interface supports write operations	C_S_AXI_CTRL_SUPPORTS_WRIT E	1 (constant in MPD for AXI Interconnect)	1	integer
Maximum number of active write transactions that the AXI slave can accept. (sets the parameter, WRITE_ISSUING in the AXI Interconnect)	C_INTERCONNECT_ S_AXI_WRITE_ACCEPTANCE	2 (set constant) for AXI4 1 (default) for AXI4LITE	1	integer
Maximum number of active read transactions that the AXI slave can accept.	C_INTERCONNECT_ S_AXI_READ_ACCEPTANCE	2 (set constant) for AXI4 1 (default) for AXI4LITE	1	integer
Indicates (for Interconnect) if AXI AR channel registers are enabled.	C_INTERCONNECT_ S_AXI_AR_REGISTER	0, 1	1	integer
Indicates (for Interconnect) if AXI AW channel registers are enabled.	C_INTERCONNECT_ S_AXI_AW_REGISTER	0, 1	1	integer
Indicates (for Interconnect) if AXI W channel registers are enabled.	C_INTERCONNECT_ S_AXI_W_REGISTER	0, 1	1	integer
Indicates (for Interconnect) if AXI B channel registers are enabled.	C_INTERCONNECT_ S_AXI_B_REGISTER	0, 1	1	integer
Indicates (for Interconnect) if AXI R channel registers are enabled.	C_INTERCONNECT_ S_AXI_R_REGISTER	0, 1	1	integer
Target FPGA Family Parameter				
Target FPGA device family	C_FAMILY	"spartan6", "virtex6", "virtex7", "kintex7", "artix7"	"virtex7"	string

Table 2: **Design Parameters *(Cont'd)***

Feature/Description	Parameter Name	Allowable Values	Default Values	VHDL Type
Calculated Parameters (auto calculated in HDL, included here for documentation purposes)				
Number of ECC bits on BRAM interface	C_NUM_ECC_BITS	7 = when 32-bit BRAM data width, or 8 = when 64-bit BRAM data width 16 = when 128-bit BRAM data width	7	integer

Notes:

1. Default values are specified for C_S_AXI_BASEADDR and C_S_AXI_HIGHADDR to ensure that they are set by the user. If the value is not set, an implementation error is generated.
2. C_S_AXI_BASEADDR and C_S_AXI_HIGHADDR must be a power of 2 and specify a continuous address space. The required minimum size = 0xFFF.
3. The parameter allows for specification of up to 250 MHz, but the core is rated only for the frequencies specified in Table 69 and Table 70.
4. Parameter value is don't care, unless parameter C_ECC = 1.

Parameter - I/O Signal Dependencies

Table 3: **Parameter-I/O Signal Dependencies**

Generic Name	Affects Port	Depends on Parameter	Relationship Description
C_S_AXI_ID_WIDTH	S_AXI_AWID, S_AXI_BID, S_AXI_ARID, S_AXI_RID		Port widths are set directly from the parameter value.
C_S_AXI_DATA_WIDTH	S_AXI_WDATA, S_AXI_WSTRB, S_AXI_RDATA, BRAM_WE_A, BRAM_WrData_A, BRAM_RdData_A, BRAM_RdData_B		Port widths are set directly from the parameter value.
C_S_AXI_ADDR_WIDTH	S_AXI_AWADDR, S_AXI_ARADDR, BRAM_Addr_A, BRAM_Addr_B		Port widths are set directly or derived from the parameter value.
C_S_AXI_SUPPORTS_NARROW_BURST		C_S_AXI_PROTOCOL	Narrow bursts can only be enabled when bus set for AXI4.
C_ECC	All AXI4-Lite I/O signals	C_S_AXI_DATA_WIDTH	AXI4-Lite control I/O signals are only available when C_ECC = 1. C_ECC can only be set to 1 when C_S_AXI_DATA_WIDTH = 32, 64 or 128.
C_S_AXI_CTRL_ADDR_WIDTH, C_S_AXI_CTRL_DATA_WIDTH, C_FAULT_INJECT, C_ECC_ONOFF_RESET_VALUE		C_ECC	Parameters are don't care unless C_ECC = 1.
C_ECC	ECC_Interrupt, ECC_UE		Output signal is unused by system when C_ECC = 0.

AXI4-Lite ECC Registers

Table 4 summarizes the ECC register set that is available when ECC is enabled (C_ECC = 1) on the AXI-Lite control interface. Additional details on each register is provided in ECC Register Details, page 25.

Table 4: **AXI BRAM Interface Register Address Map** (1)

Base Address + Offset (hex)	Register	Access Type	Description
C_S_AXI_CTRL_BASEADDR + 0x0	ECC_STATUS	R/W	ECC Status Register
C_S_AXI_CTRL_BASEADDR + 0x4	ECC_EN_IRQ	R/W	ECC Enable Interrupt Register
C_S_AXI_CTRL_BASEADDR + 0x8	ECC_ON_OFF	R/W	ECC On/Off Register
C_S_AXI_CTRL_BASEADDR + 0xC	CE_CNT	R/W	Correctable Error Counter Register
C_S_AXI_CTRL_BASEADDR + 0x100 (1)	CE_FFD [31:0]	R	Correctable Error First Failing Data Register, bits [31:0] *Unused in core. Reserved for future implementation.*
C_S_AXI_CTRL_BASEADDR + 0x104	CE_FFD [63:32]	R	Correctable Error First Failing Data Register, bits [63:32] *Unused in core. Reserved for future implementation.*
C_S_AXI_CTRL_BASEADDR + 0x108	CE_FFD [95:64]	R	Correctable Error First Failing Data Register, bits [95:64] *Unused in core. Reserved for future implementation.*
C_S_AXI_CTRL_BASEADDR + 0x10C	CE_FFD [127:96]	R	Correctable Error First Failing Data Register, bits [127:96] *Unused in core. Reserved for future implementation.*
(0x120 - 0x17C) Reserved for Expansion greater than 128-bit BRAM with ECC.			
C_BASEADDR + 0x180	CE_FFE	R	Correctable Error First Failing ECC Register *Unused in core. Reserved for future implementation.*
(0x184 - 0x1BC) Reserved for Expansion greater than 128-bit BRAM with ECC.			
C_S_AXI_CTRL_BASEADDR + 0x1C0	CE_FFA [31:0]	R	Correctable Error First Failing Address Register, bits [31:0]
C_S_AXI_CTRL_BASEADDR + 0x1C4	CE_FFA [63:32]	R	Correctable Error First Failing Address Register, bits [63:32]
C_S_AXI_CTRL_BASEADDR + 0x200	UE_FFD [31:0]	R	Uncorrectable Error First Failing Data Register, bits [31:0] *Unused in core. Reserved for future implementation.*
C_S_AXI_CTRL_BASEADDR + 0x204	UE_FFD [63:32]	R	Uncorrectable Error First Failing Data Register, bits [63:32] *Unused in core. Reserved for future implementation.*
C_S_AXI_CTRL_BASEADDR + 0x208	UE_FFD [95:64]	R	Uncorrectable Error First Failing Data Register, bits [95:64] *Unused in core. Reserved for future implementation.*
C_S_AXI_CTRL_BASEADDR + 0x20C	UE_FFD [127:96]	R	Uncorrectable Error First Failing Data Register, bits [127:96] *Unused in core. Reserved for future implementation.*

Table 4: **AXI BRAM Interface Register Address Map** (1)

Base Address + Offset (hex)	Register	Access Type	Description
(0x210 - 0x27C) Reserved for expansion greater than 128-bit BRAM with ECC.			
C_S_AXI_CTRL_BASEADDR + 0x280	UE_FFE	R	Uncorrectable Error First Failing ECC Register *Unused in core. Reserved for future implementation.*
(0x284 - 0x2BC) Reserved for expansion greater than 128-bit BRAM with ECC.			
C_S_AXI_CTRL_BASEADDR + 0x2C0	UE_FFA [31:0]	R	Uncorrectable Error First Failing Address Register, bits [31:0] *Unused in core. Reserved for future implementation.*
C_S_AXI_CTRL_BASEADDR + 0x2C4	UE_FFA [63:32]	R	Uncorrectable Error First Failing Address Register, bits [63:32] *Unused in core. Reserved for future implementation.*
C_S_AXI_CTRL_BASEADDR + 0x300	FI_D [31:0]	W	Fault Inject Data Register, bits [31:0]
C_S_AXI_CTRL_BASEADDR + 0x304	FI_D [63:32]	W	Fault Inject Data Register, bits [63:32]
C_S_AXI_CTRL_BASEADDR + 0x308	FI_D [95:64]	W	Fault Inject Data Register, bits [95:64]
C_S_AXI_CTRL_BASEADDR + 0x30C	FI_D [127:96]	W	Fault Inject Data Register, [127:96]
(0x310 - 0x37C) Reserved for expansion greater than 128-bit BRAM with ECC.			
C_S_AXI_CTRL_BASEADDR + 0x380	FI_ECC	W	Fault Inject ECC Register

Notes:

1. Grayed cells indicate registers that are not used in core, but that are reserved for future implementation.

All ECC registers and bit definitions are described in ECC Register Details, page 25.

Addressing Configurations

Table 5 illustrates some example settings for C_S_AXI_BASEADDR and C_S_AXI_HIGHADDR to create a specific size of block RAM in the system. The range specified by the base and high address parameters must be a continuous address space and be equal to 2^n bytes minus 1, where n is a positive integer and 2^n is a valid memory size for the block RAM available in the FPGA device family.

Table 5: **Example Address Ranges**

Memory Size (Bytes)	Basic Address Range Required	Example C_S_AXI_BASEADDR	Example C_S_AXI_HIGHADDR
4k	0x0000_0000 to 0x0000_0FFF	0xA000 0000	0xA000 0FFF
8k	0x0000_0000 to 0x0000_1FFF	0x2400 0000	0x2400 1FFF
16k	0x0000_0000 to 0x0000_3FFF	0x1F00 0000	0x1F00 3FFF
32k	0x0000_0000 to 0x0000_7FFF	0x3000 0000	0x3000 7FFF
64k	0x0000_0000 to 0x0000_FFFF	0xB000 0000	0xB000 FFFF
128k	0x0000_0000 to 0x0001_FFFF	0x2000 0000	0x2001 FFFF
256k	0x0000_0000 to 0x0003_FFFF	0xFFFC 0000	0xFFFF FFFF

Table 6 illustrates the BRAM configuration (and corresponding address assignment) for targeting Virtex-6 FPGA designs utilizing the 36 Kb BRAM module.

Table 6: **BRAM Configuration for Virtex-6 FPGAs**

Supported Memory Sizes / BRAM Memory Configuration	Number of BRAM Primitives (36k/each)	Size of BRAM_Addr (each port)	Typical BRAM_Addr Bit Usage with BRAM Configuration
32-bit Data BRAM Data Width			
4k / (1024 x 32)	1	10	BRAM_Addr [11:2]
8k / (2048 x 32)	2	11	BRAM_Addr [12:2]
16k / (4096 x 32)	4	12	BRAM_Addr [13:2]
32k / (8192 x 32)	8	13	BRAM_Addr [14:2]
64k / (16384 x 32)	16	14	BRAM_Addr [15:2]
64-bit Data BRAM Data Width			
8k / (1024 x 64)	2	10	BRAM_Addr [12:3]
16k / (2048 x 64)	4	11	BRAM_Addr [13:3]
32k / (4096 x 64)	8	12	BRAM_Addr [14:3]
64k / (8192 x 64)	16	13	BRAM_Addr [15:3]
128k / (16384 x 64)	32	14	BRAM_Addr [16:3]
128-bit Data BRAM Data Width			
16k / (1024 x 128)	4	10	BRAM_Addr [13:4]
32k / (2048 x 128)	8	11	BRAM_Addr [14:4]
64k / (4096 x 128)	16	12	BRAM_Addr [15:4]
128k / (8192 x 128)	32	13	BRAM_Addr [16:4]
256k / (16384 x 128)	64	14	BRAM_Addr [17:4]

Table 7 illustrates the BRAM configuration width sizes when targeting Spartan-6 FPGAs. The Spartan-6 FPGA 18 Kb block RAM is utilized by the EDK or can be selectable in the BRAM Generator.

Table 7: **BRAM Configuration for Spartan-6 FPGAs**

Supported Memory Sizes / BRAM Memory Configuration	Number of BRAM Primitives (18k/each)	Size of BRAM_Addr (each port)	Typical BRAM_Addr bit usage with BRAM Configuration
32-bit BRAM Data Width			
8k / (2048 x 32)	4	11	BRAM_Addr [14:2]
16k / (4096 x 32)	8	12	BRAM_Addr [15:2]
32k / (8192 x 32)	16	13	BRAM_Addr [16:2]
64k / (16384 x 32)	32	14	BRAM_Addr [17:2]
64-bit BRAM Data Width			
16k / (2048 x 64)	8	11	BRAM_Addr [13:3]
32k / (4096 x 64)	16	12	BRAM_Addr [14:3]
64k / (8192 x 64)	32	13	BRAM_Addr [15:3]

Table 7: **BRAM Configuration for Spartan-6 FPGAs *(Cont'd)***

Supported Memory Sizes / BRAM Memory Configuration	Number of BRAM Primitives (18k/each)	Size of BRAM_Addr (each port)	Typical BRAM_Addr bit usage with BRAM Configuration
128k / (16384 x 64)	64	14	BRAM_Addr [16:3]
128-bit BRAM Data Width			
32k / (2048 x 128)	16	11	BRAM_Addr [15:4]
64k / (4096 x 128)	32	12	BRAM_Addr [16:4]
128k / (8192 x 128)	64	13	BRAM_Addr [17:4]
256k / (16384 x 128)	128	14	BRAM_Addr [18:4]

Data Organization

The BRAM interface is designed to connect to the BRAM memory in a little-endian data structure, in matching the AXI data structure. The byte ordering and numbering for a 32-bit BRAM is shown in Table 8 and Table 9 for a 64-bit BRAM.

Table 8: **Little Endian Byte Ordering (32-bit BRAM)**

Byte Address	n+3	n+2	n+1	n
Byte Label	3	2	1	0
Byte Significance	MSByte			LSByte
Bit Label	31			0
Bit Significance	MSBit			LSBit

Table 9: **Little Endian Byte Ordering (64-bit BRAM)**

Byte Address	n+7	n+6	...	n+1	n
Byte Label	7	6	...	1	0
Byte Significance	MSByte		...		LSByte
Bit Label	63				0
Bit Significance	MSBit				LSBit

ECC

To mitigate the effect of BRAM Single Event Upsets, SEUs, the AXI BRAM 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 any single bit errors and detect any double bit errors in the data read. Errors are either signaled on the AXI connection, AXI_RRESP (a SLVERR value) to MicroBlaze™ processor (or the requesting AXI master device), or through an interrupt signal.

ECC is available in both AXI4-Lite and AXI4 interface configurations of the AXI BRAM controller core. ECC is available for both single and dual port connection types to the BRAM block (the setting of C_SINGLE_PORT_BRAM can be 0 or 1). ECC is supported on BRAM block data widths of 32, 64, or 128 bits only. The only combination not allowed for ECC support is when C_S_AXI_DATA_WIDTH = 256 or larger data widths.

The available ECC control and status registers described in (ECC Register Details, page 25) are available when C_ECC = 1.

ECC is inserted on the write datapath in the AXI BRAM controller for the data presented to the BRAM block at each memory location. In a full AXI interface connection to support all data transaction types (including narrow and unaligned data transfers, the ECC logic incorporates a read-modify-write sequence. The design ensures that each write to block RAM stores the full data width, with the supporting ECC data bits. The AXI BRAM controller supports 32-bits of data, with 7 bits of ECC data generated. In a 64-bit configuration, 8 bits of ECC data are created to store with the corresponding data. For 128-bit BRAM data width with ECC requires 9 bits of ECC to be stored with the data.

Each set of ECC bits are based (and calculated) on the full word (for 32-bit BRAM configurations) or the full double word (in 64-bit BRAM configurations). For each AXI transaction, a read-modify-write sequence is performed to read the full word (or BRAM data width) of data, merge in the new write data (indicated by the AXI_WSTRB signal) on the AXI bus, and the write the new data (and ECC bits) to block RAM. This sequence is shown in Figure 10 (on the AXI4-Lite interface) and Figure 17 (for the AXI4 interface).

Table 10 summarizes the additional ECC data bits to the BRAM interface.

