AXI EMC v3.0

LogiCORE IP Product Guide

Vivado Design Suite

PG100 April 5, 2017

Table of Contents

Introduction

The LogiCORE™ IP AXI External Memory Controller (EMC) is a soft Xilinx® IP core for use with external memory devices. The adaptable block provides memory controller functionality for SRAM, NOR Flash and PSRAM/CellularRAM memory devices. The core provides an AXI4 Slave Interface that can be connected to AXI4 Master or Interconnect devices in the AXI4 Systems.

Features

- Supports AXI4 Slave Memory Map interface data width of 32 and 64 bits
- Supports optional AXI4-Lite Slave data width of 32 bits for Write/Read registers
- Supports AXI4 increment and wrap transactions
- Supports AXI4 narrow and unaligned transfers
- Supports up to four external memory banks
- Supports Synchronous SRAMs with configurable byte parity check and pipeline stages
- Supported memory types
  - Synchronous SRAM
  - Asynchronous SRAM
  - Linear Flash (or Parallel NOR Flash)
  - PSRAM (or Cellular RAM)
- Provides configuration registers to dynamically change access mechanism for PSRAM and Micron® Flash memories
- Provides Parity error status register for Synchronous SRAM memories

LogiCORE IP Facts Table	
Core Specifics	
Supported Device Family[1]	UltraScale+™ Families, UltraScale™ Architecture, Zynq®-7000 All Programmable SoC, 7 Series
Supported User Interfaces	AXI4-Lite, AXI4
Resources	See Table 2-2 to Table 2-4.
Provided with Core	
Design Files	VHDL
Example Design	VHDL
Test Bench	VHDL
Constraints File	XDC
Simulation Model	N/A
Supported S/W Driver[2]	Standalone
Tested Design Flows[3]	
Design Entry	Vivado® Design Suite
Simulation	For a list of supported simulators, see the Xilinx Design Tools: Release Notes Guide
Synthesis	Vivado Synthesis
Support	
Provided by Xilinx at the Xilinx Support web page	

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

Overview

The architectural block diagram of the AXI External Memory Controller (EMC) core is shown in Figure 1-1.

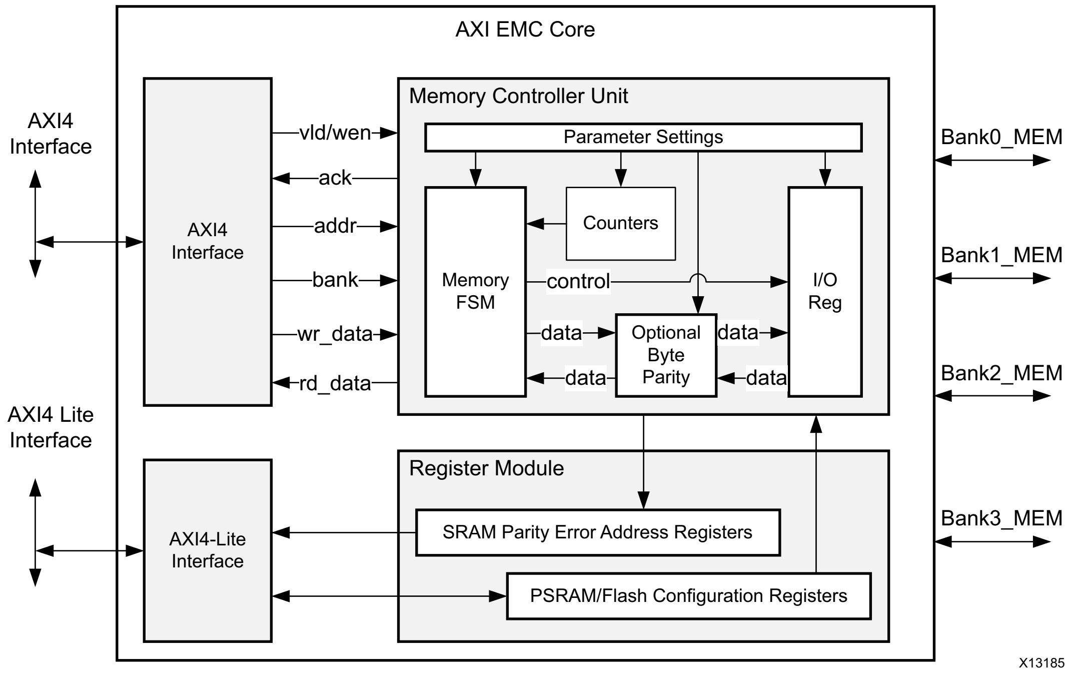


Figure 1-1: **Top Level Block Diagram of the AXI EMC Core**

The AXI EMC core provides the memory controller functionalities for PSRAM, flash, and SRAM technologies. The controller supports memory devices from multiple vendors and also allows interfacing to multiple of these memory technologies (or types) by supporting up to four memory banks. The core provides an AXI4 Slave interface for addressing and accessing data from the connected memory devices.

The core also provides optional AXI4-Lite interface for configuring access mechanisms as defined by the connected memory types. In addition, the controller provides a set of core generation parameters which defines the memory interface timing for the connected memories.

Module Descriptions

The AXI EMC operations can be categorized into seven modules:

- AXI4 Slave Interface
- AXI4-Lite Interface
- Memory Controller Unit
- Byte Parity Logic
- EMC Register
- I/O Registers
- Flash (Memory) Access through STARTUPE Primitive

AXI4 Slave Interface

AXI EMC core provides AXI4 slave interface to map to AXI4 master or AXI4 interconnect devices in the FPGA logic. The AXI4 slave interface of the core complies with AMBA® AXI4 protocol specifications for 32-bit and 64 bit data widths. The core supports single beat or burst AXI4 transactions. The burst transactions for incremental bursts [INCR] can be from 1 beat to 256 beats and the burst transactions for wrapping burst [WRAP] can be 2, 4, 8, or 16 beats. The core also supports AXI4 Narrow and AXI4 Unaligned transfers.

The AXI4 interface of the core also performs the channel arbitration and address decoding functionalities. The channel arbitration implements a round robin algorithm and is applicable only when both AXI4 write channels and AXI4 read channels are active simultaneously. The address decoding logic utilizes the AXI4 address to determine the bank address and read/write address in the selected memory bank. For more details on Memory Address Generation, see Address Map Description in Chapter 3.

The performance of AXI EMC core gets determined by Memory Width, Memory access latencies and AXI burst size. In general larger AXI4 burst sizes can increase the overall performance of the core. For more details on EMC core performance, see Performance in Chapter 2.

Note: The AXI4 FIXED transactions are not supported by the core and when issued can result in nondeterministic core behavior.

AXI4-Lite Interface

AXI EMC core provides AXI4-Lite slave interface to allow access to Internal Control and Status registers of the core. The AXI4 slave interface of the core complies with AMBA AXI4-Lite protocol specifications for 32-bit data width. The AXI4-Lite interface of the core is optional and is enabled only when the **Enable Internal registers** option is set. For more details on Core registers accessible through the AXI-Lite interface, see EMC Register.

Memory Controller Unit

The Memory Controller Unit performs Reads/Writes to the external memories in compliance with access mechanism defined for the selected memory. It generates the memory chip select (or bank select) and the control signals associated with the memory in the selected bank. The Read/Write Address, Write Data and the Bank Address generated for the memory is as provided by the AXI4 interface module to the Memory Controller Unit.

The Memory Controller Unit in addition aligns the address and data received from the AXI4 interface module to match the address and data width configured for the bank. Different data widths for various banks are now only supported when selected memory is Synchronous/Asynchronous SRAM. For other memory types, data widths have to be same across banks. The width conversion in this case is performed dynamically as per the width defined for each memory bank.

The Memory interface timing for the asynchronous memory interfaces are parameterized and can be set independently for each bank. The Memory Controller Unit maintains internal counters and registers to match the interface timing as defined by these parameters. For more details on memory timing parameters, see Customizing and Generating the Core in Chapter 4. In addition for configuration registers associated with PSRAM/Flash memories, see Register Description in Chapter 2. For example memory interfaces and associated memory settings, see Connecting to Memory in Chapter 3.

Byte Parity Logic

The Byte Parity Logic is applicable only for the SRAM type and is used to calculate the parity bit for each data byte written to or read from the memory. This parity bit is attached to the data byte and is then written to or read from the memory. The parity calculation logic is valid only when **Parity** option is set to either **Odd Parity** or **Even Parity**. The Parity type has to be the same across banks. The Parity configuration option allows No Parity, Even Parity and Odd Parity options. For a memory read, the parity bit is calculated on a read byte and then compared with the parity bit read from memory. When the parity error occurs, the AXI4 logic responds with an AXI SLVERR response. The error address is updated in the Parity Error Address register. For more than one SRAM bank, the number of Parity Error Address registers is defined by the **Number of Memory Banks** configuration option.

EMC Register

There are two register sets in the EMC Register Module: the Parity Error Address register (PERR_ADDR_REG_x) and the PSRAM/Micron® Technology Flash Configuration register (PSRAM_FLASH_CONFIG_REG_x). The PERR_ADDR_REG_x registers contain the address for the parity error that occurred. The PSRAM_FLASH_CONFIG_REG_x register is provided to configure the PSRAM or flash memory access mechanisms such as operation mode for the memory. For more details on Configuration and Status registers, see Register Description in Chapter 2.

I/O Registers

The I/O register module provides additional timing control for synchronous SRAM memories. All input/output signal on the memory interface is driven on the rising edge of the EMC clock.