Table 10: **ECC Data Sizes**

C_S_AXI_DATA_WIDTH	Number of ECC data bits	Number of ECC WEs
32	7	1
64	8	1
128	9	2

Previous to the AXI BRAM Controller v1.03a release, a Hamming Code algorithm was utilized in generating the ECC bits for 32-bit and 64-bit BRAM data widths. With the introduction of support for ECC in 128-bit BRAM data widths, a new ECC algorithm is utilized, referred to as Hsiao ECC coding.

Hsiao ECC in v1.03a

The Hsiao algorithm, which is utilized in the Xilinx MIG HDL and MPMC tools is incorporated in the AXI BRAM controller. The Hsiao algorithm provides an ECC encoding/decoding for optimized resource utilization and performance. The Hsiao algorithm is designed such that an equal number of inputs are used to generate each check bit in the ECC code. The number of inputs to generate each check bit depends on the overall data width of the block RAM + required number of ECC bits for that data width.

The AXI BRAM HDL generates the h-matrix as shown in Figure 4 for 32-bit BRAM data width configurations. The same methodology is applied to 64-bit and 128-bit data width configurations with ECC enabled. The h-matrix follows the rules of the Hsiao ECC algorithm. See the IBM documentation by M.Y. Hsiao in Reference Documents, page 66.

The dimensions of the h-matrix is determined by the number of ECC bits and the overall code width (which equals the BRAM data width + ECC code width). For 32-bit systems, the h-matrix is 7 x 39. And filled by combinations of (7 1) for the ECC bits, plus combinations of (7 3). Not all (7 3) combinations are used in the 32 data bits of the h-matrix.

For 64-bit systems, the h-matrix is 8 x 72. The h-matrix is constructed of all the (8 1) combinations for the ECC bits, all (7 3) combinations are used, and 8 combinations of the weight-5 codes (8 5) are used.

The Hsiao generator creates the “one at a time” column entries for the ECC parity bits. The generator then starts at one end of the h-matrix (shown as the last entry in Figure 4), and generates the combination of weight-3 codes. For 32-bit data widths, not all weight-3 codes are used. For 64-bit h-matrix generation, when all weight-3 codes are used, it generates weight-5 codes.

The previously described ECC encoding is not unique; there are many other valid SEC-DEC codes that can be utilized. Special consideration must be applied to systems in which port B of the block RAM is used by a separate system or a different BRAM controller. The ECC algorithms for encoding and decoding data must be identical between both ports.

For ECC bit generation, the column of the h-matrix is merged with the data pattern written to memory. The h-matrix bits pertaining to each check bit are XOR together to generate each check bit.

Upon ECC decoding, the entire h-matrix (data bits + ECC bits) are used to generate the syndrome value. For Hsiao ECC, when the syndrome value is equal to all zeros, no errors exist in the data. When the syndrome value has at least one bit equal to '1', and the number of 1's in the syndrome is an odd number, then a single bit error is detected. The syndrome value that matches the h-matrix indicates the bit position in error. When the syndrome value has at least one bit equal to '1', and the number of 1's in the syndrome is an even number, then a double bit error is detected.

For more details on the ECC Hsiao algorithm, contact the Xilinx Technical Support.

The parity check matrix for 32-bit BRAM data (39, 32) Hsiao code is shown in Figure 4.

H-Matrix Bit	H-Matrix Value (Hex) (6:0)	H-Matrix Value (Binary) (6:0)	ECC Bit Positions Set to '1'	Notes
38	01	0000001	1	
37	02	0000010	2	
36	04	0000100	3	
35	08	0001000	4	
34	10	0010000	5	
33	20	0100000	6	
32	40	1000000	7	When the h-matrix reaches the data width, only one bit in the the ECC word is asserted.
31	0E	0001110	234	
30	13	0010011	125	
29	15	0010101	135	
28	16	0010110	235	
27	19	0011001	145	
26	1A	0011010	245	
25	1C	0011100	345	
24	23	0100011	126	
23	25	0100101	136	
22	26	0100110	236	
21	29	0101001	146	
20	2A	0101010	246	
19	2C	0101100	346	
18	31	0110001	156	
17	32	0110010	256	
16	34	0110100	356	
15	38	0111000	456	
14	43	1000011	127	
13	45	1000101	137	
12	46	1000110	237	
11	40	1001001	147	
10	4A	1001010	247	
9	4C	1001100	347	
8	51	1010001	157	
7	52	1010010	257	
6	54	1010100	357	
5	58	1011000	457	
4	61	1100001	167	
3	62	1100010	267	And so on...
2	64	1100100	367	Again, finds next combination based on prior seed.
1	68	1101000	467	Generates “1101000” based on initial seed of prior combination, “111000”.
0	70	1110000	567	Starting point in H-matrix calculation. Three unique bits of the ECC word are asserted.

DS777_07

Figure 4: **32-bit Hsiao Encoding**

The h-matrix generated for 64-bit ECC encoding in shown in Figure 5.

H-Matrix Bit	H-Matrix Value (Hex) (7:0)	H-Matrix Value (Binary) (7:0)	Notes	H-Matrix Bit	H-Matrix Value (Hex) (7:0)	H-Matrix Value (Binary) (7:0)	Notes
71	01	00000001		39	31	00110001	
70	02	00000010		...			
69	04	00000100		31	4A	01001000	
68	08	00001000		30	4C	01001100	
67	10	00010000		29	51	01010001	
66	20	00100000		28	52	01010010	
65	40	01000000	When the H-matrix reaches the data width maximum, only one bit of the ECC is asserted.	27	54	01010100	
				26	58	01011000	
64	80	10000000		25	61	01100001	
63	E6	11100110		24	62	01100010	
62	E9	11101001		23	64	01100100	
61	Ea	11101010		22	68	01101000	
60	Ec	11101100		21	70	01110000	
59	F1	11110001		20	83	10000011	
58	F2	11110010		19	85	10000101	
57	F4	11110100	Starts with weight-5 codes. Some implementations of Hsiao dispurse the weight-5 codes with the weight-3 codes, but they are assigned to the upper order data bits.	18	86	10000110	
56	F8	11111000		17	89	10001001	
55	07	00000111		16	8A	10001010	
54	0B	00001011		15	8C	10001100	
53	0d	00001101	64-bit Hsiao uses all weight-3 code combinations.	14	91	10010001	
52	0E	00001110		13	92	10010010	
51	13	00010011		12	94	10010100	
50	15	00010101		11	98	10011000	
49	16	00010110		10	A1	10100001	
48	19	00011001		9	A2	10100010	
47	1A	00011010		8	A4	10100100	
46	1C	00011100		7	A8	10101000	
45	23	00100011		6	B0	10110000	
44	25	00100101		5	C1	11000001	
43	26	00100110		4	C3	11000010	
42	29	00101001		3	C4	11001000	And so on...
41	2A	00101010		2	C8	11001000	Again, finds next combination based on prior seed.
40	2C	00101100		1	D0	11010000	Generates “1101000” based on initial seed of prior combination, “111000”.
				0	E0	11100000	Starting point in H-matrix calculation.

DS777_07

Figure 5: **64-bit Hsiao Encoding**

The Hsiao ECC encoding (137, 128) for 128-bit configurations is shown in Figure 6. Not all encodings are shown, but the concept is identical to the 32-bit and 64-bit implementations. The 128-bit ECC algorithm is shown in Figure 6.

H-Matrix Bit	H-Matrix Value (Hex) (8:0)	H-Matrix Value Binary (8:0)	Notes
136	001	000000001	
135	002	000000010	
134	004	000000100	
133	008	000001000	
132	010	000010000	
131	020	000100000	
130	040	001000000	
129	080	010000000	
128	100	100000000	When the H-matrix reaches the data width maximum, the ECC bits have one bit asserted.
127	165	101100101	
126	166	101100110	
125	169	101101001	
124	16C	101101100	
123	171	101110001	
122	172	101110010	
121	174	101110100	
120	178	101111000	Weight-5 codes
119	187	110000111	
118 117	18B 18D	110001011 110001101	
116	18E	110001110	
...			
85	1E8	111101000	
84	1F0	111110000	Start of weight-5 codes
83	007	000000111	Last combination of a weight-3 code
...			
7	160	101100000	
6	181	110000001	
5	182	110000010	
4	184	110000100	
3	188	110001000	
2	190	110010000	
1	1A0	110100000	
0	1C0	111000000	Start with weight-3 code (MSB all 1s)

DS777_08

Figure 6: **128-bit Hsiao Encoding**

ECC Register Details

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, 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.

This register is available only when C_ECC = 1.

Table 11: **ECC Status Register (ECC_STATUS)**

Reserved		ECC_STATUS	
31	2	1	0

Table 12: **ECC Status Register Bit Definitions**

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

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 (ECC_Interrupt). 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.

This register is available only when C_ECC = 1.

Table 13: **ECC Interrupt Enable Register (ECC_EN_IRQ)**

Reserved		ECC_EN_IRQ	
31	2	1	0

Table 14: **ECC Interrupt Enable Register Bit Definitions**

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

ECC_ON_OFF

The ECC On/Off Control Register allows the application to enable or disable ECC checking. The design parameter, C_ECC_ONOFF_RESET_VALUE (default on) determines the reset value for the enable/disable setting of ECC. This facilitates start-up operations when ECC might or might not be initialized in the BRAM block. When disabled, ECC checking is disabled for read, but ECC generation is active for write operations.

This register is not implemented if the value of C_ECC is 0. In this case, ECC checking is always off.

Table 15: **ECC On/Off Control Register (ECC_ON_OFF)**

Reserved		ECC_ON_OFF
31	1	0

Table 16: **ECC On/Off Control Register Bit Definitions**

Bit(s)	Name	Core Access	Reset Value	Description
0	ECC_ON_OFF	R/W	Specified by design parameter, C_ECC_ONOFF_RESET_VALUE	If '0', ECC checking is disabled on read operations. (ECC generation is enabled on write operations when C_ECC = 1). If '1', ECC checking is enabled on read operations. All correctable and uncorrectable error conditions are captured and status updated.

CE_CNT

This register 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 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. The AXI BRAM controller has the value of the CE counter width set to 8 bits.

This counter register is only accessible when C_ECC = 1

Table 17: **Correctable Error Counter Register (CE_CNT)**

Reserved		CE_CNT	
31	8	7	0

Table 18: **Correctable Error Counter Register Bit Definitions**

Bit(s)	Name	Core Access	Reset Value	Description
(7:0)	CE_CNT	R/W	0	Registers holds number of correctable errors encountered

CE_FFA [31:0]

This register stores the address (bits [31:0]) of the first occurrence of an access with a correctable error. When the CE_STATUS bit in the ECC Status Register is cleared, the register is re-enabled to store the address of the next correctable error. Storage of the failing address is enabled after reset.

This register is only implemented when C_ECC = 1.

Table 19: **Correctable Error First Failing Address Register (CE_FFA [31:0])**

CE_FFA [31:0]	
31	0

Table 20: **Correctable Error First Failing Address [31:0] Register Bit Definitions**

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

CE_FFA [63:32]

This register is unused in this release of the AXI BRAM controller IP core. The maximum address width setting on the block RAM is 32.

This register stores the address (bits [63:32]) of the first occurrence of an access with a correctable error. When the CE_STATUS bit in the ECC Status Register is cleared, the register is re-enabled to store the address of the next correctable error. Storage of the failing address is enabled after reset.

This register is only implemented when C_ECC = 1.

Table 21: **Correctable Error First Failing Address Register (CE_FFA [63:32])**

CE_FFA [63:32]	
31	0

Table 22: **Correctable Error First Failing Address [63:32] Register Bit Definitions**

Bit(s)	Name	Core Access	Reset Value	Description
(31:0)	CE_FFA [63:32]	R	0	Address (bits [63:32]) of the first occurrence of a correctable error

CE_FFD [31:0]

This register is unused in this release of the AXI BRAM controller IP core.

This register stores the (uncorrected) failing data (bits [31:0]) of the first occurrence of an access with a correctable error. When the CE_STATUS bit in the ECC Status Register is cleared, the register is re-enabled to store the data of the next correctable error. Storage of the failing data is enabled after reset.

This register is only implemented when C_ECC is set to 1.

Table 23: **Correctable Error First Failing Data Register (CE_FFD [31:0])**

CE_FFD [31:0]	
31	0

Table 24: **Correctable Error First Failing Data [31:0] Register Bit Definitions**

Bit(s)	Name	Core Access	Reset Value	Description
(31:0)	CE_FFD [31:0]	R	0	Data (bits [31:0]) of the first occurrence of a correctable error.

CE_FFD [63:32]

This register is unused in this release of the AXI BRAM controller IP core.

This register is only used when the C_S_AXI_DATA_WIDTH > 32.

This register stores the (uncorrected) failing data (bits [63:32]) 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. Storage of the failing data is enabled after reset.

This register is only implemented when C_ECC is set to 1.

Table 25: **Correctable Error First Failing Data Register (CE_FFD [63:32])**

CE_FFD [63:32]	
31	0

Table 26: **Correctable Error First Failing Data [63:32] Register Bit Definitions**

Bit(s)	Name	Core Access	Reset Value	Description
(31:0)	CE_FFD [63:32]	R	0	Data (bits [63:32]) of the first occurrence of a correctable error.

CE_FFD [95:64]

This register is unused in this release of the AXI BRAM controller IP core.

This register is only used when the C_S_AXI_DATA_WIDTH > 64.

This register is unused in this release of the AXI BRAM Controller IP core; however, it will be available in a future release of the IP core when ECC is supported for 128-bit AXI BRAM data width configurations.

This register stores the (uncorrected) failing data (bits [95:64]) 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. Storage of the failing data is enabled after reset.

This register is only implemented when C_ECC is set to 1.

Table 27: **Correctable Error First Failing Data Register (CE_FFD [95:64])**

CE_FFD [95:64]	
31	0

Table 28: **Correctable Error First Failing Data [95:64] Register Bit Definitions**

Bit(s)	Name	Core Access	Reset Value	Description
(31:0)	CE_FFD [95:64]	R	0	Data (bits [95:64]) of the first occurrence of a correctable error.

CE_FFD [127:96]

This register is unused in this release of the AXI BRAM controller IP core.

This register is only used when the C_S_AXI_DATA_WIDTH > 64.

This register is unused in this release of the AXI BRAM Controller IP core; however, it will be available in a future release of the IP core when ECC is supported for 128-bit AXI BRAM data width configurations.

This register stores the (uncorrected) failing data (bits [127:96]) 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. Storage of the failing data is enabled after reset.

This register is implemented only when C_ECC is set to 1.

Table 29: **Correctable Error First Failing Data Register (CE_FFD [127:96])**

CE_FFD [127:96]	
31	0

Table 30: **Correctable Error First Failing Data [127:96] Register Bit Definitions**

Bit(s)	Name	Core Access	Reset Value	Description
(31:0)	CE_FFD [127:96]	R	0	Data (bits [127:96]) of the first occurrence of a correctable error.

CE_FFE

This register is unused in this release of the AXI BRAM controller IP core.

This register stores the ECC bits 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. Storage of the failing ECC is enabled after reset.

This register is only implemented when C_ECC = 1.

Table 31 and Table 32 describe the register bit usage when C_S_AXI_DATA_WIDTH = 32.

Table 31: **Correctable Error First Failing ECC Register (CE_FFE) for 32-bit BRAM**

Reserved		CE_FFE	
31	7	6	0

Table 32: **Correctable Error First Failing ECC Register Bit Definitions for 32-bit BRAM**

Bit(s)	Name	Core Access	Reset Value	Description
(6:0)	CE_FFE	R	0	ECC (bits [6:0]) of the first occurrence of a correctable error

Table 33 and Table 34 describe the register bit usage when C_S_AXI_DATA_WIDTH = 64.

Table 33: **Correctable Error First Failing ECC Register (CE_FFE) for 64-bit BRAM**

Reserved		CE_FFE	
31	8	7	0

Table 34: **Correctable Error First Failing ECC Register Bit Definitions for 64-bit BRAM**

Bit(s)	Name	Core Access	Reset Value	Description
(7:0)	CE_FFE	R	0	ECC (bits [7:0]) of the first occurrence of a correctable error.

Table 35 and Table 36 describe the register bit usage when C_S_AXI_DATA_WIDTH = 128.

This register is unused in this release of the AXI BRAM controller IP core.

Table 35: **Correctable Error First Failing ECC Register (CE_FFE) for 128-bit BRAM**

Reserved		CE_FFE [8:0]	
31	9	8	0

Table 36: **Correctable Error First Failing ECC Register Bit Definitions for 128-bit BRAM**

Bit(s)	Name	Core Access	Reset Value	Description
(8)	CE_FFE [8]	R	0	ECC (bit [8]) of the first occurrence of a correctable error.
(7:0)	CE_FFE [7:0]	R	0	ECC (bits [7:0]) of the first occurrence of a correctable error.