Flash (Memory) Access through STARTUPE Primitive

STARTUP is a primitive in the Xilinx FPGA. This primitive can be used after the FPGA configuration in the design. For more understanding on the use of this primitive, read the targeted FPGA user guide, The STARTUPE3 primitive is present in UltraScale™ architecture and later devices. This primitive has a dedicated clock and data pins that can be used to provide the clock and partial data interface that is, Bits[3:0] of data bus to the slave memory.

For more information see the *UltraScale Architecture Libraries Guide* (UG974) [Ref 2].

Application

Figure 1-2 shows an example of the AXI EMC core use case.

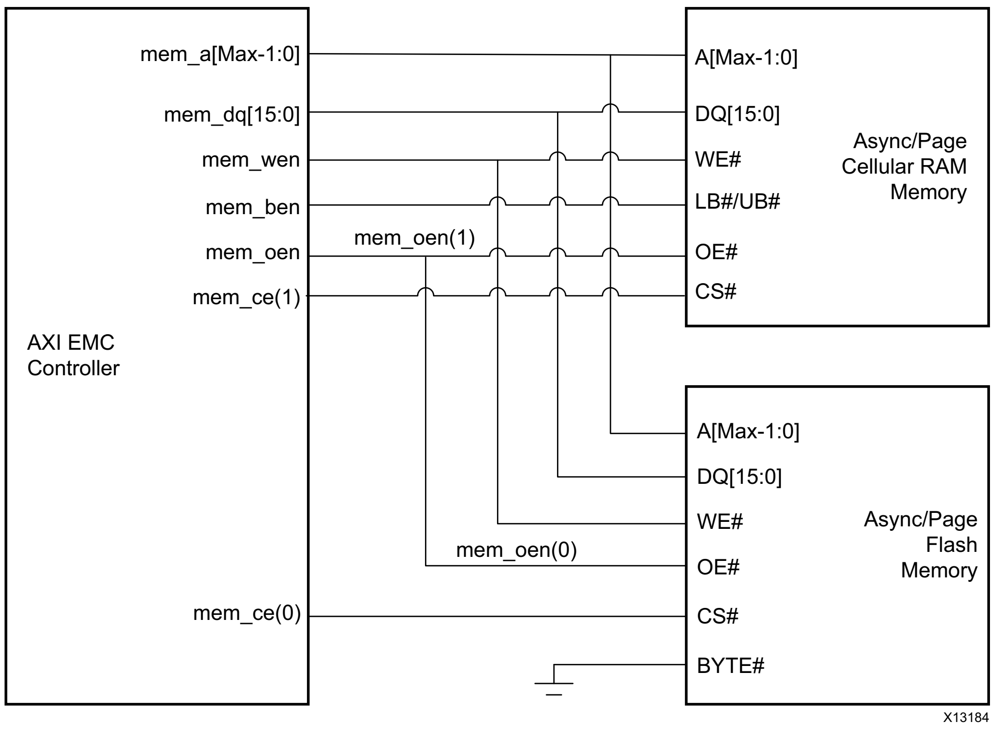


Figure 1-2: **AXI EMC Core Application Diagram**

In this use case, AXI EMC core is connected to external NOR Flash and Cellular RAM memory devices. Both of these memory devices have compatible pin interfaces which allow maximum pin sharing for reduced off device connectivity. The NOR Flash memory shown in the use case can be used as a primary nonvolatile storage device containing device boot code or application code.

The CellularRAM shown in the use case is a volatile Pseudo SRAM and can be used for cached or buffered memory applications. The AXI EMC core here allows the flash memory or Cellular RAM to be dynamically configured for asynchronously read operations or synchronously page read operations. The low latency and high performance AXI EMC core in this use case is ideally suited for eXecute In Place (XIP) system memory solutions.

Licensing and Ordering Information

This Xilinx® LogiCORE™ IP module is provided at no additional cost with the Xilinx Vivado® Design Suite under the terms of the Xilinx End User License.

Information about this and other Xilinx LogiCORE IP modules is available at the Xilinx Intellectual Property page. For information on pricing and availability of other Xilinx LogiCORE IP modules and tools, contact your local Xilinx sales representative.

Product Specification

This chapter contains details about the performance and ports of the core.

Standards Compliance

This core has bus interfaces that comply with the ARM® AMBA® AXI4 Protocol Specification Version 1.0 [Ref 1].

Performance

The performance and resource utilization numbers for the AXI EMC core are generated from a common benchmarking setup implementing a multi-masters/multi-slave AXI4 system.

The AXI EMC core is characterized as per the benchmarking methodology described in "Vivado IP Optimization (F_{Max} Characterization) appendix," in the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 3].

Maximum Frequencies

The maximum frequencies for AXI EMC are provided in Table 2-1.

Note: Performance numbers for UltraScale™ architecture and Zynq®-7000 devices are expected to be similar to 7 series device numbers.

Table 2-1: **Maximum Frequencies**

Family	Speed Grade	F_{Max} (MHz)	
		AXI4	AXI4-Lite
Virtex-7	–1	200	180
Kintex-7		200	180
Artix-7		150	120
Virtex-7	–2	240	200
Kintex-7		240	200
Artix-7		180	140

Table 2-1: **Maximum Frequencies *(Cont'd)***

Family	Speed Grade	F_{Max} (MHz)	
		AXI4	AXI4-Lite
Virtex-7	–3	280	220
Kintex-7		280	220
Artix-7		200	160

Resource Utilization

Note: UltraScale architecture results are expected to be similar to 7 series device results.

The AXI EMC resource utilization for various parameter value combinations measured with a Virtex® -7 device are detailed in Table 2-2.

Table 2-2: **Device Utilization – Virtex-7 FPGAs**

Parameter Values							Device Resources			
Pipe Delay	Enable Data Width Matching	Memory Data Width	AXI Data Width	Memory Type	Enable Internal Registers	No. of Memory Banks	LUTs	FFs	Slices	F_{Max} (MHz)
2	0	32	32	1	0	1	628	495	293	288
2	1	16	32	1	0	1	664	469	296	266
2	1	8	32	1	0	1	659	452	267	266
2	0	32	32	0	0	1	564	555	252	296
2	0	32	32	0	1	1	636	656	304	312
2	1	16	32	0	0	1	625	509	264	281
2	1	8	32	0	0	1	589	485	267	258

The AXI EMC resource utilization for various parameter value combinations measured with a Kintex®-7 and Zynq-7000 device are detailed in Table 2-3.

Table 2-3: **Device Utilization – Kintex-7 and Zynq-7000 (Kintex-7 Based) FPGAs**

Parameter Values								Device Resources			
Pipe Delay	**Parity**	**Enable Data Width Matching**	**Memory Data Width**	**AXI Data Width**	**Memory Type**	**Enable Internal Registers**	**No. of Memory Banks**	**LUTs**	**FFs**	**Slices**	**F_{Max} (MHz)**
2	0	1	8	32	1	0	1	669	452	271	281
2	1	0	32	32	0	0	1	559	555	277	320
2	0	1	16	32	1	0	1	676	469	334	274
2	1	1	8	32	0	0	1	612	482	266	274
2	0	0	32	32	1	0	1	634	495	361	304
2	1	0	32	32	0	1	1	631	656	336	312
2	1	1	16	32	0	0	1	616	506	273	274

The AXI EMC resource utilization for various parameter combinations measured with an Artix®-7 device are detailed in Table 2-4.

Table 2-4: **Device Utilization – Artix-7 and Zynq-7000 FPGAs**

Parameter Values								Device Resources			
Pipe Delay	**Parity**	**Enable Data Width Matching**	**Memory Data Width**	**AXI Data Width**	**Memory Type**	**Enable Internal Registers**	**No. of Memory Banks**	**LUTs**	**FFs**	**Slices**	**F_{Max} (MHz)**
2	1	0	32	32	0	0	1	559	555	261	204
2	0	1	16	32	1	0	1	681	469	290	180
2	0	0	32	32	1	0	1	633	495	301	188
2	0	1	8	32	1	0	1	676	452	269	172
2	1	0	32	32	0	1	1	630	656	299	204
2	1	1	16	32	0	0	1	615	506	274	172
2	1	1	8	32	0	0	1	608	482	267	180

Port Descriptions

The I/O signals are listed and described in Table 2-5.

Table 2-5: **I/O Signal Descriptions**

Signal Name	Interface	I/O	Initial State	Description
AXI Global System Signals				
s_axi_aclk[1]	AXI	I	–	AXI clock
s_axi_aresetn	AXI	I	–	AXI reset; active-Low
rdclk[2]	System	I	–	Read clock to capture the data from memory
AXI4-Lite Interface Signals[3]				
s_axi_reg*	AXI-Lite	I	–	See Appendix A of the *Vivado AXI Reference Guide* (UG1037) [Ref 4] for a description of AXI4 signals.
AXI4 Channel Signals				
s_axi_mem*[8]	AXI	I	–	See Appendix A of the *Vivado AXI Reference Guide* (UG1037) [Ref 4] for a description of AXI4 signals.
External Memory Interface Signals				
mem_dq_i [Max Data Width of Memories – 1:0]	External memory	I	–	Memory input data bus. Max Memory Data Width is the maximum memory data width configured across all memory banks and can be set as 8, 16, 32, or 64 bits wide.
mem_dq_o [Max Data Width of Memories – 1:0]	External memory	O	0	Memory output data bus. Max Memory Data Width is the maximum memory data width configured across all memory banks and can be set as 8, 16, 32, or 64 bits wide.
mem_dq_t [Max Data Width of Memories – 1:0]	External memory	O	0	Memory output 3-state signal. Max Memory Data Width is the maximum memory data width configured across all memory banks and can be set as 8, 16, 32, or 64 bits wide.
mem_dq_parity_i [Max Data Width of Memories/8 – 1:0][4]	External memory	I	–	Memory parity input data bits. Max Memory Data Width is the maximum memory data width configured across all memory banks and can be set as 8, 16, 32, or 64 bits wide.
mem_dq_parity_o [Max Data Width of Memories/8 – 1:0][4]	External memory	O	0	Memory parity output data bits. Max Memory Data Width is the maximum memory data width configured across all memory banks and can be set as 8, 16, 32, or 64 bits wide.
mem_dq_parity_t [Max Data Width of Memories/8 – 1:0][4]	External memory	O	0	Memory parity 3-state signals. Max Memory Data Width is the maximum memory data width configured across all memory banks and can be set as 8, 16, 32, or 64 bits wide.

Table 2-5: **I/O Signal Descriptions *(Cont'd)***

Signal Name	Interface	I/O	Initial State	Description
mem_a[31:0]	External memory	O	0	Memory address bus.
mem_rpn	External memory	O	1	Memory reset/power down.
mem_cen [No. of Memory Banks – 1:0]	External memory	O	1	Memory chip enables[5]; active-Low. The valid range for Memory Banks is from 1 to 4.
mem_oen [No. of Memory Banks – 1:0]	External memory	O	1	Memory output enable. The valid range for Memory Banks is from 1 to 4.
mem_wen	External memory	O	1	Memory write enable.
mem_qwen [(Max Data Width of Memories/8) – 1:0]	External memory	O	1	Memory qualified write enables. Max Memory Data Width is the maximum memory data width configured across all memory banks and can be set as 8, 16, 32, or 64 bits wide.
mem_ben [(Max Data Width of Memories/8) – 1:0]	External memory	O	0	Memory byte enables. Max Memory Data Width is the maximum memory data width configured across all memory banks and can be set as 8, 16, 32, or 64 bits wide.
mem_ce [No. of Memory Banks – 1:0]	External memory	O	0	Memory chip enables[5]; active-High. The valid range for Memory Banks is from 1 to 4.
mem_adv_ldn	External memory	O	1	Memory advance burst address/load new address.
mem_lbon	External memory	O	1	Memory linear/interleaved burst order.
mem_cken	External memory	O	0	Memory clock enable.
mem_rnw	External memory	O	1	Memory read not write.
mem_cre	External memory	O	0	Command sequence configuration of PSRAM.
STARTUP_IO	STARTUP_IO			Has STARTUP block ports unused by AXI EMC IP. ***Note:*** See the STARTUP document for complete details of Ports and their functionality.

Table 2-5: **I/O Signal Descriptions *(Cont'd)***

Signal Name	Interface	I/O	Initial State	Description
mem_wait	External memory	I	–	Input signal from Micron Flash device; This signal is used only when Memory Type is set to Micron Flash.

Notes:

1. The same clock and reset signals should be used for both the register and memory interfaces.
2. This clock is used to capture the data from memory. Xilinx recommends that this clock is tied with AXI Clock source.
3. The AXI4-Lite interface is optional and is provided only when the Enable Internal Registers option is set.
4. This pin is only enabled when parity type selected is either odd parity or even parity in the Vivado IDE.
5. Most asynchronous memory devices only use mem_cen. Most synchronous memory devices use both mem_cen and mem_ce.
6. All mem_* ports have IOB attributes set to TRUE when STARTUP is not included in the design.
7. All mem_* ports except mem_dq_*[3:0] and mem_cen[0] which are connected to STARTUP have IOB attributes set TRUE when STARTUP is included in the design.
8. Write commands to Flash should be issued with awlen=0.

Register Description

There are four internal registers in the AXI EMC design (Table 2-6). The memory map of the AXI EMC registers is determined by setting the Base Address in the Address Editor tab of the Vivado IP integrator. The internal registers of the AXI EMC are at a fixed offset from the base address and are byte accessible. The AXI EMC internal registers and their offset are listed in Table 2-6.

Table 2-6: **Internal Registers and Offsets**

AXI Lite Base Address + Offset (hex)	Register Name	Access Type	Default Value (hex)	Description
Base Address + 0x00	PARITY_ERR_ADDR_REG_0	Read	0x0	Bank-0 parity error address register
Base Address + 0x04	PARITY_ERR_ADDR_REG_1	Read	0x0	Bank-1 parity error address register
Base Address + 0x08	PARITY_ERR_ADDR_REG_2	Read	0x0	Bank-2 parity error address register
Base Address + 0x0C	PARITY_ERR_ADDR_REG_3	Read	0x0	Bank-3 parity error address register
Base Address + 0x10	PSRAM_FLASH_CONFIG_REG_0	Write/ Read	0x24	Bank-0 PSRAM/Flash configuration register
Base Address + 0x14	PSRAM_FLASH_CONFIG_REG_1	Write/ Read	0x24	Bank-1 PSRAM/Flash configuration register
Base Address + 0x18	PSRAM_FLASH_CONFIG_REG_2	Write/ Read	0x24	Bank-2PSRAM/Flash configuration register

Table 2-6: **Internal Registers and Offsets *(Cont'd)***

AXI Lite Base Address + Offset (hex)	Register Name	Access Type	Default Value (hex)	Description
Base Address + 0x1C	PSRAM_FLASH_CONFIG_REG_3	Write/ Read	0x24	Bank-3PSRAM/Flash configuration register

Notes:

1. Based upon the no. of banks chosen, only those no. registers are available and accessible for read and write-if allowed.
2. Other registers are considered as nonexistent and AXI transaction returns gracefully if accessed.
3. A write transaction will be executed only when AWVALID and WVALID are asserted. The AWREADY is gated by assertion of WVALID and vice-versa.

PARITY ERROR ADDRESS Register (PARITY_ERR_ADDR_REG_x)

The AXI EMC Parity Error Address registers are read-only registers that provide the AXI address on which the parity error occurred. These registers are valid only when AXI4-Lite interface is enabled and Memory Type selected for the bank is Sync/Async SRAM. The number of such registers depends on the **Number of Memory Banks** set for the core.

There can be four PARITY_ERR_ADDR_REG registers. Whenever a parity error occurs for a AXI read operation, the AXI EMC core updates this register with the address at which the error occurred. The Parity Error Address register is shown in Figure 2-1 and described in Table 2-7.

31
0
DS762_04

Figure 2-1: **AXI EMC Parity Error Address Register**

Table 2-7: **Parity Error Address Register Description**

Bits	Name	Core Access	Reset Value	Description
31:0	PARITY_ERR_ADDR_REG_x	Read	0x0	Parity error register width

PSRAM/Flash Configuration Register (PSRAM_FLASH_CONFIG_REG_X)

The PSRAM/Flash configuration registers are write and read registers that are used for configuration of the controller for the PSRAM or flash memories. These registers are used to configure:

- PSRAM memory when AXI4-Lite interface is enabled and Memory Type selected for the bank is PSRAM.
- Flash memories in burst mode when AXI4-Lite interface is enabled, Memory Type selected for the bank is Micron® Flash, and Burst mode for Micron Flash memory is enabled.

The number of such registers depends on the **Number of Memory Banks** set for the core and the **Memory Type** selected in the associated Memory Bank.

IMPORTANT: *For Memory Types selected across memory banks, see Allowable Memory Combination in Chapter 3.*

There can be four PSRAM_FLASH_CONFIG_REG registers. The PSRAM/Micron Flash Burst mode configuration register is shown in Figure 2-2 and described in Table 2-8.

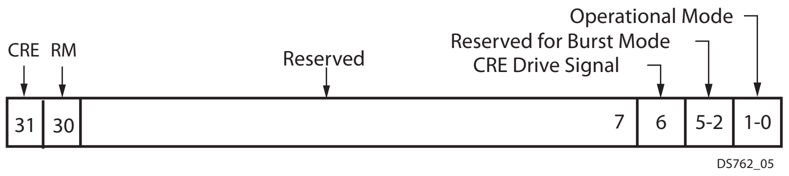


Figure 2-2: **PSRAM/Micron Flash Configuration Register**

Table 2-8: **PSRAM/Micron Flash Configuration Register (PSRAM_FLASH_CONFIG_REG_X)**

Bits	Name	Core Access	Reset Value	Description
31	Control register enable (CRE)	Write/ Read	0x0	• 0x0 = Normal read/writes to Micron Flash • 0x1 = Enables Micron flash RCR register configuration ***Note:*** Valid only if connected memory is Micron Flash.
30	Read Mode (RM)	Write/ Read	0x0	• 0x0 = Asynchronous Page Mode • 0x1 = Synchronous Burst Mode (Applicable when Micron Flash is switched to Sync Read Burst Mode)
29:7	Reserved	N/A	0	Reserved. If read, might return undefined value. If written to these bits, is not updated.
6	CRE drive	Write/ Read	0	• 0x0 = Remove Drive Command for CRE • 0x1 = Drive Command for CRE ***Note:*** Valid only for PSRAM memories.
5:2	Reserved	Write/ Read	0x9	Reserved. If read, might return undefined value. If written to these bits, is not updated.
1:0	Operational Mode	Write/ Read	0x0	This bits describe the mode in which PSRAM is configured (Applicable only when target memory is PSRAM) • 0x00 = Asynchronous Mode • 0x01 = Page Mode • 0x10 = Reserved for future use (Burst Mode)

Notes:

1. For Latency configurations and access details, see the respective PSRAM data sheet.

Configuring the Micron Flash in Sync Burst Read Mode

Table 2-9 provides the required bit settings for Sync Burst Micron Flash Memories Read Configuration register. When Micron Flash is configured with sync burst read operations, the register configuration setting must be as provided in this table.