UE_FFA [31:0]

This register is unused in this release of the AXI BRAM controller IP core.

This register stores the address (bits [31:0]) 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. Storage of the failing address is enabled after reset.

This register is only implemented when C_ECC = 1.

Table 37: **Uncorrectable Error First Failing Address Register (UE_FFA [31:0])**

UE_FFA [31:0]	
31	0

Table 38: **Uncorrectable Error First Failing Address [31:0] Register Bit Definitions**

Bit(s)	Name	Core Access	Reset Value	Description
(31:0)	UE_FFA [31:0]	R	0	Address (bits [31:0]) of the first occurrence of a uncorrectable error

UE_FFA [63:32]

This register is unused in this release of the AXI BRAM controller IP core. The maximum address width setting on the block RAM is 32.

This register stores the address (bits [63:32]) 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. Storage of the failing address is enabled after reset.

This register is only implemented when C_ECC = 1.

Table 39: **Uncorrectable Error First Failing Address Register (UE_FFA [63:32])**

UE_FFA [63:32]	
31	0

Table 40: **Uncorrectable Error First Failing Address [63:32] Register Bit Definitions**

Bit(s)	Name	Core Access	Reset Value	Description
(31:0)	UE_FFA [63:32]	R	0	Address (bits [63:32]) of the first occurrence of a uncorrectable error

UE_FFD [31:0]

This register is unused in this release of the AXI BRAM controller IP core.

This register stores the (uncorrected) failing data (bits [31:0]) 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. Storage of the failing data is enabled after reset.

This register is only implemented when C_ECC = 1.

Table 41: **Uncorrectable Error First Failing Data Register (UE_FFD [31:0])**

UE_FFD [31:0]	
31	0

Table 42: **Uncorrectable Error First Failing Data [31:0] Register Bit Definitions**

Bit(s)	Name	Core Access	Reset Value	Description
(31:0)	UE_FFD [31:0]	R	0	Data (bits [31:0]) of the first occurrence of a uncorrectable error.

UE_FFD [63:32]

This register is unused in this release of the AXI BRAM controller IP core.

This register is only used when the C_S_AXI_DATA_WIDTH > 32.

This register stores the (uncorrected) failing data (bits [63:32]) 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. Storage of the failing data is enabled after reset.

This register is only implemented when C_ECC = 1.

Table 43: **Uncorrectable Error First Failing Data Register (UE_FFD [63:32])**

UE_FFD [63:32]	
31	0

Table 44: **Uncorrectable Error First Failing Data [63:32] Register Bit Definitions**

Bit(s)	Name	Core Access	Reset Value	Description
(31:0)	UE_FFD [63:32]	R	0	Data (bits [63:32) of the first occurrence of an uncorrectable error.

UE_FFD [95:64]

This register is unused in this release of the AXI BRAM controller IP core.

This register is only used when the C_S_AXI_DATA_WIDTH > 64.

This register is unused in this release of the AXI BRAM Controller IP core; however, it will be available in a future release of the IP core when ECC is supported for 128-bit AXI BRAM data width configurations.

This register stores the (uncorrected) failing data (bits [95:64]) 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. Storage of the failing data is enabled after reset.

This register is only implemented when C_ECC = 1.

Table 45: **Uncorrectable Error First Failing Data Register (UE_FFD [95:64])**

UE_FFD [95:64]	
31	0

Table 46: **Uncorrectable Error First Failing Data [95:64] Register Bit Definitions**

Bit(s)	Name	Core Access	Reset Value	Description
(31:0)	UE_FFD [95:64]	R	0	Data (bits [95:64]) of the first occurrence of an uncorrectable error.

UE_FFD [127:96]

This register is unused in this release of the AXI BRAM controller IP core.

This register is only used when the C_S_AXI_DATA_WIDTH > 64.

Unused in this release of the AXI BRAM Controller IP core. Will be available in a future release of the IP core when ECC is supported for 128-bit AXI BRAM data width configurations.

This register stores the (uncorrected) failing data (bits [127:96]) 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. Storage of the failing data is enabled after reset.

This register is only implemented when C_ECC = 1.

Table 47: **Uncorrectable Error First Failing Data Register (UE_FFD [127:96])**

UE_FFD [127:96]	
31	0

Table 48: **Uncorrectable Error First Failing Data [127:96] Register Bit Definitions**

Bit(s)	Name	Core Access	Reset Value	Description
(31:0)	UE_FFD [127:96]	R	0	Data (bits [127:96]) of the first occurrence of an uncorrectable error.

UE_FFE

This register is unused in this release of the AXI BRAM controller IP core.

This register stores the ECC bits 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 ECC of the next uncorrectable error. Storage of the failing ECC is enabled after reset.

This register is only implemented when C_ECC = 1.

Table 49 and Table 50 describe the register bit usage when C_S_AXI_DATA_WIDTH = 32.

Table 49: **Uncorrectable Error First Failing ECC Register (UE_FFE) for 32-bit BRAM**

Reserved		UE_FFE	
31	7	6	0

Table 50: **Uncorrectable Error First Failing ECC Register Bit Definitions for 32-bit BRAM**

Bit(s)	Name	Core Access	Reset Value	Description
(6:0)	UE_FFE	R	0	ECC (bits [6:0]) of the first occurrence of an uncorrectable error

Table 51 and Table 52 describe the register bit usage when C_S_AXI_DATA_WIDTH = 64.

Table 51: **Uncorrectable Error First Failing ECC Register (UE_FFE) for 64-bit BRAM**

Reserved		UE_FFE	
31	8	7	0

Table 52: **Uncorrectable Error First Failing ECC Register Bit Definitions for 64-bit BRAM**

Bit(s)	Name	Core Access	Reset Value	Description
(7:0)	UE_FFE	R	0	ECC (bits [7:0]) of the first occurrence of an uncorrectable error.

Table 53 and Table 54 describe the register bit usage when C_S_AXI_DATA_WIDTH = 128.

Table 53: **Uncorrectable Error First Failing ECC Register (UE_FFE) for 128-bit BRAM**

Reserved		UE_FFE [8:0]	
31	9	8	0

Table 54: **Uncorrectable Error First Failing ECC Register Bit Definitions for 128-bit BRAM**

Bit(s)	Name	Core Access	Reset Value	Description
8	UE_FFE [8]	R	0	ECC (bit [8]) of the first occurrence of an uncorrectable error.
(7:0)	UE_FFE [7:0]	R	0	ECC (bits [7:0]) of the first occurrence of an uncorrectable error.

FI_D0

This register is used to inject errors in data (bits [31:0]) written to the block RAM and can be used to test the error correction and error signaling. The bits set in the register toggle the corresponding data bits (word 0 or bits [31:0]) 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 only implemented if C_FAULT_INJECT = 1 and C_ECC = 1.

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

Table 55: **Fault Injection Data Register (FI_D0)**

FI_D0	
31	0

Table 56: **Fault Injection Data (Word 0) Register Bit Definitions**

Bit(s)	Name	Core Access	Reset Value	Description
(31:0)	FI_D0	W	0	Bit positions set to '1' toggle the corresponding bits [31:0] of the next data word written to the block RAM. The register is automatically cleared after the fault has been injected.

For larger data width BRAMs with ECC (64- and 128-bit configurations), special consideration must be given across FI_D0, FI_D1, FI_D2 and FI_D3 such that only a single error condition is introduced.

FI_D1

This register is only used when the C_S_AXI_DATA_WIDTH > 32.

This register is used to inject errors in data (bits [63:32]) written to the block RAM and can be used to test the error correction and error signaling. The bits set in the register toggle the corresponding data bits (word 1 or bits [63:32]) 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.

This register is only implemented if C_FAULT_INJECT = 1 and C_ECC = 1.

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

Table 57: **Fault Injection Data Register (FI_D1)**

FI_D1	
31	0

Table 58: **Fault Injection Data (Word 1) Register Bit Definitions**

Bit(s)	Name	Core Access	Reset Value	Description
(31:0)	FI_D1	W	0	Bit positions set to '1' toggle the corresponding bits [63:32] of the next data word written to the block RAM. The register is automatically cleared after the fault has been injected.

FI_D2

This register is only used when the C_S_AXI_DATA_WIDTH > 64.

This register is used to inject errors in data (bits [95:64]) written to the block RAM and can be used to test the error correction and error signaling. The bits set in the register toggle the corresponding data bits (word 2 or bits [95:64]) 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.

This register is only implemented if C_FAULT_INJECT = 1 and C_ECC = 1.

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

Table 59: **Fault Injection Data Register (FI_D2)**

FI_D2	
31	0

Table 60: **Fault Injection Data (Word 2) Register Bit Definitions**

Bit(s)	Name	Core Access	Reset Value	Description
(31:0)	FI_D2	W	0	Bit positions set to '1' toggle the corresponding bits [95:64] of the next data word written to the block RAM. The register is automatically cleared after the fault has been injected.

FI_D3

This register is only used when the C_S_AXI_DATA_WIDTH > 64.

This register is used to inject errors in data (bits [127:96]) written to the block RAM and can be used to test the error correction and error signaling. The bits set in the register toggle the corresponding data bits (word 3 or bits [127:96]) 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.

This register is only implemented if C_FAULT_INJECT = 1 and C_ECC = 1.

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

Table 61: **Fault Injection Data Register (FI_D3)**

FI_D3	
31	0

Table 62: **Fault Injection Data (Word 3) Register Bit Definitions**

Bit(s)	Name	Core Access	Reset Value	Description
0 to 31	FI_D3	W	0	Bit positions set to '1' toggle the corresponding bits [127:96] of the next data word written to the block RAM. The register is automatically cleared after the fault has been injected.

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 signaling. 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 only implemented if C_FAULT_INJECT = 1 and C_ECC = 1.

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

Table 63 and Table 64 describes the register bit usage when C_S_AXI_DATA_WIDTH = 32.

Table 63: **Fault Injection ECC Register (FI_ECC) for 32-bit BRAM**

Reserved		FI_ECC	
31	7	6	0

Table 64: **Fault Injection ECC Register Bit Definitions for 32-bit BRAM**

Bit(s)	Name	Core Access	Reset Value	Description
(6:0)	FI_ECC	R	0	Bit positions set to '1' toggle the corresponding bit of the next ECC written to the block RAM. The register is automatically cleared after the fault has been injected.

Table 65 and Table 66 describe the register bit usage when C_S_AXI_DATA_WIDTH = 64.

Table 65: **Fault Injection ECC Register (FI_ECC) for 64-bit BRAM**

Reserved		FI_ECC	
31	8	7	0

Table 66: **Fault Injection ECC Register Bit Definitions for 64-bit BRAM**

Bit(s)	Name	Core Access	Reset Value	Description
(7:0)	FI_ECC	R	0	Bit positions set to '1' toggle the corresponding bit of the next ECC written to the block RAM. The register is automatically cleared after the fault has been injected.

Table 67 and Table 68 describe the register bit usage when C_S_AXI_DATA_WIDTH = 128.

Table 67: **Fault Injection ECC Register (FI_ECC) for 128-bit BRAM**

Reserved		FI_ECC	
31	9	8	0

Table 68: **Fault Injection ECC Register Bit Definitions for 128-bit BRAM**

Bit(s)	Name	Core Access	Reset Value	Description
8	FI_ECC [8]	R	0	Bit position set to '1' toggle the corresponding bit of the next ECC written to the block RAM. The register is automatically cleared after the fault has been injected.
(7:0)	FI_ECC [7:0]	R	0	Bit positions set to '1' toggle the corresponding bit of the next ECC written to the block RAM. The register is automatically cleared after the fault has been injected.

AXI4-Lite Controller Design

The AXI BRAM Controller IP core can be configured in a simplified footprint core by setting the design parameter, C_S_AXI_PROTOCOL = AXI4LITE.

In this configuration, the AXI BRAM Controller can be configured to interface to a single port or dual port BRAM. All arbitration to the block RAM (in a single port mode) is performed on the AXI write address (AW) and read address (AR) channels. Only one active operation is allowable in the core at a time. The AXI BRAM controller AW and AR channel interfaces assign priority to the read channel when presented with simultaneous assertions of the ARVALID and AWVALID signals; ensuring the AXI BRAM controller is compliant with any AXI4-Lite master. When the BRAM controller is created in a system topology with the AXI Interconnect IP, it is ensured that ARVALID and AWVALID are not asserted at the same time and the AXI Interconnect handles the arbitration between the write and read channels. The AXI Interconnect provides optimizations in the AXI BRAM controller core through the build process by duplicating the ARADDR and AWADDR signals.

To configure the dual-port BRAM mode to the BRAM block, ensure the design parameter, C_SINGLE_PORT_BRAM = 0.

The AXI4-Lite BRAM Controller is designed for minimal FPGA resource utilization. Minimal registers are utilized, only those to support the register outputs to the AXI4-Lite interface according to standard.

AXI4-Lite Single Write

The write datapath timing in the core is shown in Figure 7 for the AXI4-Lite interface connections. Figure 7 assumes that no valid transaction is active on the AXI AR and R channels.

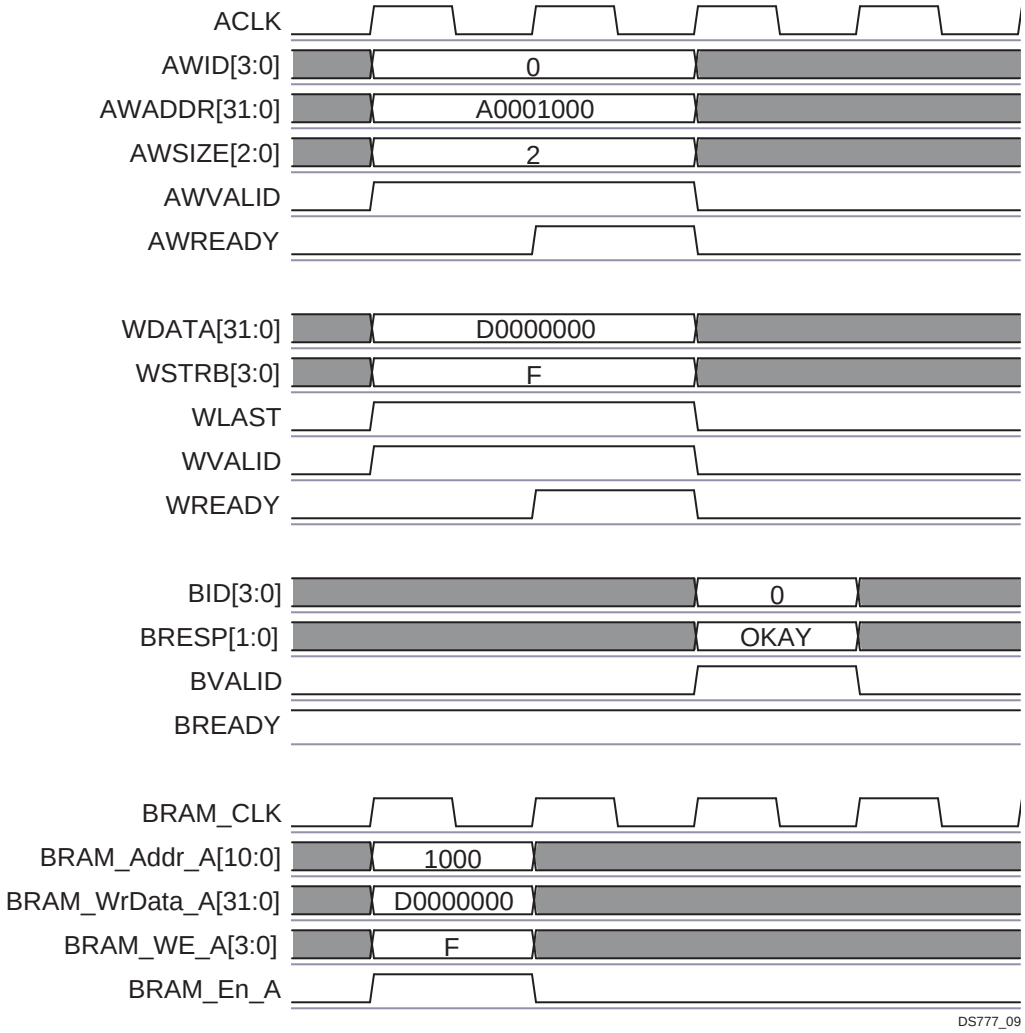

Figure 7: **AXI4-Lite Write Timing**

AXI4-Lite Multiple Write

The AXI4-Lite BRAM controller is able to accept continuous single write operations as shown in Figure 8.

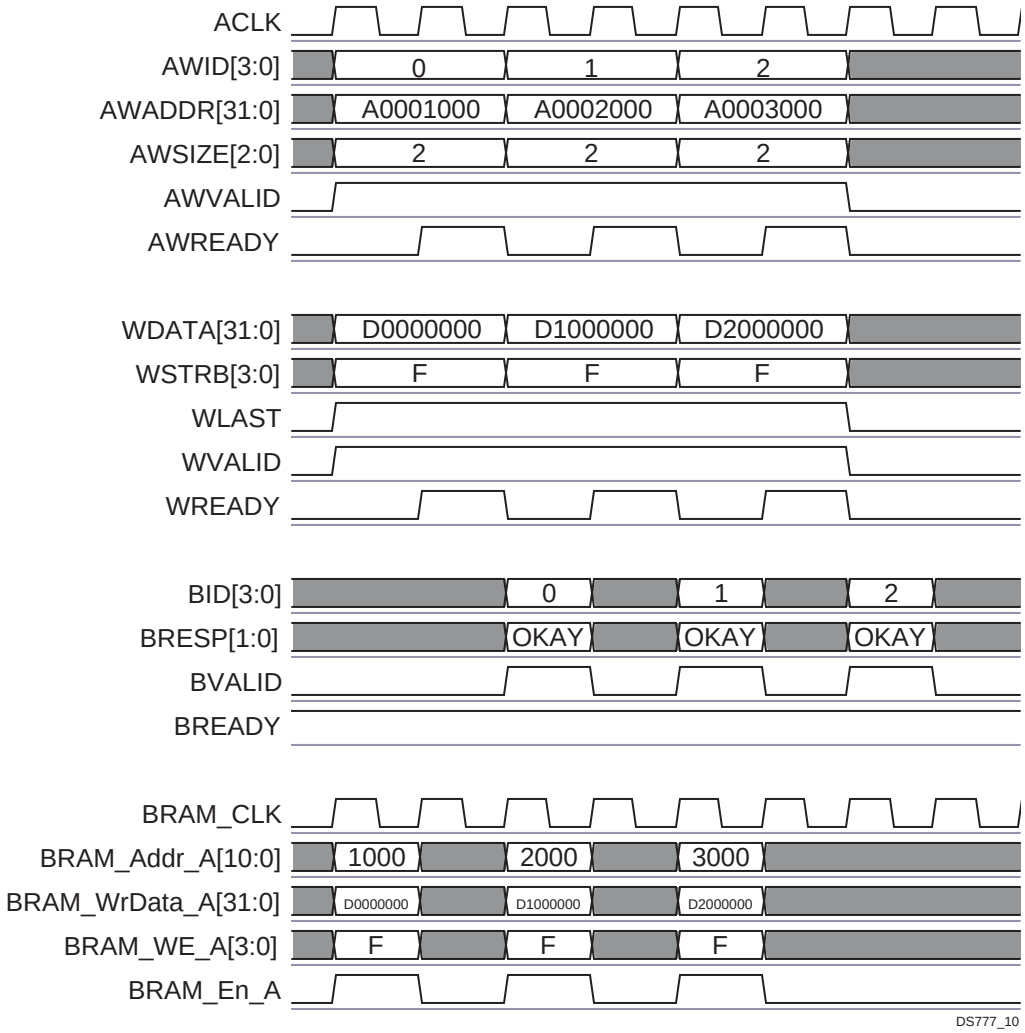

Figure 8: **Multiple AXI4-Lite Write Transactions**

AXI4-Lite Single Read

The read datapath timing is shown in Figure 9 for AXI4-Lite transactions. Multiple read request operations are shown in Figure 9 to illustrate the timing of the controller and on the BRAM interface.

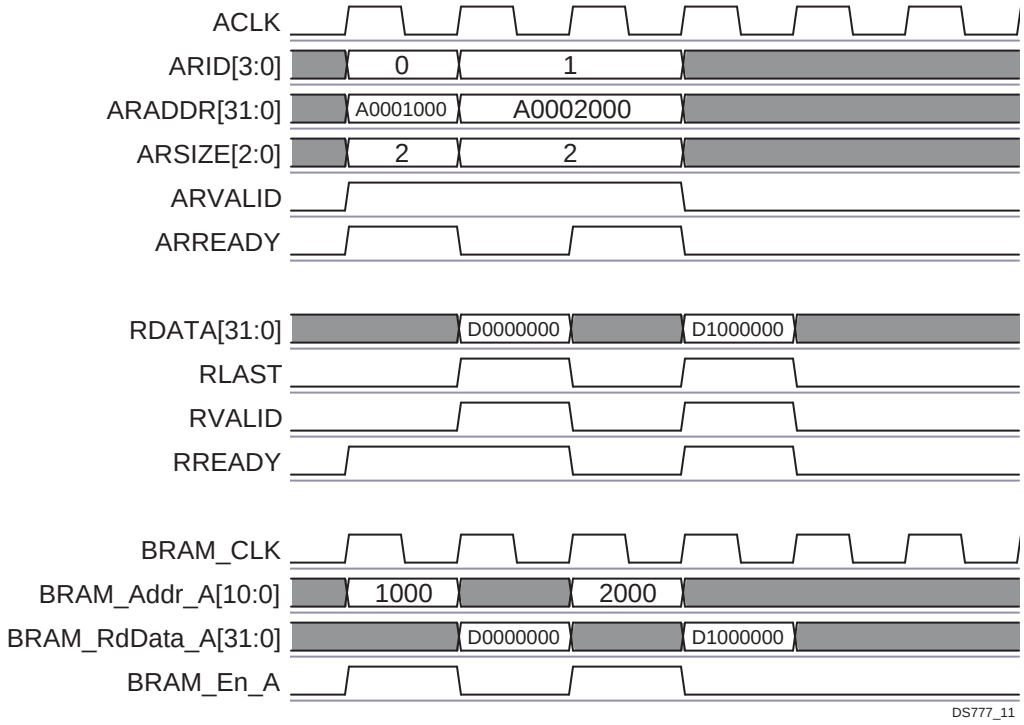

Figure 9: **AXI4-Lite Read Timing**

AXI4-Lite Write with ECC

Figure 10 illustrates the timing of the read-modify-write (RMW) sequence when C_ECC = 1 for write transfers on the AXI4-Lite interface. To save resources the AWADDR and WDATA are not registered in the AXI4-Lite BRAM controller, so the AWREADY and WREADY output flag assertions are delayed until the RMW sequence is completed. Figure 10 illustrates the capability of the AXI4-Lite controller to handle back-to-back write request operations.

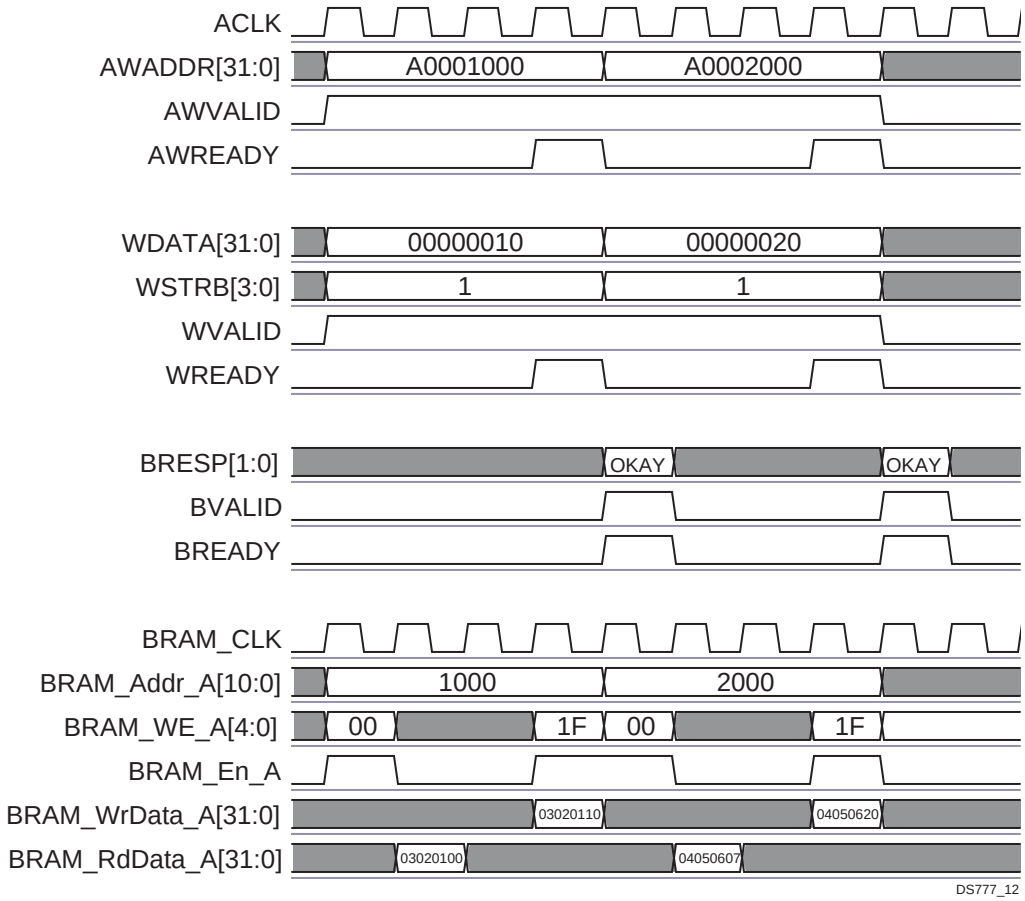

Figure 10: **AXI4-Lite Write with ECC**

AXI4 Controller Design

The AXI BRAM controller meets the performance requirement of AXI based systems and provides minimal latency to/from block RAM. Single clock cycle latency is achieved on each channel Valid to Ready response, depending on the current core activity. The AXI BRAM controller separates the write and read channel activity (when C_SINGLE_PORT_BRAM = 0), so that the controller can handle simultaneous read and write operations from AXI. All timing relationships between the write address and write data channel (as well as read address and read data channels) are designed to meet the AXI standard, so as to avoid any deadlock situations.

No address decoding is performed by the AXI BRAM Controller; all operations received are accepted. The Xilinx AXI Interconnect provides the address decode.

All AXI master size matching to the data width of the block RAM is handled by the Xilinx AXI Interconnect module. The slave IP port of the BRAM controller on the AXI Interconnect is equal to the width of the block RAM (equal to

the size of the BRAM controller). The AXI BRAM controller does not need to know the data width or burst size of the requesting master, as the Interconnect translates all operations to fit to the data width and burst size of the data bus connected to the BRAM controller (and block RAM).

AXI Singles

A single AXI transaction is qualified by the AWSIZE/ARSIZE equal to "000" and AWLEN/ARLEN set to "0000" for 1 data transfer. All AXI4-Lite operations are supported by the AXI BRAM controller when (C_S_AXI_PROTOCOL = AXI4).

Bursting

All burst types presented to the BRAM controller are supported. However, fixed burst types are translated into incrementing burst types issued to the block RAM. Wrapping burst types are support for processor cacheline read and writes with block RAM.

Each burst is specified to have N data transfers (1-256 specified in AWLEN/ARLEN) of M bytes (1-128 bytes specified in AWSIZE/ARSIZE). The AXI BRAM controller supports up to the AXI4 extension on burst sizes to 256 data beats.

The size of the burst must be equal or less than the size of the BRAM data width. For instance, 64-bit BRAM only allows bursts of up to 64 byte sizes. Burst sizes less than the full width of the AXI BRAM controller data bus is referred to as a "narrow" burst. All "narrow" bursts are supported by the AXI BRAM controller. For example, a master is requesting a byte burst operation to/from a 64-bit data wide BRAM. For "narrow" bursts, the AXI protocol defines that the valid byte lanes rotate through the correct byte lanes. In the AXI BRAM controller, each write data beat of a burst on the AXI is translated to a write operation to the block RAM. The BRAM byte enables are configured such that only the valid bytes are stored in memory. The AXI BRAM controller does not buffer any subsize data beats into the full width to the block RAM. An example is shown in the section, Narrow Write Bursting, page 49.

Unaligned burst transfers are allowed and the AXI master must indicate this with the lower order bits of the address bus provided with the write address channel data handshaking. An example is shown in section, Unaligned Write Bursting, page 50. Unaligned burst transfers can also be indicated with an aligned address, but use the write data channel data strobes to indicate the valid byte lanes.

Each write and read channel of the BRAM controller utilizes an address counter that is loaded at the beginning of the burst operation. AXI provides the starting address of the burst, and the BRAM controller increments the address based on the BRAM data width.

AXI does not allow any read nor write burst termination. Each AXI master must complete each burst transaction that is initiated. Each burst transaction is complete when the LAST signal is asserted, by the master of writes, and by the BRAM controller on reads.

The write enables presented to the block RAM are calculated based on burst type, address offset of the write address, along with the AXI write data strobes. The write enables are generated on a per byte basis to the block RAM. Write enables are 4 bits wide for 32-bit BRAMs, 8 bits wide for 64-bit BRAMs, and 16 bits wide for 128-bit BRAM instantiations.

Cacheline

Cacheline operations are implemented as WRAP burst types on the AXI bus when presented to the block RAM. The allowable burst sizes for WRAP bursts are 2, 4, 8, and 16 data transfers. The AWBURST/ARBURST must be set to "10" for the WRAP burst type.

WRAP bursts are handled in the address generator logic of the controller to the block RAM. The address seen by the block RAM must increment to the boundary, then wrap back around to the beginning of the cacheline address. For example, a processor issuing a target word first cache line read request to address, 0x04h. A 32-bit BRAM sees the following sequence of address-requested reads: 0x04h, 0x08h, 0x0Ch, 0x00h. This example is illustrated in Cacheline Reads, page 59 for a cacheline read.

Pipelining

Each write and read channel interface to the AXI can hold two active addresses/operations. The AXI BRAM Controller IP allows two active write and two active read transactions to be captured and held (when C_SINGLE_PORT_BRAM = 0). The data provided with each address channel operation must remain in order. No out of order operations are allowed in the AXI BRAM controller. The pipelining capability allows the same master or another master to request a subsequent write or read transaction. With the storage of multiple write addresses in the BRAM controller, the write data may not be immediately captured and stored by the controller. The write data channel interface of the BRAM controller does not assert WREADY until the block RAM is ready to accept the data of the pipelined write transaction. On the write address channel interface, the BRAM controller keeps AWREADY asserted, until the pipeline full condition is reached. At this point, the AWREADY is negated and no further write addresses are accepted, until the first address data phase is complete and the second pipelined address can be processed by the controller. An example of the timing relationship is shown in Write Pipeline, page 51.

On read transactions, two deep address pipelining is supported in the BRAM controller. The BRAM controller issues the request to block RAM and provides data on the read data channel, as long as the requesting master asserts RREADY.

When the IP core is configured for single port BRAM utilization (C_SINGLE_PORT_BRAM = 1), a 2-deep address pipeline exists in the core, but is shared between requesting AR and AW address transactions.

ID Wrapping

The BRAM controller does not support data reordering. All ID values are wrapped back around on the data channel during handshaking. The AWID (on the write address channel) is captured and returned as the BID signal (of the write response channel) during the write data transaction. The ARID (on the read address channel) is captured and returned as the RID signal (of the read response channel) during the read data response transaction.

Power Considerations

To conserve power consumption on the BRAM interface, the BRAM enable signal is only asserted during active read or write operations. The enable signal on each port to block RAM is negated when no activity exists on the AXI bus.

ECC

When C_ECC = 1, the AXI BRAM controller implements a read-modify-write on all write transfers. Read operation timing remains identical to transfers when ECC is disabled. The read-modify-write latency must be accounted for on all write transfers to block RAM.

AXI4 Timing

The timing diagrams shown in the subsequent sections represent the timing relationships of the AXI slave BRAM controller IP connection to the Xilinx AXI Interconnect. All write operations are initiated on the Write Address Channel (AW) of the AXI bus which specifies the type of write transaction and the corresponding address information. The handshaking protocol follows a Valid and Ready mechanism. All address and control information is only valid when the Valid signal is asserted. When the slave asserts the Ready signal, it captures the signals and accepts the operation. The Write Data Channel (W) communicates all write data for the single or burst write operation. The Write Response Channel (B) is used as the handshaking or response on the write operation.

On read operations, the Read Address Channel (AR) communicates all address and control information when the AXI master asserts the Valid signal. The slave IP (or AXI BRAM controller) asserts the Ready signal on the RAC when the read operation can be processed. When the read data is available to send back to the AXI master, the Read Data Channel (R) translates the data and status of the operation.