The Read Configuration register is in the flash memory. The address for accessing this register is set as Memory Bank Base address+ 32’h0000_509F.

Table 2-9: **Micron Flash Sync Mode – Read Configuration Register (RCR)**

Read Configuration Register (RCR)															
Read Mode	**Latency Count**				**WAIT Polarity**	**RES**	**WAIT Delay**	**Burst Seq**	**CLK Edge**	**RES**	**RES**	**Burst Wrap**	**Burst Length**		
RM	LC[3:0]				WP	R	WD	BS	CE	R	R	BW	BL[2:0]		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	0	1	0	0	0	0	1	0	0	1	1	1	1

Note: The RCR settings are specific for the P30 flash family and slower speeds. The Micron P30/P33 and G18F/MT28GU01GAAA1E families should be supported and RCR recommendations referenced.

The guideline for configuring the Sync Read Mode for Micron Flash are as follows. This process is applicable to the memory bank with Memory Type configured for Micron Flash.

- By default the Micron Flash operates in Asynchronous mode. These settings are required while switching the read operation of memory from Asynchronous Read to Synchronous Read.
- Set Bits[31:30] of PSRAM_FLASH_CONFIG_REG_X register for selected memory bank to 1.
- Drive the value of Memory Bank Base address+ 0x0000_589E on AXI write address bus with 0x0300_0060 on AXI write data bus. This should be of single beat transaction.
- Drive the value of Memory Bank Base address+ 0x0000_589E on AXI write address bus with 0x6000_0003 on AXI write data bus. This should be of single beat transaction.
- The previous two steps write 589E into the Read Configuration register of the Micron Flash.
- Reset Bit[31] of the PSRAM_FLASH_CONFIG_REG_X register for selected memory bank to continue with normal memory read/writes.
- The AXI EMC core behavior is designed on the basis of the above register bit settings, for any other bit settings the core behavior is not guaranteed.
- While switching from Sync to Async mode, the Sync mode settings of memory should be disabled by resetting the Read Configuration register (Bit[15]) to 1. After this bit is reset, set the AXI EMC cores internal registers Bit[30] to 0.

- After both the core and memory are reset with the bit settings mentioned above, the memory and the core operates in Asynchronous Read mode.

IMPORTANT: *When CRE bit in PSRAM_FLASH_CONFIG_REG_X (Bit[31]) is set to "1," then the core would ignore value driven in wstrb bus and would always transfer the LSB bytes of wdata bus to the Memory Interface.*
If the pin does not have a pull-down register connected, the WAIT pin is a 3-state output from the flash. It is recommended to change the polarity of WAIT to active-High in Read Configuration register of the Micron Flash and connect to EMC through an inverter.

Designing with the Core

This chapter includes guidelines and additional information to facilitate designing with the AXI External Memory Controller (EMC) core.

General Design Guidelines

Address Map Description

The AXI EMC core supports up to four banks of external memory. The number of banks used is determined by the **Number of Memory Banks** configuration option and can take values between 1 and 4. The banks that are used are banks 0 to bank 3. Each bank of memory has its own independent base address and High address range. The address range of a bank of memory is restricted to have a size (in bytes) that is a power of 2 and is address-aligned. For an address-range of size is 2^n, the *n* least significant bits of the base address is 0 and *n* least significant bits of the High address are 1. For example, a memory bank with an addressable range of 16 MB (2^{24}) could have a base address of `0xFF000000` and a High address of `0xFFFFFFFF`. A memory bank with an addressable range of 64 kB (2^{16}) could have a base address of `0xABCD0000` and a High address of `0xABCDFFFF`. The AXI EMC core transactions must fall between the bank x base address and High address. The addresses for the memory banks can be configured through per bank **Base Address** and **High Address** configuration options provided in the Customize IP dialog box of the Vivado® IP catalog or through the Address Editor tab of IP integrator.

AXI EMC core also supports internal registers for Memory configurations. These registers are enabled by setting the **Enable Internal** register option. The memory map for these internal registers get determined by the **Base Address** settings for the AXI EMC core in the Address Editor tab of Vivado IP integrator. For more detail on AXI EMC core internal register configurations, see Register Description in Chapter 2.

Allowable Memory Combination

Table 3-1 lists the allowed memory combinations across multiple banks of the AXI EMC core. Memories can be arranged in any order with a maximum up to four.

Each row is a group of acceptable memory combinations. Memories between different rows cannot be mixed within the same Memory Controller.

Table 3-1: **Allowable Memory Combination**

PSRAM (Async Mode)	Micron Flash (Async Mode)	Linear Flash	Async SRAM
PSRAM (Page Mode)	Page Mode Flash	–	–
Sync SRAM	Async SRAM	–	–

Note: By default PSRAM/Micron® operate in asynchronous mode. Select the **Enable Internal Registers** option while customizing the AXI EMC core if PSRAM/Micron needs to operate in other modes.

Memory Data Types and Organization

Memory can be accessed through the AXI EMC core as one of four types:

- Byte (8-bit)
- Halfword (16-bit)
- Word (32-bit)
- Doubleword (64-bit)

Data to and from the AXI interface is organized as little-endian. The bit and byte labeling for the big-endian data types is shown in Figure 3-1.

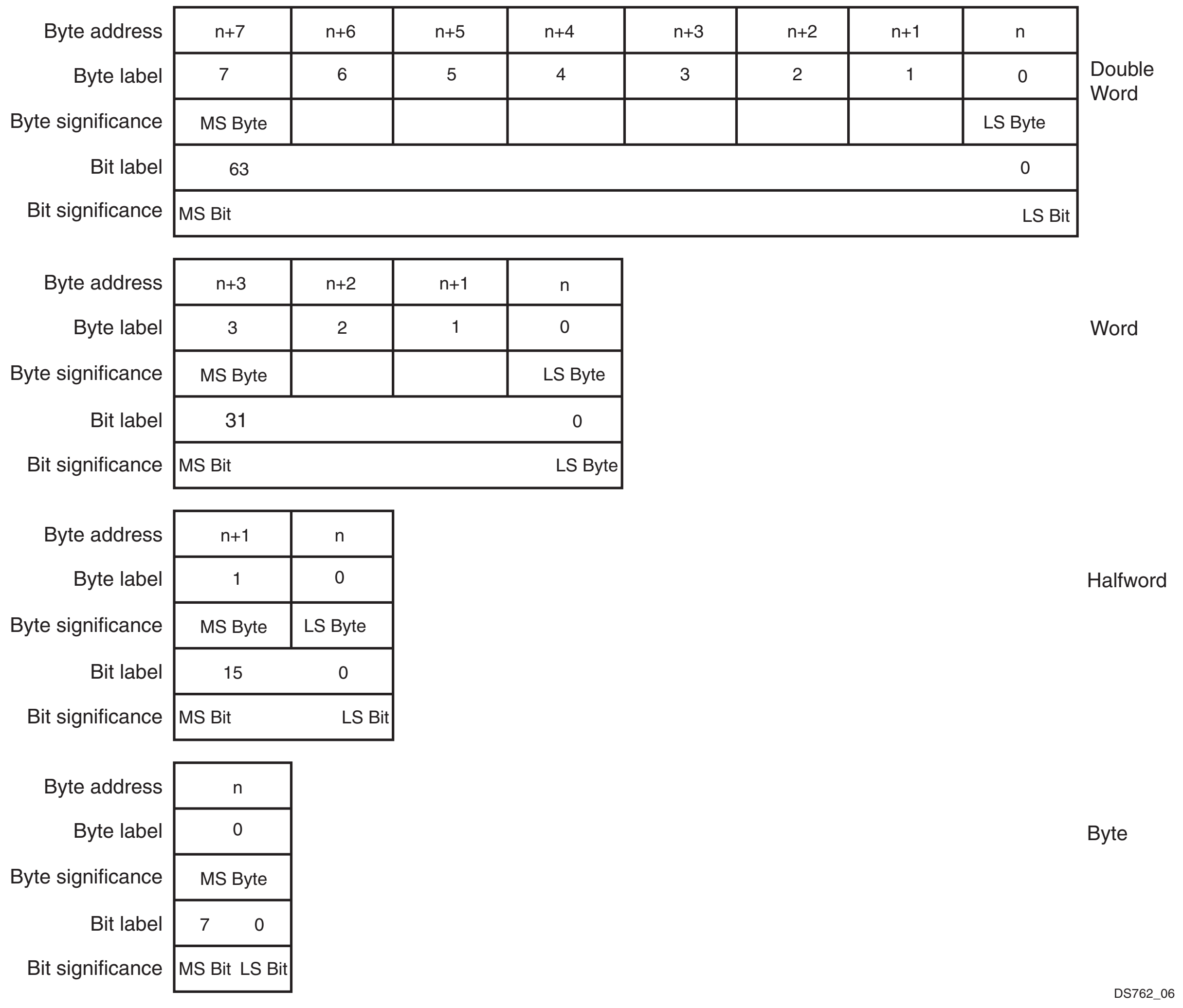


Figure 3-1: **Memory Data Types**

Connecting to Memory

Clocking Synchronous Memory

The AXI EMC core does not provide a clock output to any synchronous memory. The AXI clock should be routed through an ODDR module to provide the clock to synchronous memory. To synchronize the synchronous memory clock to the internal FPGA clock, the FPGA system design should include a Mixed Mode Clock Manager (MMCM), external to the AXI EMC core, that uses the synchronous memory clock input as the feedback clock as shown in Figure 3-2. The synchronous clock output from the FPGA must be routed back to the FPGA on a clock pin.