AXI4 Operations

Single Write

Figure 11 illustrates an example for the timing of an AXI single 32-bit write operation to a 32-bit wide BRAM. This example illustrates the single write to BRAM address 0x1000h, provided that C_S_AXI_BASEADDR is set to 0xA000 0000 and the C_S_AXI_HIGHADDR allows space for more than 4k of addressable BRAM.

As recommended, the AXI BRAM controller keeps the AWREADY signal asserted on the bus, as the address can be captured in the clock cycle when the S_AXI_AWVALID and S_AXI_AWREADY signals are both asserted. Once the write address pipeline (two deep) is full, then the slave BRAM controller negates the AWREADY registered output signal.

The same principle applies to the write data channel, WVALID and WREADY signals. The BRAM controller negates the WREADY signal when the data pipeline writing to block RAM is full. This condition may occur when the BRAM controller is processing a prior burst write data operation.

When ECC is enabled on full data width BRAM write transfers, the timing of the transaction is identical to when C_ECC = 0.

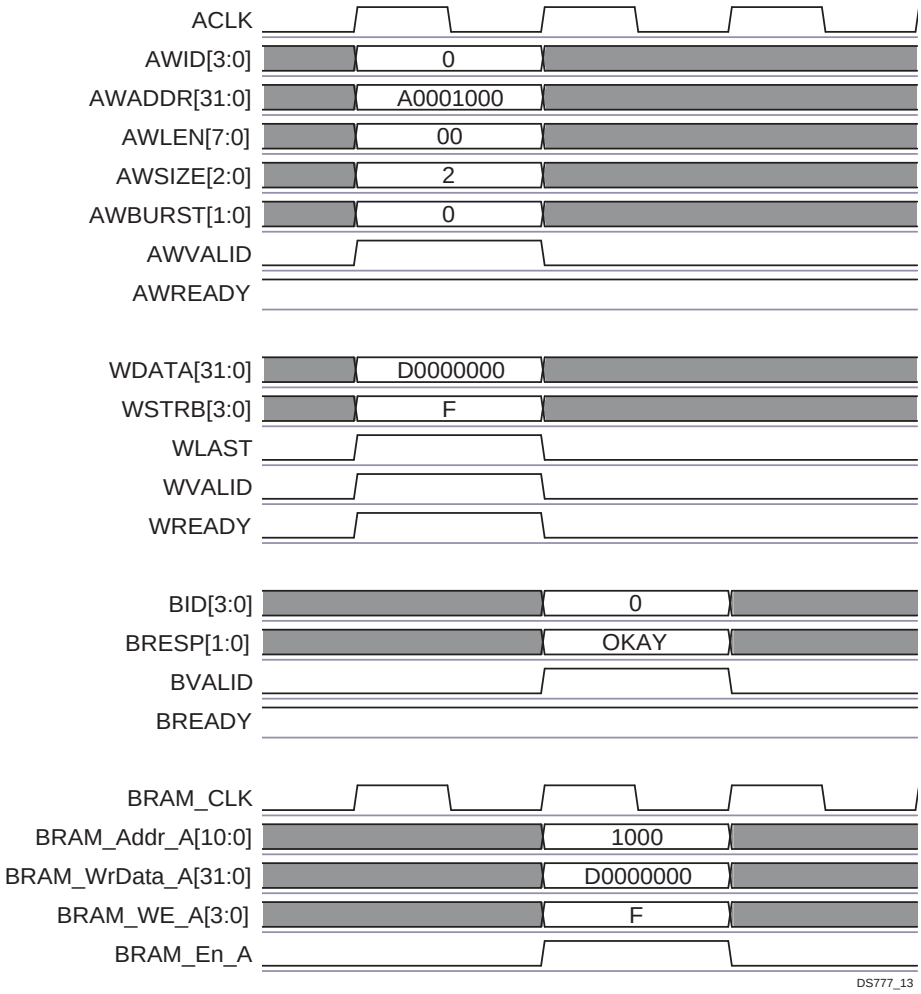

Figure 11: **AXI Single Write Timing Diagram**

It is possible on the write data channel for the data to be presented to the AXI BRAM controller prior to the write address channel (AWVALID). In this case, the AXI BRAM controller does not initiate the write transactions (the write data is ignored), until the write address channel has valid information for the BRAM controller to accept.

Single Read

Figure 12 illustrates an example of the timing for an AXI single read operation from a 32-bit BRAM.

The registered ARREADY signal output on the AXI Read Address Channel interface defaults to a high assertion. The AXI BRAM controller can accept the read address in the same clock cycle as the ARVALID signal is first valid. The BRAM controller registers in the read address when the ARVALID and the ARREADY signals are both asserted. When the BRAM controller read address pipeline is full (two-deep), it clears the ARREADY signal until the pipeline is in a non full condition.

The AXI BRAM controller can accept the same clock cycle assertion of the RREADY by the master, if the master can accept data immediately. When the RREADY signal is asserted on the AXI bus by the master, the BRAM controller negates the RVALID signal.

For single read transactions, the RLAST is asserted with the RVALID by the AXI BRAM controller.

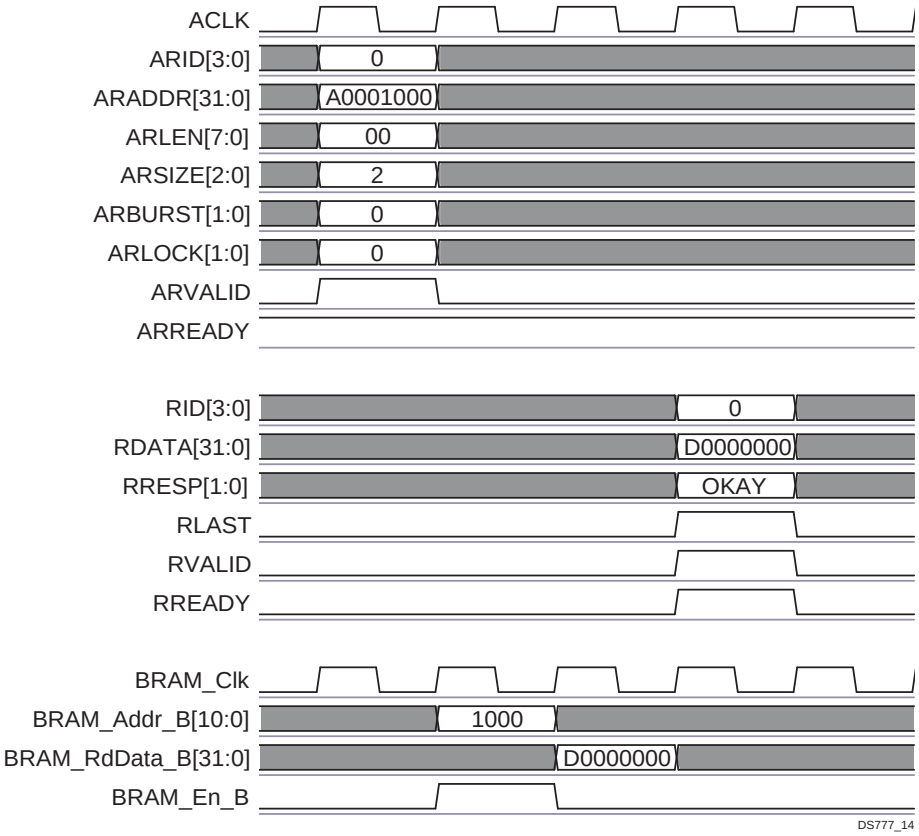

Figure 12: **AXI Single Read Timing Diagram**

On AXI read transactions, the read data always follows the read address handshaking. The AXI BRAM controller does not assert RVALID until both the ARVALID and ARREADY signals are asserted in the same clock cycle. In other words, there is no ability for early access to the BRAM controller nor internal caching ability in the BRAM controller.

AXI Burst Operations

Write Burst

Figure 13 illustrates an example of the timing for an AXI write burst of four words to a 32-bit BRAM. The address write channel handshaking stage communicates the burst type as INCR, the burst length of 4 data transfers (AWLEN = 0011b). The write burst utilizes all byte lanes of the AXI data bus to the block RAM (AWSIZE = 010b). The write burst shown in Figure 13 is set to start at BRAM address 0x1000h, provided that the C_S_AXI_BASEADDR design parameter is set to 0xA000 0000 and the C_S_AXI_HIGHADDR allows space for more than 4k of addressable block RAM.

On the AXI write transactions, the slave does not wait for the write data channel, WVALID signal to be asserted prior to the assertion of the write address channel signal, AWREADY. This could potentially cause a deadlock condition and is not allowed.

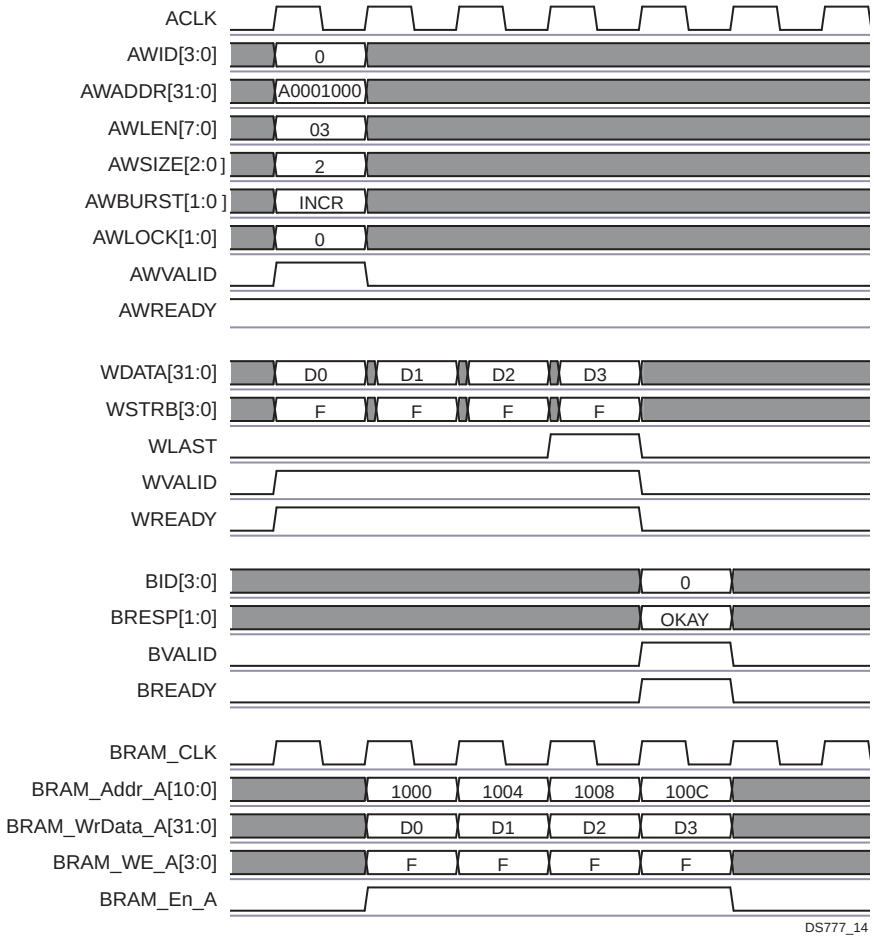

Figure 13: **AXI Burst Write Timing Diagram**

Narrow Write Bursting

Figure 14 illustrates an example of the BRAM controller supporting a *narrow* burst operation. A *narrow* burst is defined as a master bursting a data size smaller than the BRAM data width. If the burst type (AWBURST) is set to INCR or WRAP, then the valid data on the BRAM interface to the AXI bus rotates for each data beat. The BRAM controller handles each data beat on the AXI as a corresponding data beat to the block RAM, regardless of the smaller valid byte lanes. In this scenario, the AXI WSTRB is translated to the BRAM write enable signals. The BRAM address only increments when the full address (data) width boundary is met with the *narrow* write to block RAM.

Figure 14 illustrates an example of AXI *narrow* bursting with a 32-bit BRAM and the AXI master request is a halfword burst of 4 data beats. AWSIZE is set to 001b.

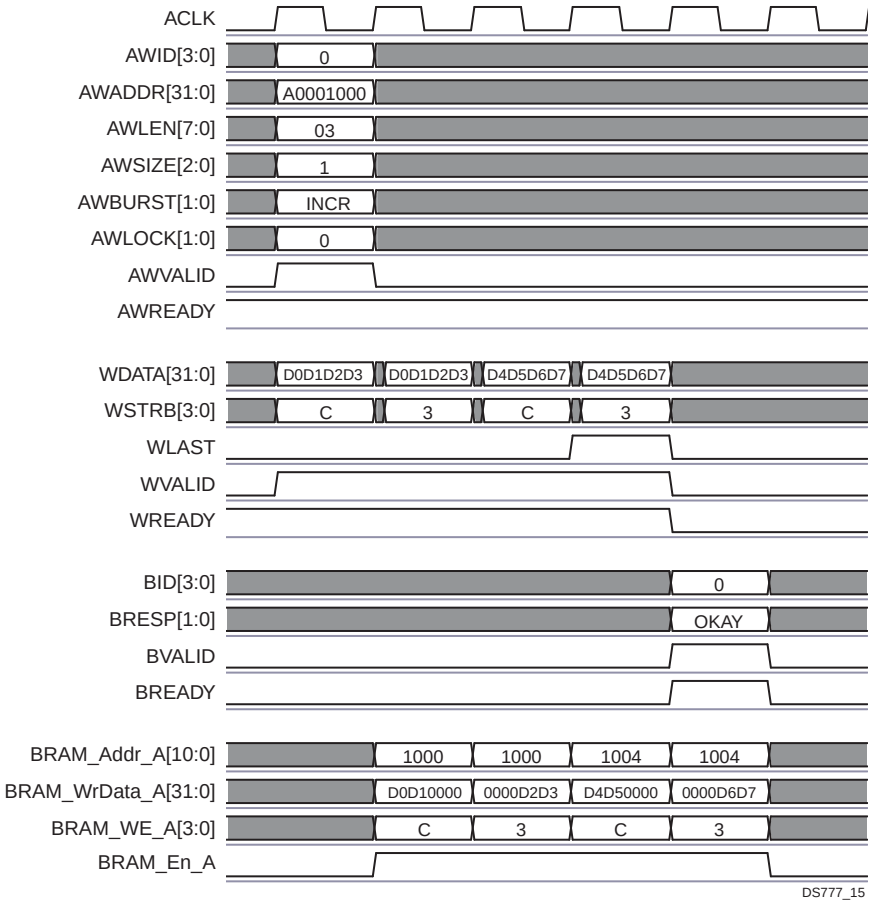

Figure 14: **AXI Narrow Burst Write Diagram**

Unaligned Write Bursting

The AXI BRAM controller supports unaligned burst transfers. Unaligned burst transfers for example, occur when a 32-bit word burst size does not start on an address boundary that matches a word memory location. The starting memory address is permitted to be something other than 0x0h, 0x4h, 0x8h, etc.

The example shown in Figure 15, illustrates an unaligned word burst transaction of 4 data beats, that starts at address offset, 0x1002h (provided that C_S_AXI_BASEADDR is set to 0xA000 0000 and C_S_AXI_HIGHADDR allows more than 4k of addressable memory). The associated timing relationship is shown is Figure 16. Note the unaligned address corresponds to the BRAM_WE signals on the write port to reflect the valid byte lanes.

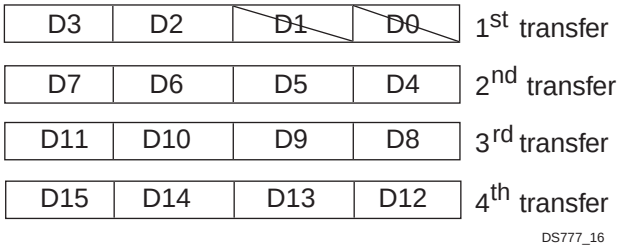

Figure 15: **AXI Unaligned Burst Write Transfers**

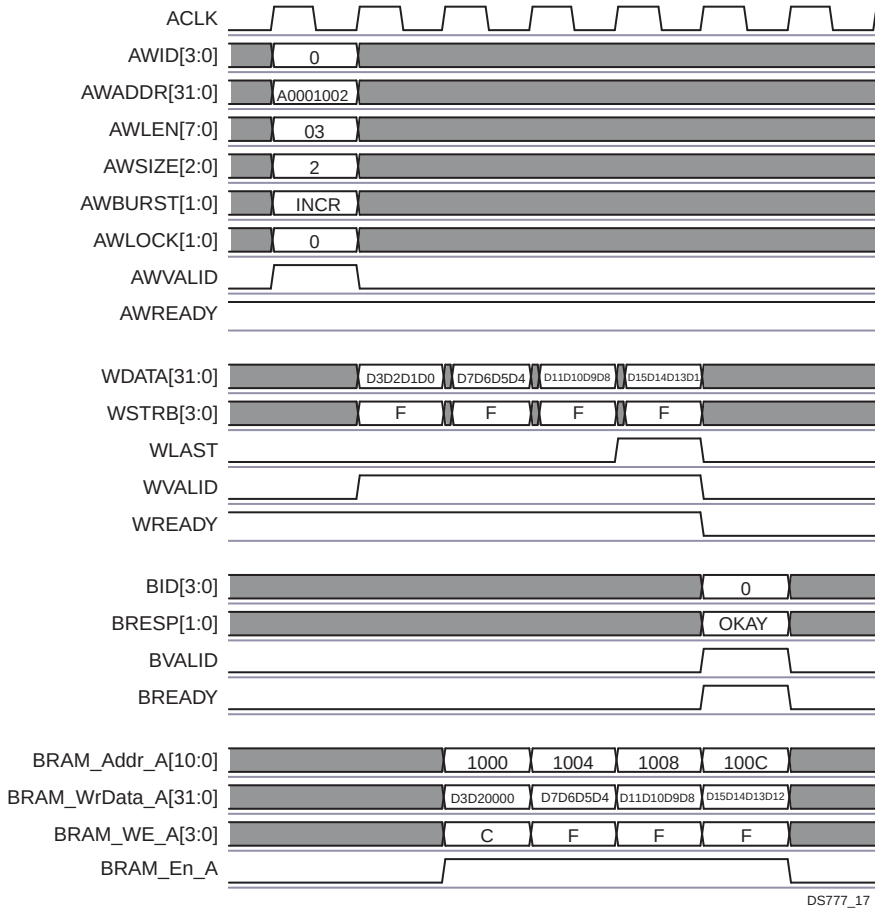

Figure 16: **AXI Unaligned Burst Write Timing**

ECC Write Burst

Figure 17 illustrates the timing for an AXI write burst of two data beats. Note the additional latency due to the read-modify-write sequence required to correctly update the ECC check bits in block RAM.

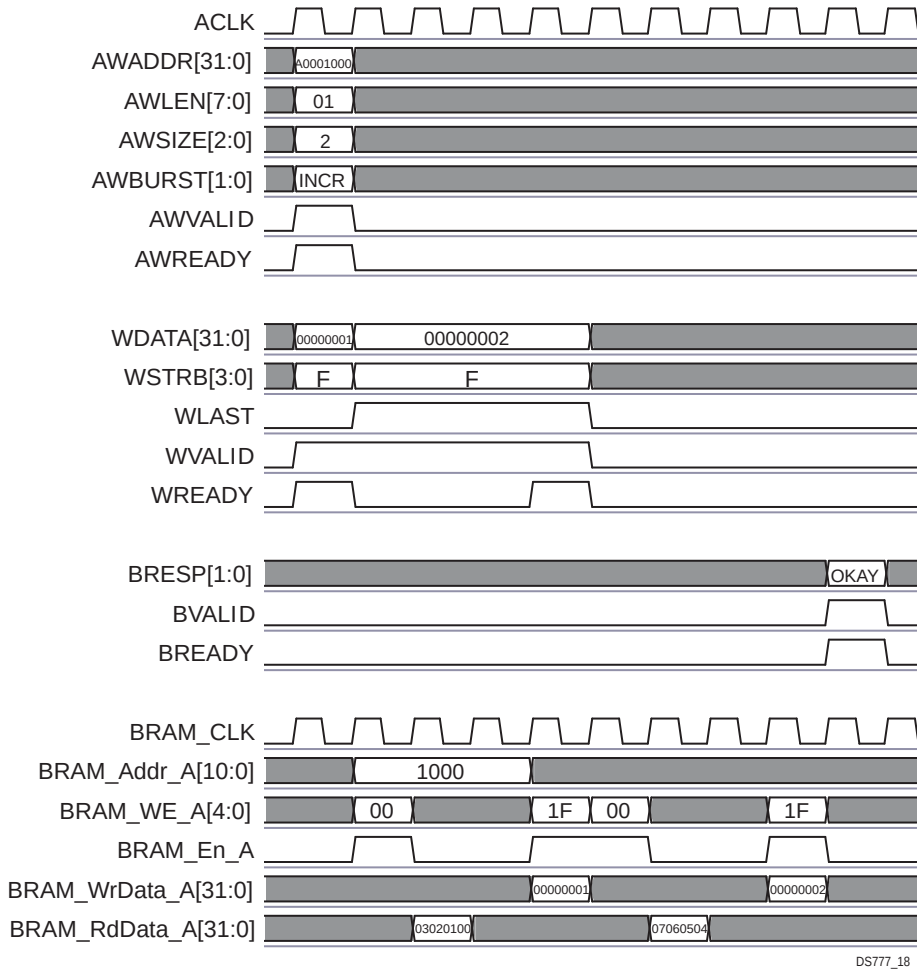

Figure 17: **AXI Write Burst w/ ECC**

Write Pipeline

The AXI BRAM controller IP core is capable of supporting up to two active write operations. Once the write address pipeline is full, the controller negates AWREADY to the bus. Only when the first pipelined operation is complete, may a new write transfer occur subsequently (indicated by the assertion of AWREADY).

The AXI BRAM controller does not support write data interleaving, so the data on the write data channel must be in the same order as the addresses (for that data) presented on the write address channel.

The write data channel responds to the write transfer with the assertion of WREADY. The BRAM controller accepts data when the write data channel is not already busy from a previous transaction. The BRAM controller supports the early assertion of WREADY to WVALID when asserted prior to AWVALID and AWREADY. The BRAM controller only accepts one data transfer before the write address is accepted.

The BRAM controller can accept data on the write data channel in the same clock cycle when the write address is valid and accepted by the controller. The WREADY signal remains asserted to continually accept the burst, as the BRAM address counter is loaded and the write data can pass to block RAM.

The AXI BRAM controller ensures that BREADY is not asserted prior to WVALID and WREADY.

Figure 18 illustrates the capability to pipeline write addresses in the BRAM controller. This example illustrates when a gap is seen on the write data channel by the BRAM controller from the AXI Interconnect.

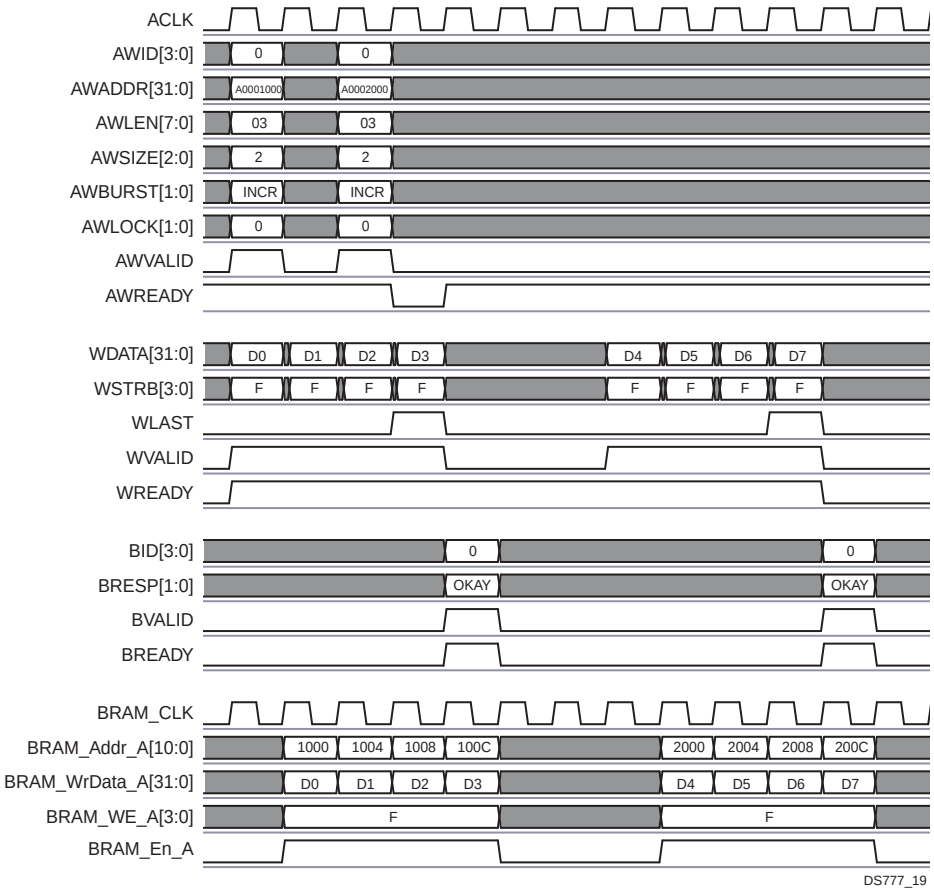

Figure 18: **AXI Pipelined Burst Write Transfers**

The AXI BRAM controller can support back-to-back write burst operations if supplied with a continuous data stream from the AXI Interconnect. In this case, there are no idle clock cycles on the BRAM interface between the two pipelined write burst operations. Figure 19 illustrates the timing for back-to-back pipelined write bursts of four data beats.

To achieve 100 % BRAM interface utilization on the write port the following conditions must be satisfied.

- No single write bursts.
- Write burst must be greater than two data beats.
- Write burst operation must be an INCR or WRAP burst type.

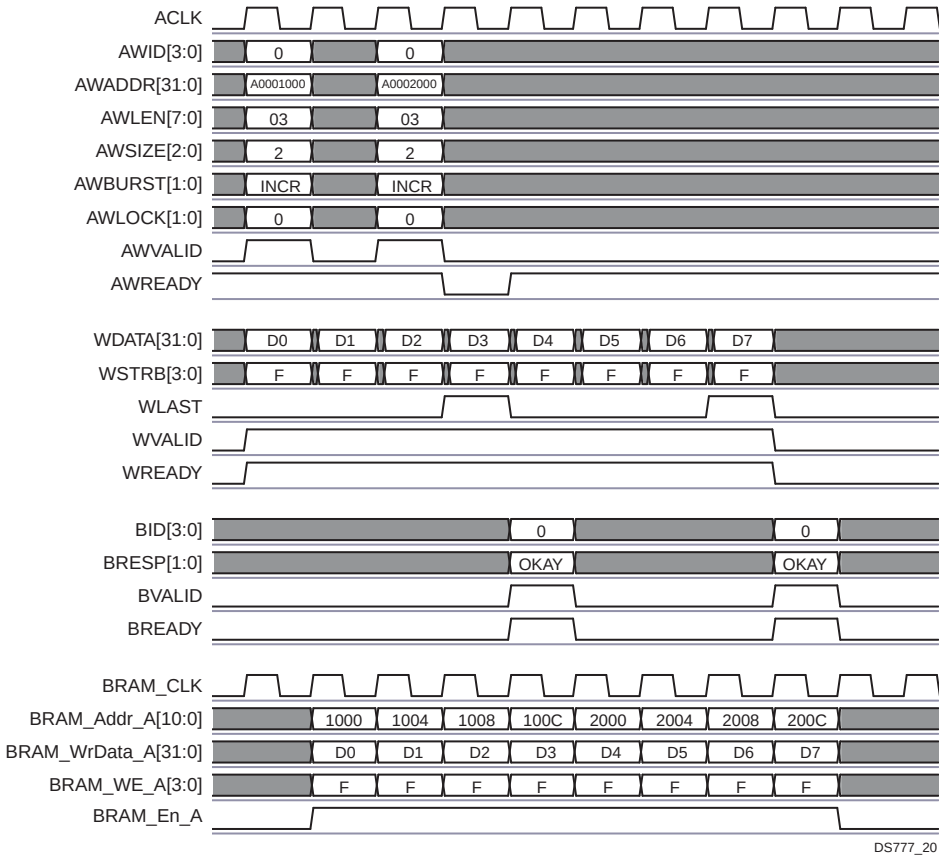

Figure 19: **AXI Back to Back Write Burst Timing**

Delayed Write Address

The BRAM controller throttles the write data after one data beat prior to a provided valid write address. The BRAM controller only accepts early WVALID and WDATA when configured in a dual port mode when ECC functionality is disabled. The scenario timing is shown in Figure 20.

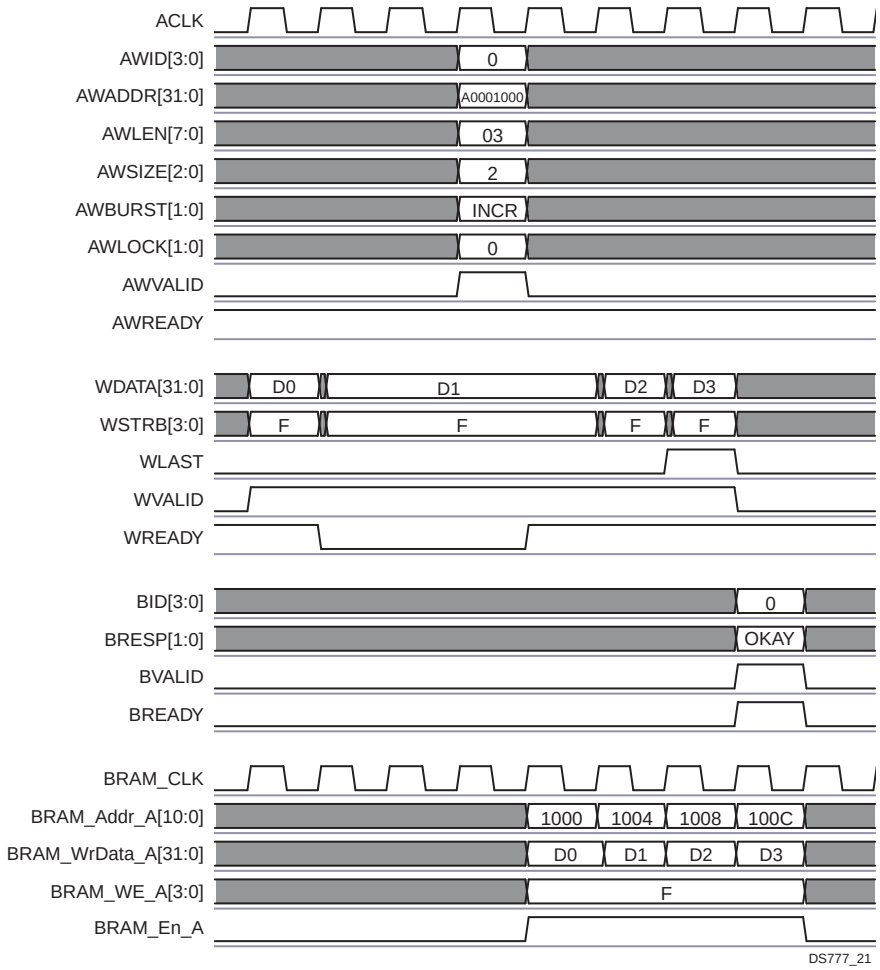

Figure 20: **AXI Delayed Write Address**

Read Bursting

Figure 21 illustrates the example timing for an AXI read burst with block RAM handled by the BRAM controller. The memory read burst starts at address 0x1000h of the block RAM, provided C_S_AXI_BASEADDR = 0xA000 0000 and C_S_AXI_HIGHADDR allows more than 4k of addressable memory. The AXI Read Address Channel interface keeps the ARREADY signal asserted until the read address pipeline is full in the BRAM controller. On the AXI Read Data Channel, the BRAM controller supports the AXI master/Interconnect to respond to the RVALID assertion with a same clock cycle assertion of RREADY. If the requesting AXI master/Interconnect throttles upon accepting the read burst data (by negating RREADY), the BRAM controller can manage this and holds the data pipeline until RREADY is asserted.