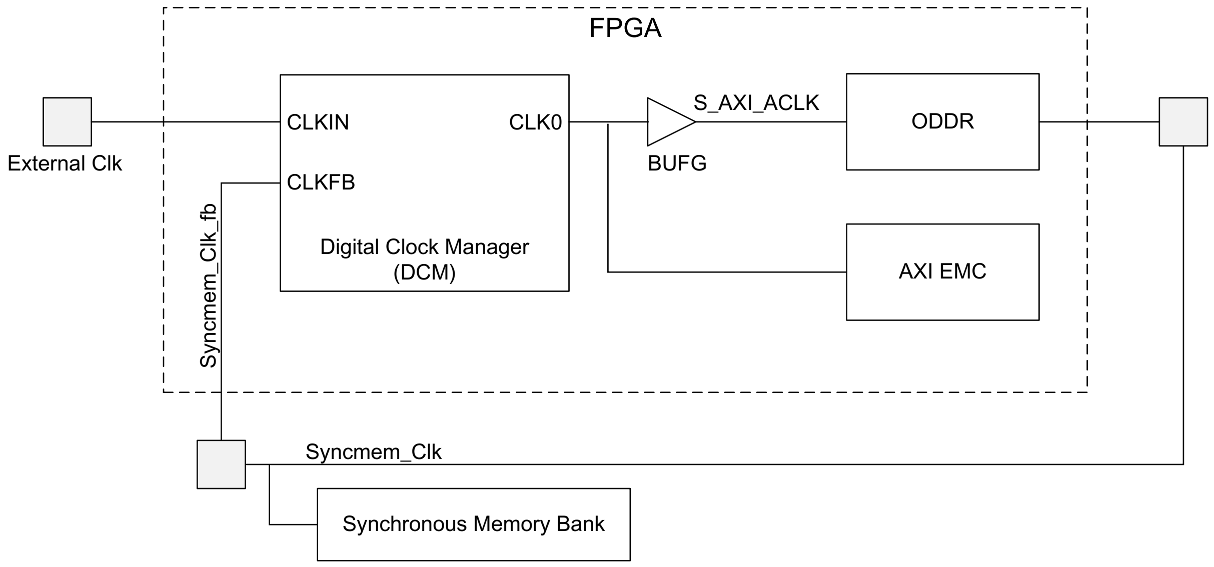


Figure 3-2: **Synchronous Memory Bank Clocked by FPGA Output with Feedback**

The following constraint has to be added in XDC when clock to Synchronous RAM is forwarded through ODDR:

```
create_generated_clock -name clk_forward -source [get_pins -hier *ODDR/C] -invert
-divide_by 1 [get_ports clk_fwd]
```

Because forwarded clock is 180° out-of-phase, the data is to be latched on second rising edge, the following constraints need to be set:

```
set_multicycle_path 2 -setup -from clk_forward -to [get_clocks -of_objects [get_pins
-hierarchical *Mem_DQ_I_int_reg[*]/C]]
set_multicycle_path 1 -hold -end -from clk_forward -to [get_clocks -of_objects
[get_pins -hierarchical *Mem_DQ_I_int_reg[*]/C]]
```

If Parity is enabled, the following constraints should also be added to XDC:

```
set_multicycle_path 2 -setup -from clk_forward -to [get_clocks -of_objects [get_pins
-hierarchical *Mem_DQ_PARITY_I_int_reg[*]/C]]
set_multicycle_path 1 -hold -end -from clk_forward -to [get_clocks -of_objects
[get_pins -hierarchical *Mem_DQ_PARITY_I_int_reg[*]/C]]
```

If the synchronous memory is clocked by the same external clock as the FPGA, or if the clock feedback is not available, the DCM shown in Figure 3-3 should be included in the FPGA external to the AXI EMC core.

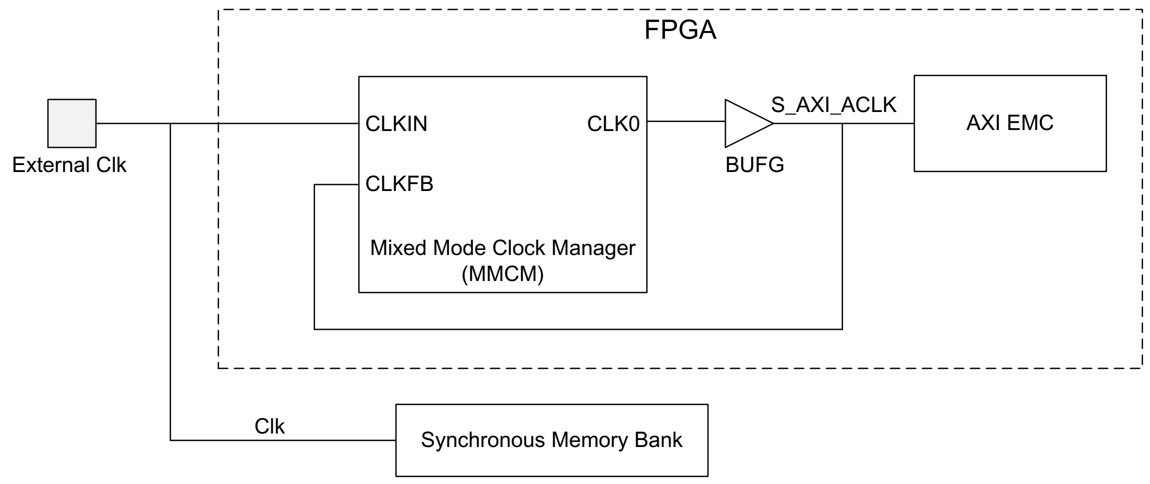


Figure 3-3: **Synchronous Memory Bank Clocked by External Clock**

1. When using Micron® Flash in **synchronous** mode, ensure that memory clock interface is connected to 50 MHz and AXI Clock interface is connected to 100 MHz. Both of these clocks must originate from same source.

IMPORTANT: *By default, Micron Flash memories operate in asynchronous read mode. To operate Micron Flash in synchronous mode:*
*(a) Select **Enable Internal Registers** option while customizing the AXI EMC core.*
(b) See the Core register Configuration and Memory Configuration section.

2. In all IP operating modes, Xilinx recommends that the same clock is connected to both rdclk and `s_axi_aclk` ports.

Address Bus, Data Bus, and Control Signal Connections

The three primary considerations for connecting the controller to memory devices are the width of the AXI data bus, the width of the memory subsystem and the number of memory devices used. The data and address signals at the memory controller are labeled with little-endian bit labeling (for example, D(31:0) where D(31) is the MSB), and memory devices are also little-endian.

Note: Most asynchronous memory devices only use MEM_CEN, while most synchronous memory devices use both MEM_CEN and MEM_CE. MEM_CEN is a function of the address decode while MEM_CE is a function of the state machine logic.

For Address and Data connections to the external memory devices, see Tables 3-3, 3-5, 3-7, 3-9, 3-11, 3-13 and 3-15. In addition, Table 3-2 shows variables used in defining memory subsystem.

Table 3-2: **Variables Used in Defining Memory Subsystem**

Variable	Allowed Range	Definition
BN	3 down to 0	Memory bank number
DN	3 down to 0	Memory device number within a bank. The memory device attached to the most significant bit in the memory subsystem is 0 to 3.
MW	8 to 64	Width in bits of memory subsystem (8, 16, 32, and 64 are allowed values)
DW	63 down to 1	Width in bits of data bus for memory device
MAW	32 down to 1	Width in bits of address bus for memory device
AU	63 down to 1	Width in bits of smallest addressable data word on the memory device
AS	X(1)	Address shift for address bus = $\log_2((MW \times AU/DW)/8)$
HAW	32 down to 1	Width in bits of AXI address bus

Notes:

1. The value of X depends on variables MW, AU and DW.

Connecting to SRAM

Table 3-3: **Core To Memory Interconnect**

Description	AXI EMC Signals (MSB:LSB)	Memory Device Signals (MSB:LSB)
Data bus	MEM_DQ(((DN + 1) × DW) – 1:DN × DW)	D(DW – 1:0)
Address bus	MEM_A(MAW – AS – 1:AS)	A(MAW – 1:0)
Chip enable (active-Low)	MEM_CEN(BN)	CEN
Output enable (active-Low)	MEM_OEN	OEN
Write enable (active-Low)	MEM_WEN	WEN (for devices that have byte enables)
Qualified write enable (active-Low)	MEM_QWEN(DN × DW/8)	WEN (for devices that do not have byte enables)
Byte enable (active-Low)	MEM_BEN((((DN + 1) × DW/8) – 1):(DN × DW/8))	BEN(DW/8 – 1:0)

Example 1: Connection to 32-bit Memory Using Two IDT71V416S SRAM Parts

Table 3-4 shows variables for a simple SRAM example.

Table 3-4: **Variables For Simple SRAM Example**

Variable	Value	Definition
BN	0	Memory bank number
DN	1 down to 0	Memory device number within a bank. The memory device attached to the most significant bit in the memory subsystem is 0; device numbers increase toward the least significant bit.
MW	32	Width in bits of memory subsystem
DW	16	Width in bits of data bus for memory device
MAW	18	Width in bits of address bus for memory device
AU	16	Width in bits of smallest addressable data word on the memory device
AS	2	Address shift for address bus = $\log_2((MW \times AU/DW)/8)$
HAW	32	Width in bits of host address bus (for example, AXI)

Table 3-5 shows connection to 32-bit memory using two IDT71V416S (256K × 16-bit) parts.