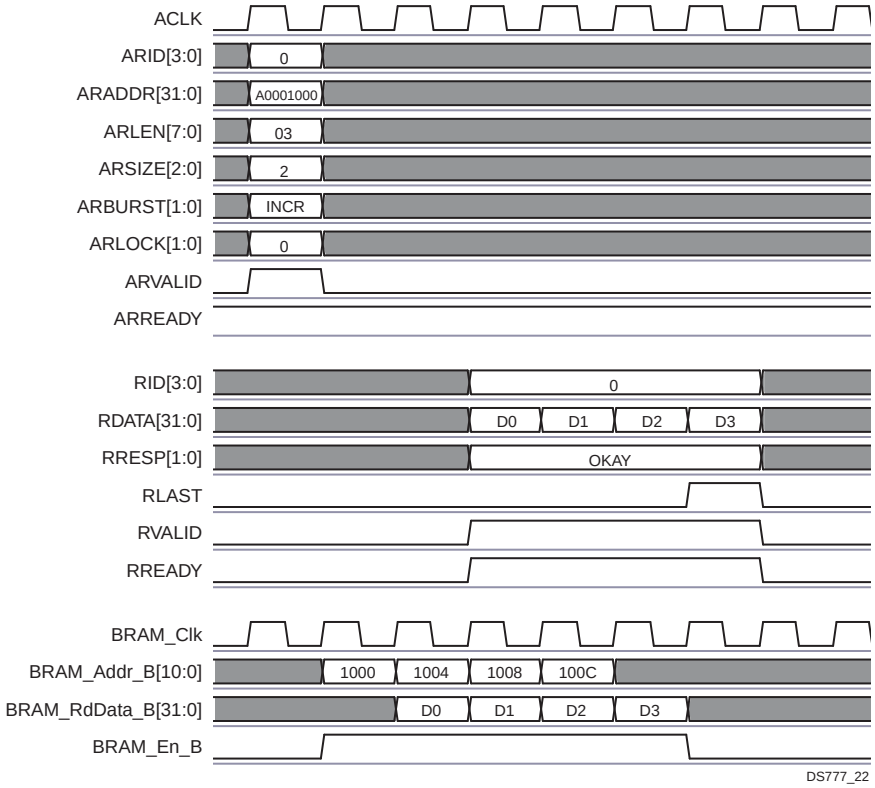

Figure 21: **AXI Read Burst Diagram**

Read Throttling

The AXI BRAM controller must support read throttling. During a read operation, the AXI BRAM issues read commands to the block RAM, but can only read ahead two addresses (the amount of BRAM read data beats supported in the BRAM controller read data skid buffer pipeline). The requesting AXI master is not required to capture all the data immediately, but might throttle and only assert the RREADY signal when data can be accepted. The BRAM controller must halt the read operation and hold existing read data when the requesting master negates RREADY. Figure 22 illustrates this behavior and corresponding BRAM port operation. The two stage read data pipeline ensures that all outputs to block RAM and outputs to the AXI read data channel are registered.

The behavior shown in Figure 22 reflects the condition when the master waits to assert RREADY until RVALID is asserted. The BRAM controller can accept the master assertion of RREADY prior to the assertion of RVALID. Both signals must be asserted to advance the read data skid buffer pipeline in the BRAM controller.

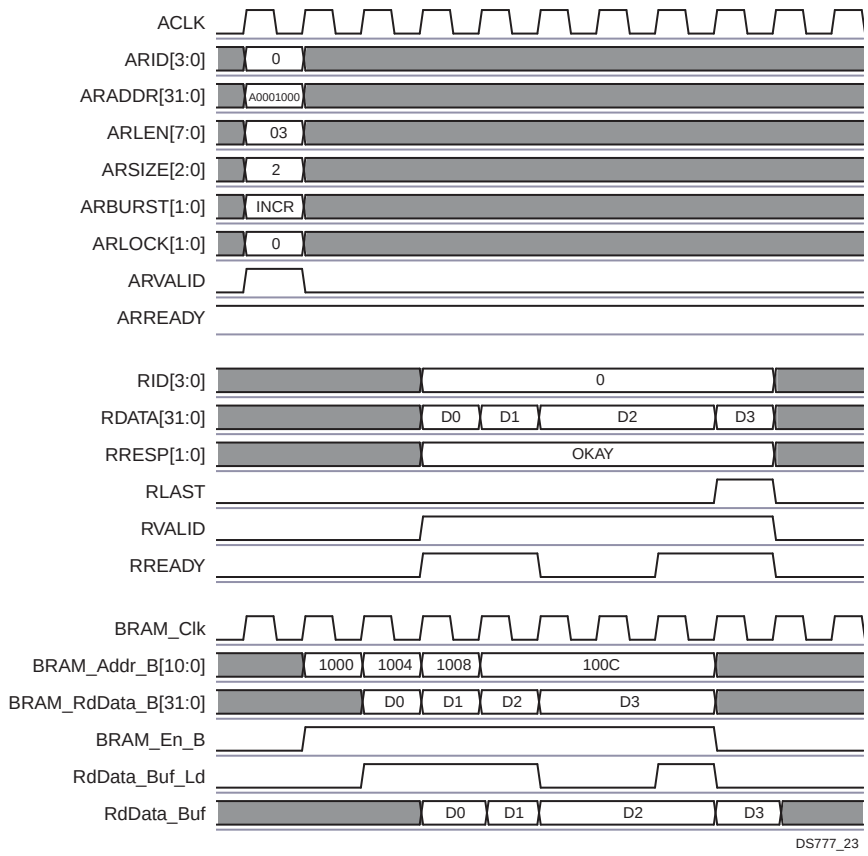

Figure 22: **AXI Read Burst Throttling**

Read Address Pipeline

Figure 23 and Figure 24 illustrate examples of the timing for pipelined read burst operations. The AXI BRAM controller can handle pipelined read addresses as a continuous burst to block RAM. The master of the pipelined read operation can accept data in the clock cycle following the assertion of RLAST (from the prior read operation) under the following conditions:

- The read operation is not a single data beat transfer.
- The read burst is greater than two data beats.
- The AXI burst operation size is equal to the data port size of the AXI Interconnect.

- The requesting burst type is of type INCR or WRAP.
- And, no throttling is detected (on the AXI read channel) for the current read burst after the second to last BRAM address is registered out to memory.

A continuous read burst to block RAM is supported in both a dual port configuration (using the second port to block RAM), or in a single port BRAM configuration (pending no active pending write transfers).

If any of these previously cited conditions exist on the pipelined read operations, the master must wait until RVALID is reasserted to begin reading data for the subsequent burst. The expected delay is two AXI clock cycles until RVALID is asserted after the prior RLAST (when any of these above conditions exist). Figure 24 illustrates the timing for this scenario.

Figure 23 illustrates the ability of the BRAM controller to accept pipelined read request addresses and maintain 100% bus utilization to the block RAM. The data burst must be greater than two data beats to reach a maximum 100% data throughput from the block RAM with no idle clock cycles on the AXI read data channel. The requesting burst type must be INCR or WRAP and the requesting read burst size must be equal to the size of the AXI Interconnect read data port (no "narrow" burst type transactions) to achieve 100% bus utilization on pipelined read bursts.

Utilization of the read data skid buffer illustrates the master capability to throttle on accepting read data. The resulting BRAM transaction timing is shown in Figure 24.

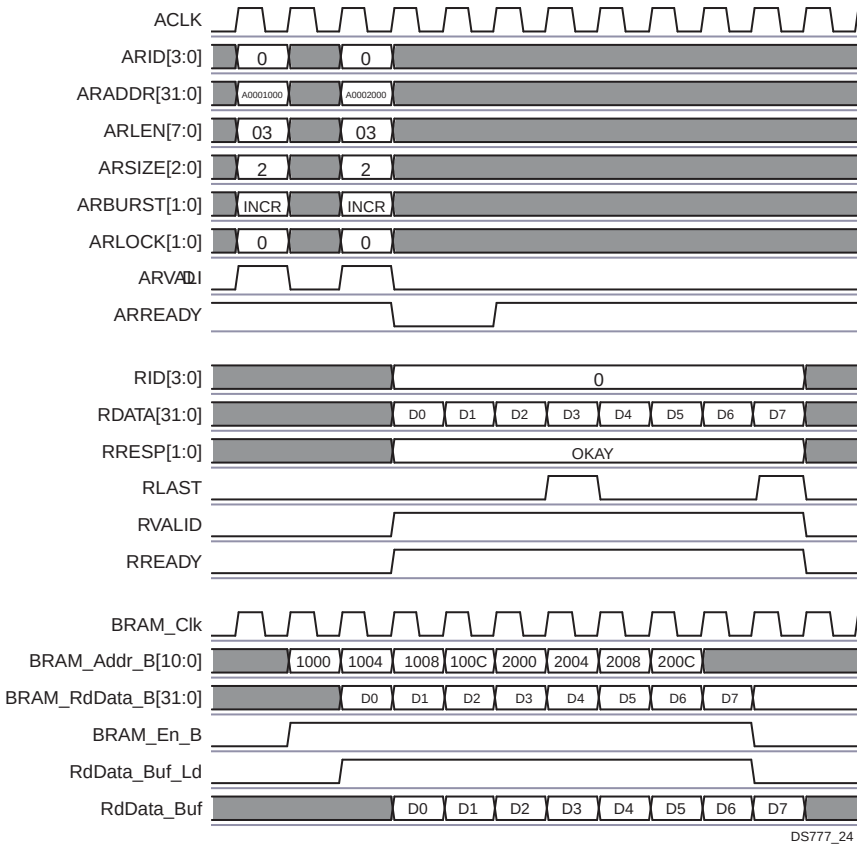

Figure 23: **AXI 100% Bus Utilization on Pipelined Read Bursts**

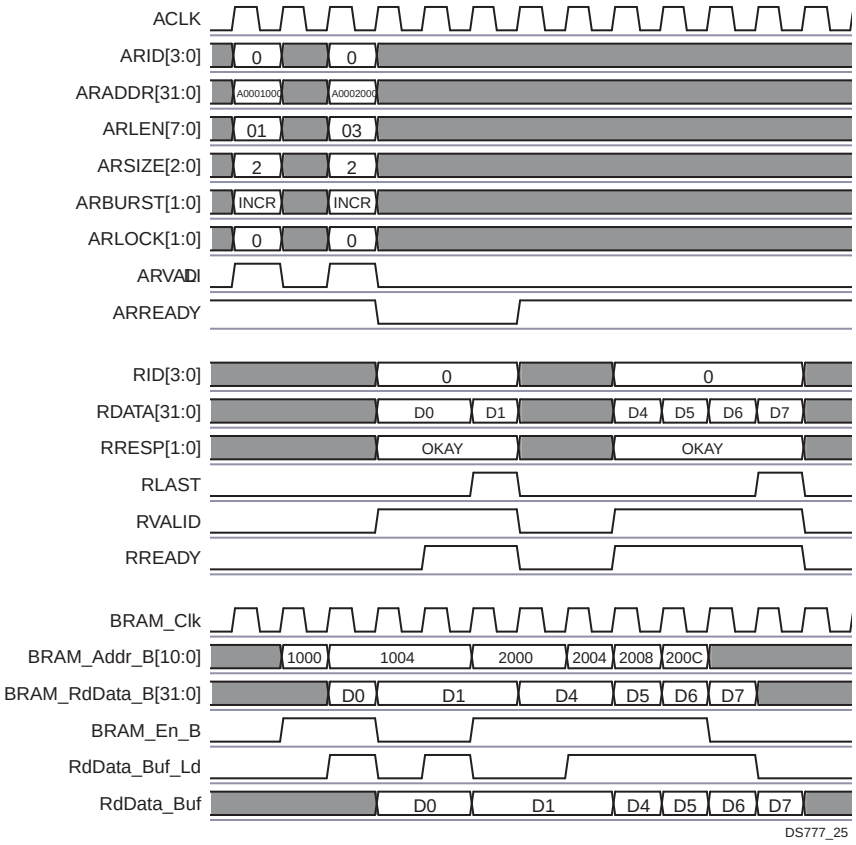

Figure 24: **AXI Read Pipeline Throttling Timing**

Cacheline Reads

Figure 25 illustrates the timing on AXI WRAP or cacheline burst transactions. The address generated to the block RAM starts at the target word and wraps around once the address boundary is reached.

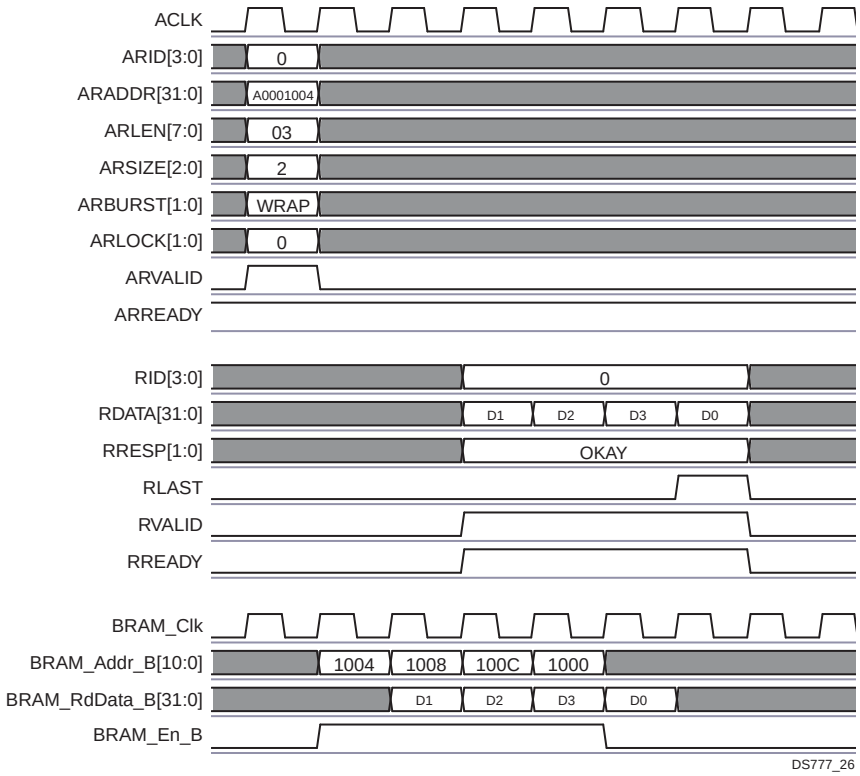

Figure 25: **AXI Cacheline Read**

Dual Port BRAM Capability

Isolating the write and read ports to block RAM with each write and read channel interface on the AXI bus, both BRAM ports can be utilized simultaneously. Figure 26 illustrates this condition to provide no arbitration on the write or read channels dependency on the other.

Single Port BRAM Capability

The AXI BRAM controller can be configured (on an AXI4 interface) to use only a single port to block RAM. In this configuration, the AXI BRAM arbitrates on the AR and AW channels and creates a single two-stage pipeline. The arbitration scheme uses a least recently used algorithm on ties of assertions between the S_AXI_ARVALID and S_AXI_AWVALID signals. For the illustration shown in Figure 27, the AW channel won arbitration and accesses the block RAM first.

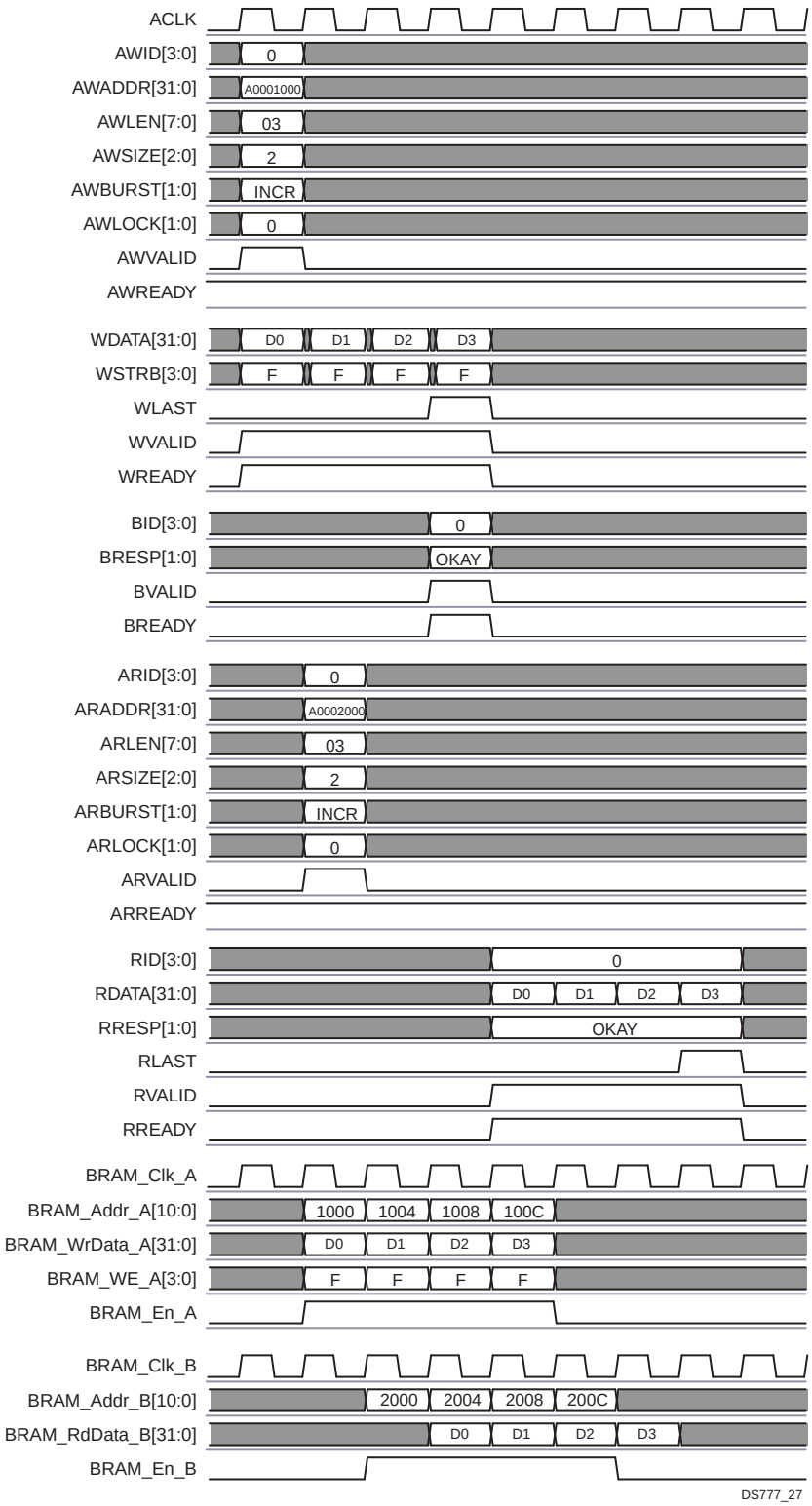

Figure 26: **AXI Dual Port BRAM Capability**

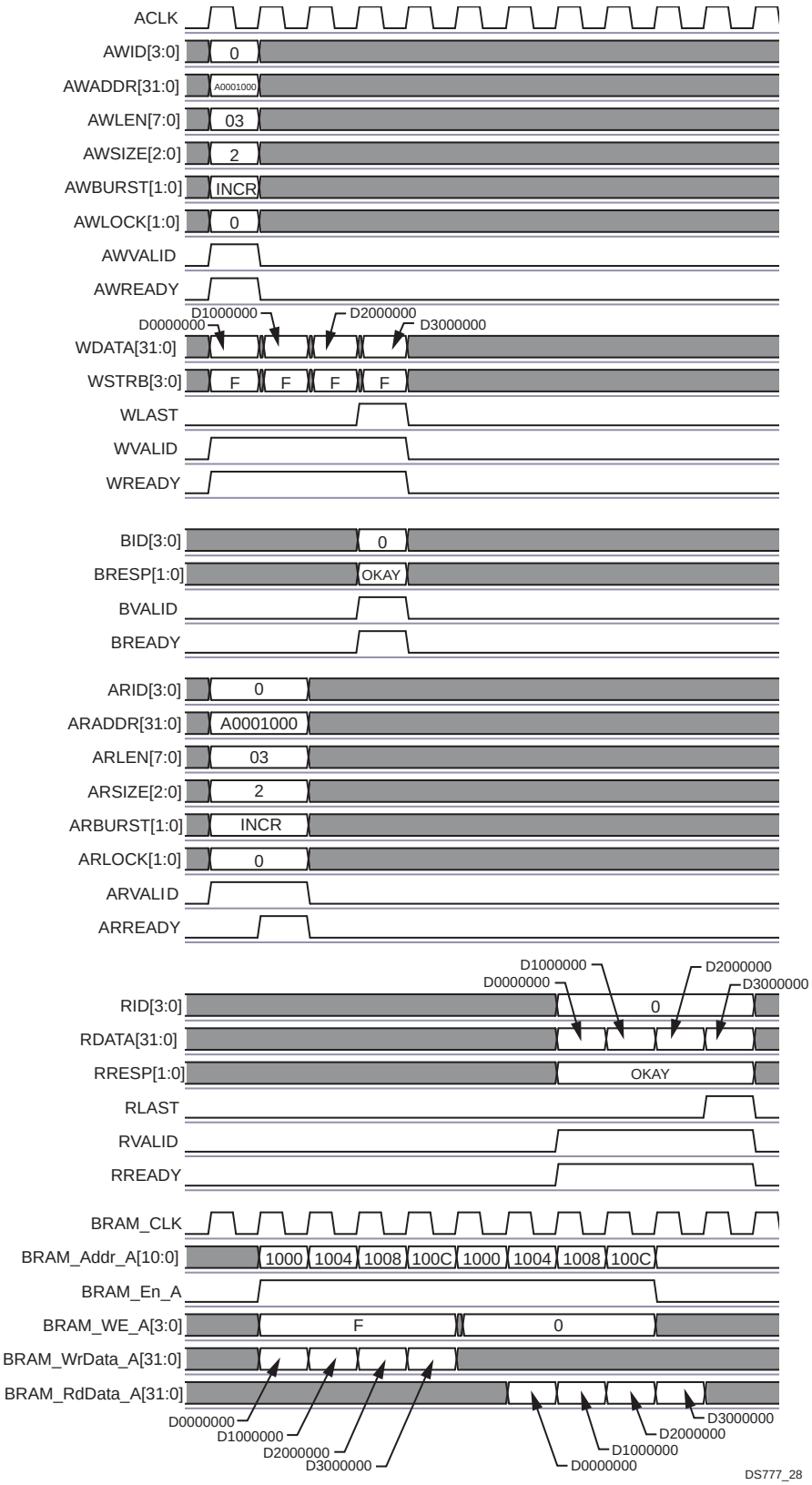

Figure 27: **AXI4 Single Port BRAM Capability**

Design Implementation

Target Technology

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

EDK Implementation

For systems requiring the AXI BRAM Controller to operate with a high frequency, the following parameters must be specified in the XPS system on the AXI BRAM Controller v1.03a MHS instantiation.

PARAMETER C_S_AXI_AW_REGISTER = 1
PARAMETER C_S_AXI_W_REGISTER = 1
PARAMETER C_S_AXI_B_REGISTER = 1
PARAMETER C_S_AXI_AR_REGISTER = 1
PARAMETER C_S_AXI_R_REGISTER = 1
Setting these parameters enables the pipeline stage on the AXI channels in the AXI Interconnect IP core.

If the AXI BRAM controller core is used in an ECC or single port implementation, a type 7 (Light_Weight) register can be enabled on the AXI channel registers in the AXI Interconnect IP core. This type of AXI interconnect register saves resources and minimizes any routing congestions, by allowing a recovery clock cycle between back-to-back transfers.

For additional timing improvements in AXI4 systems with stringent F_{MAX} requirements, the AXI BRAM IP core can be optimized when there are no AXI masters in the system that support *narrow* transactions to the block RAM. Setting the design parameter,

C_S_AXI_SUPPORTS_NARROW_BURST = 0

in the MHS of the EDK system improves timing paths. Logic to support the *narrow* transfers in the write and read channels of the AXI BRAM IP are optimized away by XST during synthesis of the core. The EDK tools automatically set the C_S_AXI_SUPPORTS_NARROW_BURST = 0 if it detects that all connected AXI masters have declared that they do not generate *narrow* burst transactions.

Device Utilization

Because the AXI BRAM Controller IP core is used with other design modules in the FPGA, the utilization and timing numbers reported in this section are estimates only. When the AXI BRAM controller core is combined with other designs in the system, the utilization of FPGA resources and timing of the BRAM controller IP core varies from the results reported here.

The AXI BRAM Controller resource utilization for various parameter combinations measured with the Virtex-6 FPGA (XC6VLX240T) as the target device is detailed in Table 69.

Table 69: **Resource Utilization Benchmarks on the Virtex-6 FPGA**

Parameter Values (Other parameters at default values)					Device Resources		
C_S_AXI_PROTOCOL	C_BRAM_DWIDTH	C_S_AXI_SUPPORTS_NARROW_BURST	C_SINGLE_PORT_BRAM	C_ECC	Slices	Slice Registers	LUTs
AXI4	32	0	0	0	221	390	405
AXI4	64	0	0	0	216	484	445
AXI4	128	0	0	0	276	684	540
AXI4	256	0	0	0	359	1078	608
AXI4	512	0	0	0	435	1880	998
AXI4	32	0	1	0	144	260	248
AXI4	64	0	1	0	149	260	248
AXI4	128	0	1	0	167	358	269
AXI4	256	0	1	0	168	358	277
AXI4	32	0	0	1	252	444	577
AXI4	64	0	0	1	318	606	778
AXI4	32	0	1	1	198	342	445
AXI4	64	0	1	1	241	497	550
AXI4LITE	32	N/A	0	0	9	8	12
AXI4LITE	32	N/A	1	0	24	9	44
AXI4LITE	32	N/A	0	1	67	147	170
AXI4LITE	32	N/A	1	1	66	147	168

Notes:

1. The number of block RAMs used depends on parameter settings for C_S_AXI_BASEADDR, C_S_AXI_HIGHADDR, and C_S_AXI_DATA_WIDTH. Please refer to Table 6 for more details.

The maximum (-1 speed grade) clock frequency on the AXI BRAM Controller IP core with an AXI4 IF in the Virtex-6 FPGA architecture (-1 speed grade) is 180 MHz.

For an AXI4-Lite system, the maximum target clock frequency in a Virtex-6 FPGA is 135 MHz.

The AXI BRAM Controller resource utilization for various parameter combinations measured with the Spartan-6 FPGA (XC6SLX45T) as the target device are detailed in Table 70.

Table 70: **Resource Utilization Benchmarks on the Spartan-6 FPGA**

Parameter Values (Other Parameters at Default Values)					Device Resources		
C_S_AXI_PROTOCOL	C_BRAM_DWIDTH	C_S_AXI_SUPPORTS_NARROW_BURST	C_SINGLE_PORT_BRAM	C_ECC	Slices	Slice Registers	LUTs
AXI4	32	0	0	0	236	404	467
AXI4	64	0	0	0	244	498	501
AXI4	32	0	0	1	236	451	534
AXI4	64	0	0	1	329	615	732
AXI4LITE	32	N/A	0	0	4	2	8
AXI4LITE	32	N/A	1	0	17	2	24

Notes:

1. The number of block RAMs used depends on parameter settings for C_S_AXI_BASEADDR, C_S_AXI_HIGHADDR, and C_S_AXI_DATA_WIDTH. See Table 7 for more details.

The maximum target clock frequency on the AXI BRAM Controller IP core with an AXI4 interface in the Spartan-6 FPGA architecture (-2 speed grade) is 120 MHz.

For an AXI4-Lite system, the maximum target clock frequency in a Spartan-6 FPGA is 90 MHz.

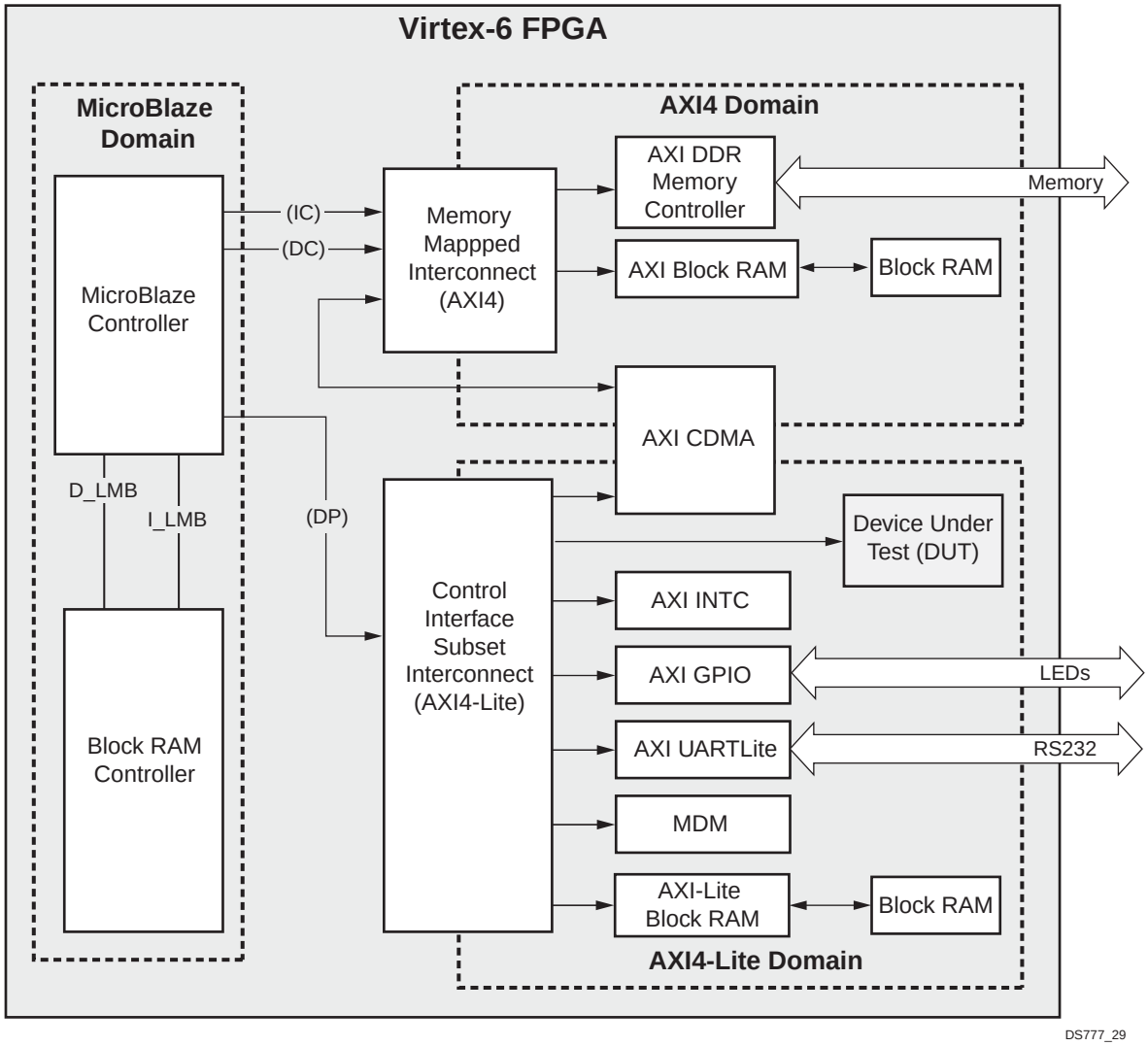

Figure 28: **Virtex-6 FPGA System with the AXI BRAM Controller as the DUT**

The target FPGA was 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 71.

Table 71: **System Performance**

Target FPGA	Target F_{MAX} (MHz)		
	AXI4	AXI4-Lite	MicroBlaze
xc6slx45t (1)	90 MHz	120 MHz	80
xc6vlx240t (2)	135 MHz	180 MHz	135

Notes:

1. Spartan-6 FPGA LUT utilization: 60%; BRAM utilization: 70%; I/O utilization: 80%; MicroBlaze Controller not AXI4 interconnect; AXI4 interconnect configured with a single clock of 120MHz.
2. Virtex-6 FPGA LUT utilization: 70%; BRAM utilization: 70%; I/O utilization: 80%.

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

Reference Documents

These documents provide information useful to this data sheet. To locate the Xilinx documents, go to www.xilinx.com/support.

1. AMBA® AXI Protocol Version: 2.0 Specification (ARM IHI 0022C)
2. AXI Interconnect IP Data Sheet (DS768)
3. Virtex-6 Family Overview (DS150)
4. Spartan-6 Family Overview (DS160)
5. Virtex-6 FPGA Memory Interface Solutions User Guide (UG406)
6. IBM Technical Journal. M. Y. Hsiao. A Class of Optimal Minimum Odd-weight-column SEC=DED Codes. July 1970.

For a glossary of technical terms used in Xilinx documentation, see http://www.xilinx.com/support/documentation/sw_manuals/glossary.pdf.

Support

Xilinx provides technical 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*.

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), ISE Design Suite or Vivado Design Suite.

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	Description of Revisions
9/21/10	1.0	Xilinx initial release.
12/14/10	1.1	Updated for 12.4 release; new version of HDL, v1.01a; includes support for AXI4-Lite interface connections with a reduced footprint optimization. In an AXI4-Lite mode, the BRAM port can be configured for single or dual port access. Also, C_S_AXI_BASEADDR and C_S_AXI_HIGHADDR have been moved as non HDL parameters.
3/1/11	1.2	Updated for the 13.1 release; added ECC logic and register AXI-Lite interface; new version of HDL.
6/22/11	1.3	Updated for the 13.2 release; added larger BRAM data widths; updated ECC Hsiao algorithm details.
10/19/11	1.4	Updated for the 13.3 release. Added support for 128-bit ECC.
07/25/12	1.5	Updated for the 2012.2 and 14.2 releases. Added support for Vivado Design Suite. Added support for Zynq-7000 devices.

Notice of Disclaimer