Table 3-5: **Connection to 32-bit Memory Using Two IDT71V416S Parts**

DN	Description	AXI EMC Signals (MSB:LSB)	Memory Device Signals (MSB:LSB)
0	Data bus	MEM_DQ(15:0)	I/O(15:0)
	Address bus	MEM_A(19:2)	A(17:0)
	Chip enable (active-Low)	MEM_CEN(0)	CS
	Output enable (active-Low)	MEM_OEN	OE
	Write enable (active-Low)	MEM_WEN	WE
	Byte enable (active-Low)	MEM_BEN(1:0)	$\overline{\text{BHE}}$:$\overline{\text{BLE}}$
1	Data bus	MEM_DQ(31:16)	I/O(15:0)
	Address bus	MEM_A(19:2)	A(17:0)
	Chip enable (active-Low)	MEM_CEN(0)	CS
	Output enable (active-Low)	MEM_OEN	OE
	Write enable (active-Low)	MEM_WEN	WE
	Byte enable (active-Low)	MEM_BEN(3:2)	$\overline{\text{BHE}}$:$\overline{\text{BLE}}$

Connecting to Byte Parity Memory

Connection to 32-bit Memory Using IS61LVPS25636A Synchronous SRAM Parts

Table 3-6 shows the variables for a simple SRAM example.

Table 3-6: **Variables for Simple SRAM Example**

Variable	Value	Definition
BN	0	Memory bank number
DN	0	Memory device number within a bank. The memory device attached to the most significant bit in the memory subsystem is 0; device numbers increase toward the least significant bit.
MW	32	Width in bits of memory subsystem
DW	32	Width in bits of data bus for memory device
MAW	18	Width in bits of address bus for memory device
AU	32	Width in bits of smallest addressable data word on the memory device
AS	2	Address shift for address bus = $\log_2((MW \times AU/DW)/8)$
HAW	32	Width in bits of host address bus (for example, AXI)

Table 3-7 shows the connection to 32-bit memory using two IDT71V416S (256K × 16-bit) parts.

Table 3-7: **Connection to 32-bit Memory Using Two IDT71V416S Parts**

DN	Description	AXI EMC Signals (MSB:LSB)	Memory Device Signals (MSB:LSB)
0	Data bus	MEM_DQ(31:0)	I/O(15:0)
	Address bus	MEM_A(19:2)	A(0:17)
	Chip enable (active-Low)	MEM_CEN(0)	CS
	Output enable (active-Low)	MEM_OEN	OE
	Write enable (active-Low)	MEM_WEN	WE
	Byte enable (active-Low)	MEM_BEN(3:0)	BWAb, BWBb, BWCb, BWDb
	Parity Bits	MEM_DQ_PARITY(3:0)	I/O(35:32)

Connecting to Intel StrataFlash

StrataFlash parts contain an identifier register, a status register, and a command interface. Therefore, the bit label ordering for these parts is critical to enable correct operation. Table 3-8 shows an example of how to connect the big-endian AXI EMC bus to the little-endian StrataFlash parts. The correct connection ordering is also indicated in a more general form in Table 3-3. StrataFlash parts have an x8 mode and an x16 mode, selectable with the BYTE# input pin. To calculate the proper address shift, the minimum addressable word is 8 bits for both x8 and x16 modes, because A0 always selects a byte.

Example 2: Connection to 32-bit Memory Using Two StrataFlash Parts in x16 Mode

This configuration supports byte read, but not byte write. The smallest data type that can be written is 16-bit data. Table 3-8 shows the variables for StrataFlash (x16 mode) example.

Table 3-8: **Variables For StrataFlash (x16 mode) Example**

Variable	Value	Definition
BN	0	Memory bank number
DN	0 to 1	Memory device number within a bank. The memory device attached to the most significant bit in the memory subsystem is 0; the device numbers increase toward the least significant bit.
MW	32	Width in bits of memory subsystem
DW	16	Width in bits of data bus for memory device
MAW	24	Width in bits of address bus for memory device
AU	8	Width in bits of smallest addressable data word on the memory device
AS	1	Address shift for address bus = $\log_2((MW \times AU/DW)/8)$
HAW	32	Width in bits of host address bus (for example, AXI)

Table 3-9 shows the connection to 32-bit memory using two StrataFlash parts.

Table 3-9: **Connection to 32-bit Memory Using Two StrataFlash Parts**

DN	Description	AXI EMC Signals (MSB:LSB)	StrataFlash Signals (MSB:LSB)
0	Data bus	MEM_DQ(15:0)	DQ(15:0)
	Address bus	MEM_A(24:1)	A(23:0)
	Chip enable (active-Low)	GND, GND, MEM_CEN(0)	CE(2:0)
	Output enable (active-Low)	MEM_OEN	OE#
	Write enable (active-Low)	MEM_QWEN(0)	WE#
	Reset/Power down (active-Low)	MEM_RPN	RP#
	Byte mode select (active-Low)	N/A – tie to GND	BYTE#
	Program enable (active-High)	N/A – tie to VCC	V_{PEN}
1	Data bus	MEM_DQ(31:16)	DQ(15:0)
	Address bus	MEM_A(24:1)	A(23:0)
	Chip enable (active-Low)	GND, GND, MEM_CEN(0)	CE(2:0)
	Output enable (active-Low)	MEM_OEN	OE#
	Write enable (active-Low)	MEM_QWEN(2)	WE#
	Reset/Power down (active-Low)	MEM_RPN	RP#
	Byte mode select (active-Low)	N/A – tie to GND	BYTE#
	Program enable (active-High)	N/A – tie to VCC	V_{PEN}

Example 3: Connection to 32-bit Memory Using Four StrataFlash Parts in x8 Mode

This configuration supports byte reads and writes. Table 3-10 shows the variables for the StrataFlash (x8 mode) example.

Table 3-10: **Variables for StrataFlash (x8 mode) Example**

Variable	Value	Definition
BN	0	Memory bank number
DN	0 to 3	Memory device number within a bank. The memory device attached to the most significant bit in the memory subsystem is 0; the device numbers increase toward the least significant bit.
MW	32	Width in bits of memory subsystem
DW	8	Width in bits of data bus for memory device
MAW	24	Width in bits of address bus for memory device
AU	8	Width in bits of smallest addressable data word on the memory device
AS	2	Address shift for address bus = $\log_2((MW \times AU/DW)/8)$
HAW	32	Width in bits of host address bus (for example, AXI)

Table 3-11 shows the connection to 32-bit memory using four StrataFlash parts.

Table 3-11: **Connection to 32-bit Memory Using Four StrataFlash Parts**

DN	Description	AXI EMC Signals (MSB:LSB)	StrataFlash Signals (MSB:LSB)
0	Data bus	MEM_DQ(7:0)	DQ(7:0)[1]
	Address bus	MEM_A(23:2)	A(21:0)
	Chip enable (active-Low)	GND, GND, MEM_CEN(0)	CE(2:0)
	Output enable (active-Low)	MEM_OEN	OE#
	Write enable (active-Low)	MEM_QWEN(0)	WE#
	Reset/Power down (active-Low)	MEM_RPN	RP#
	Byte mode select (active-Low)	N/A – tie to GND	BYTE#
	Program enable (active-High)	N/A – tie to VCC	V_{PEN}
1	Data bus	MEM_DQ(15:8)	DQ(7:0)()
	Address bus	MEM_A(23:2)	A(21:0)
	Chip enable (active-Low)	GND,GND, MEM_CEN(0)	CE(2:0)
	Output enable (active-Low)	MEM_OEN	OE#
	Write enable (active-Low)	MEM_QWEN(1)	WE#
	Reset/Power down (active-Low)	MEM_RPN	RP#
	Byte mode select (active-Low)	N/A – tie to GND	BYTE#
	Program enable (active-High)	N/A – tie to VCC	V_{PEN}

Table 3-11: **Connection to 32-bit Memory Using Four StrataFlash Parts *(Cont'd)***

DN	Description	AXI EMC Signals (MSB:LSB)	StrataFlash Signals (MSB:LSB)
2	Data bus	MEM_DQ(23:16)	DQ(7:0)(1)
	Address bus	MEM_A(23:2)	A(21:0)
	Chip enable (active-Low)	GND, GND, MEM_CEN(0)	CE(2:0)
	Output enable (active-Low)	MEM_OEN	OE#
	Write enable (active-Low)	MEM_QWEN(2)	WE#
	Reset/Power down (active-Low)	MEM_RPN	RP#
	Byte mode select (active-Low)	N/A – tie to GND	BYTE#
	Program enable (active-High)	N/A – tie to VCC	V_{PEN}
3	Data bus	MEM_DQ(31:24)	DQ(7:0)(1)
	Address bus	MEM_A(23:2)	A(21:0)
	Chip enable (active-Low)	GND, GND, MEM_CEN(0)	CE(2:0)
	Output enable (active-Low)	MEM_OEN	OE#
	Write enable (active-Low)	MEM_QWEN(3)	WE#
	Reset/Power down (active-Low)	MEM_RPN	RP#
	Byte mode select (active-Low)	N/A – tie to GND	BYTE#
	Program enable (active-High)	N/A – tie to VCC	V_{PEN}

Notes:

1. In an x8 configuration, DQ(15:8) are not used and should be treated according to the data sheet of the manufacturer.

Connecting to Spansion Page Mode Flash S29GL032N

Table 3-12 shows the variables for the Spansion Page Mode Flash (x16 mode) example.

Table 3-12: **Variables for Page Mode Flash (x16 mode) Example**

Variable	Value	Definition
BN	0	Memory bank number
DN	0	Memory device number within a bank. The memory device attached to the most significant bit in the memory subsystem is 0.
MW	32	Width in bits of memory subsystem
DW	16	Width in bits of data bus for memory device
MAW	21	Width in bits of address bus for memory device
AU	16	Width in bits of smallest addressable data word on the memory device
AS	1	Address shift for address bus = $\log_2((MW \times AU/DW)/8)$
HAW	32	Width in bits of host address bus (for example, AXI)

Table 3-13: **Connection to 16-bit Memory Using Page Mode Flash Parts**

DN	Description	AXI EMC Signals (MSB:LSB)	PagemModeFlash Signals (MSB:LSB)
0	Data bus	MEM_DQ(15:0)	DQ(15:0)
	Address bus	MEM_A(22:1)	A(21:0)
	Chip enable (active-Low)	MEM_CEN(0)	CE
	Output enable (active-Low)	MEM_OEN	OE#
	Write enable (active-Low)	MEM_QWEN(0)	WE#
	Reset/Power down (active-Low)	MEM_RPN	RP#
	Byte mode select (active-Low)	N/A - tie to VCC	BYTE#
	Program enable (active-High)	N/A - tie to VCC	WP#

Connecting to PSRAM

Table 3-14: **Variables PSRAM (x16 mode) Example**

Variable	Value	Definition
BN	0	Memory bank number
DN	0	Memory device number within a bank. The memory device attached to the most significant bit in the memory subsystem is 0.
MW	32	Width in bits of memory subsystem
DW	16	Width in bits of data bus for memory device
MAW	23	Width in bits of address bus for memory device
AU	16	Width in bits of smallest addressable data word on the memory device
AS	1	Address shift for address bus = $\log_2((MW \times AU/DW)/8)$
HAW	32	Width in bits of host address bus (for example, AXI)

Table 3-15: **Connection to 16-bit Memory Using PSRAM Parts**

DN	Description	AXI EMC Signals (MSB:LSB)	PageModeFlash Signals (MSB:LSB)
0	Data bus	MEM_DQ(15:0)	DQ(15:0)
	Address bus	MEM_A(22:1)	A(21:0)
	Chip enable (active-Low)	MEM_CEN(0)	CE
	Output enable (active-Low)	MEM_OEN	OE#
	Write enable (active-Low)	MEM_QWEN(0)	WE#
	Reset/Power down (active-Low)	MEM_RPN	RP#
	Byte Enable (active-Low)	Mem_BEN(1:0)	UB#, LB#
	Control Register Enable (active-High)	Mem_CRE	CRE

Software Considerations

Absolute memory addresses are frequently required for specifying command sequences to flash memory devices.

IMPORTANT: *Due to the memory address shift (AS), any absolute addresses that must appear on the memory device address bus must also have the source AXI address left-shifted to compensate for the AS.*

All of the EMC Memory interface pins except for `mem_cre`, `mem_wait`, and `mem_a` (for Micron Flash in Sync mode) are registered I/Os. Xilinx recommends having these flops placed in the FPGA I/O pins.

All of the Memory interface pins which are applicable to connected memory (from Table 3-2 to Table 3-15) except for Address, CRE, and WAIT should be constrained to have the flops placed. The following is an example of constraint that can be added in XDC:

```
set_property IOB true [get_cells -hierarchical -filter {NAME =~*mem_ben_reg*}]
```

For ports Address, CRE, and WAIT the constrains can be the following:

```
set_max_delay -datapath_only -from [get_cells -hier -regexp -filter { REF_NAME =~
FD.* } .*mem_a_reg_reg.*] -to [get_ports linear_flash_addr*] 6.000
```

Using the STARTUPE Primitive

The Flash access through the STARTUP Primitive parameter is applicable in UltraScale™ architecture and later devices. When this parameter is enabled, STARTUPE3 primitive is included in the design and becomes part of the core after the configuration of the FPGA.

See the answer records or the device user guide to understand more about the functionality and use of the STARTUP primitive.

STARTUP Primitive Included in Design

The core behavior and ports are as follows:

- The `mem_dq_*[3:0]` and `mem_cen[0]` ports from the core are interfaced with the STARTUP primitive. The STARTUP can also be used in the pre-configuration process of the FPGA where the external slave memory is configured prior to the FPGA.
- After configuration of the FPGA, the `mem_dq_*[3:0]` and `mem_cen[0]` ports drive the corresponding ports of the primitive which are directly connected to external slave memory. These signals are not available as the external port of the core.
- Instantiating STARTUP affects the maximum frequency of `s_axi_aclk` and `rdclk`.

STARTUP Primitive is Not in Design

The core behavior and ports are as follows:

- As the `mem_dq_*[3:0]` and `mem_cen` ports are part of the core and no primitive is instantiated in the core, these ports are available as external ports of the core and they are placed in an IOB at a user-configured location.

Timing Diagrams

Figure 3-4 is the read timing representation for Sync SRAM memories.

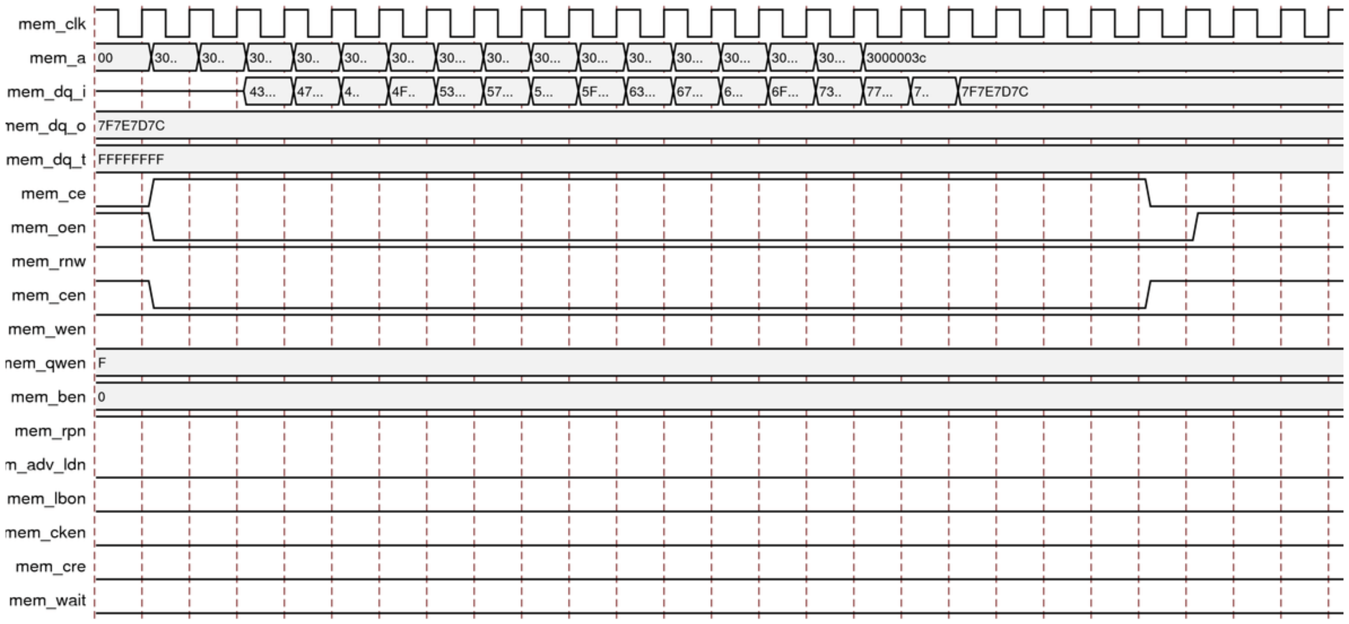


Figure 3-4: **Read Timing Representation of 32-bit Sync SRAM with Pipeline Delay 2**

Figure 3-5 is the write timing representation for Sync SRAM memories.

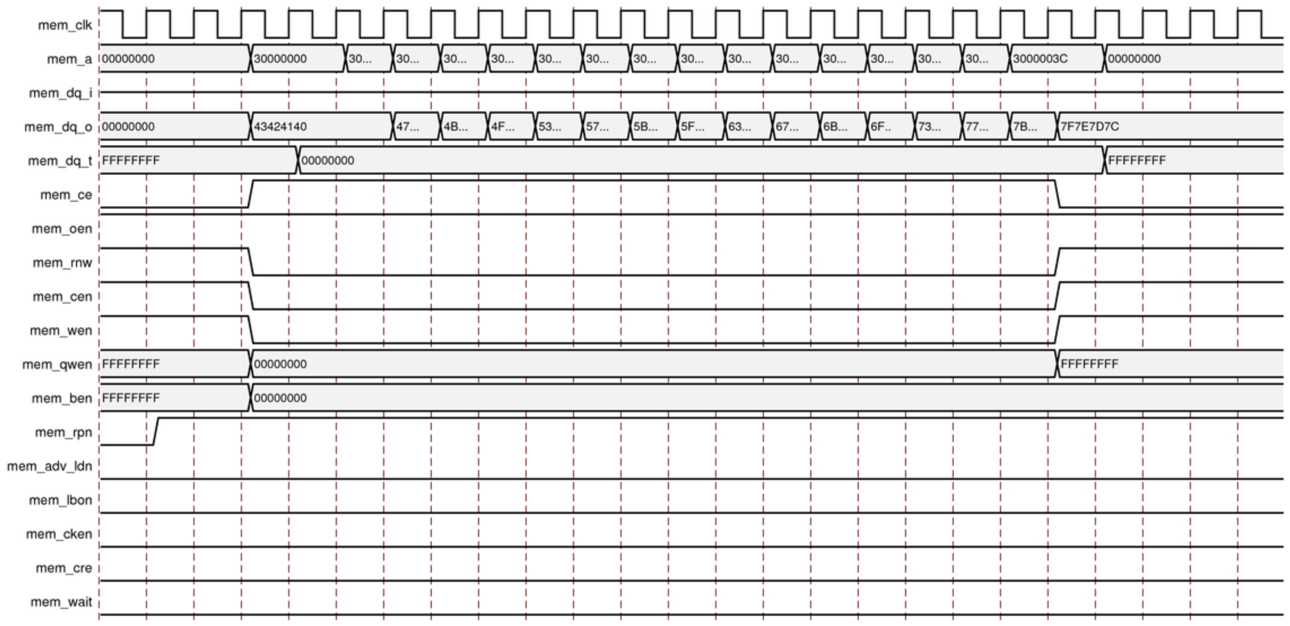


Figure 3-5: **Write Timing Representation of 32-bit Sync SRAM with Pipeline Delay 2**

Figure 3-6 is the read timing representation for Async SRAM memories.

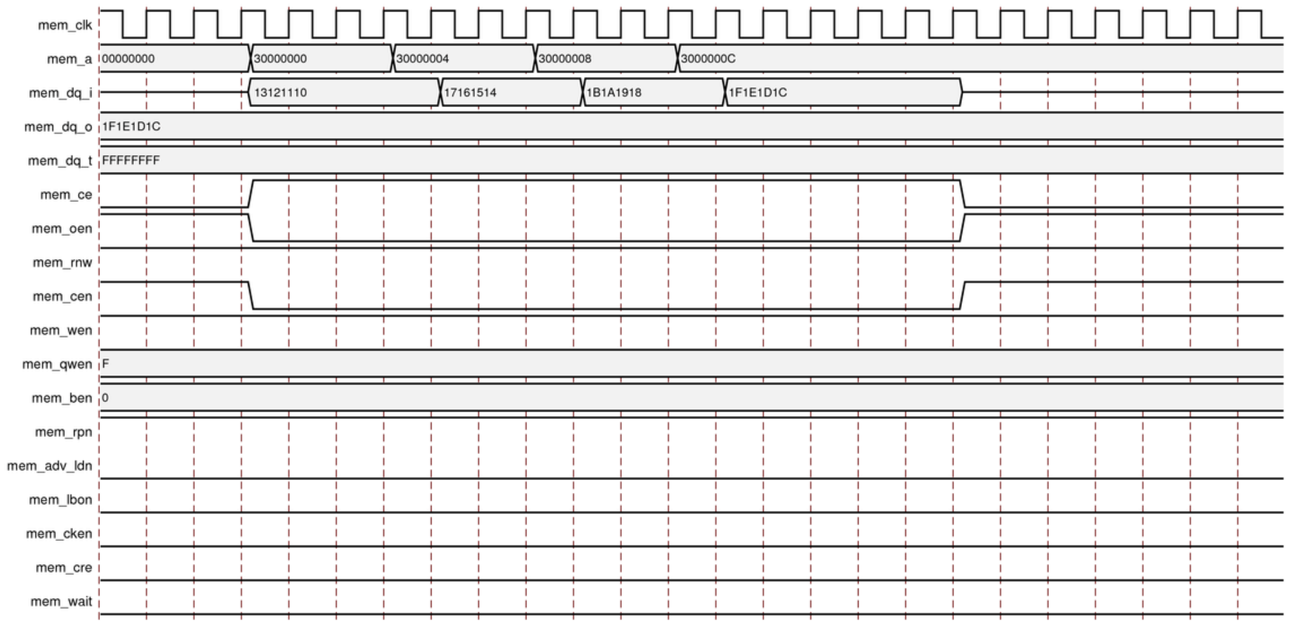


Figure 3-6: **Read Timing Representation of 32-bit Async SRAM**

Figure 3-7 is the write timing representation for Async SRAM memories.

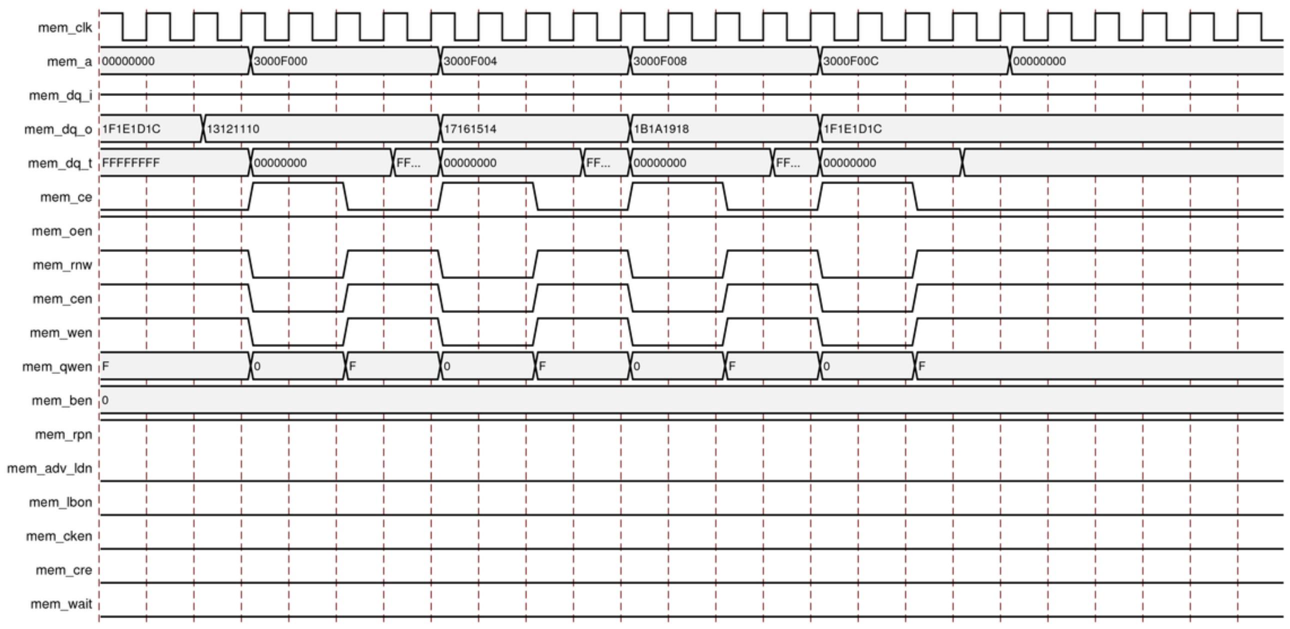


Figure 3-7: **Write Timing Representation of 32-bit Async SRAM**

Figure 3-8 is the read timing representation for Linear Flash memories.

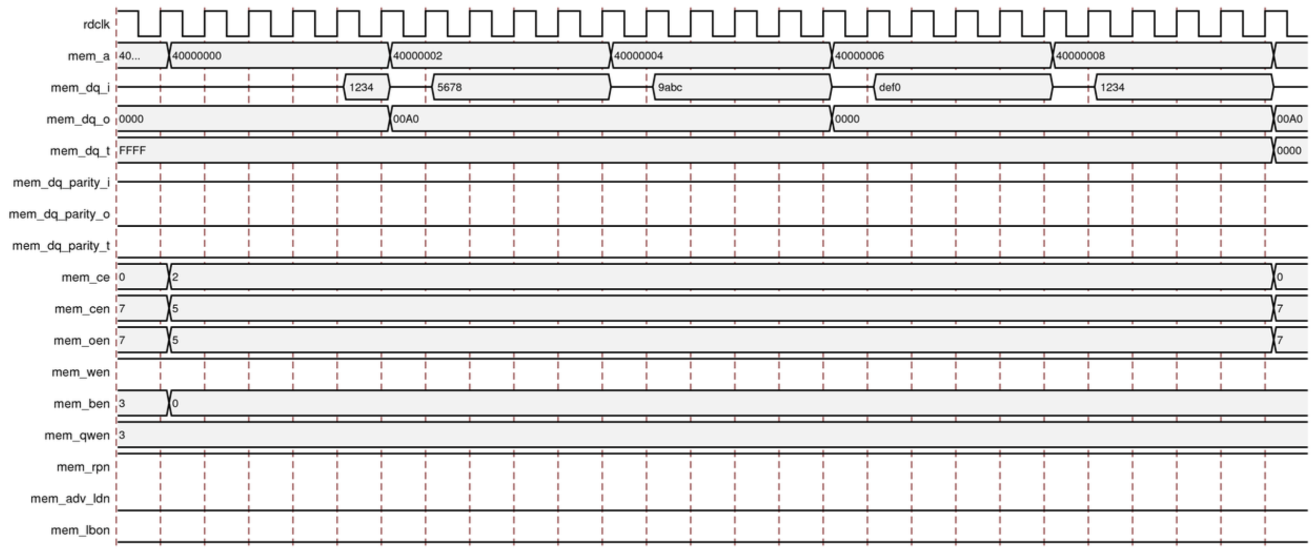


Figure 3-8: **Read Timing Representation of Linear Flash Memories**

Figure 3-9 is the write timing representation for Linear Flash memories.

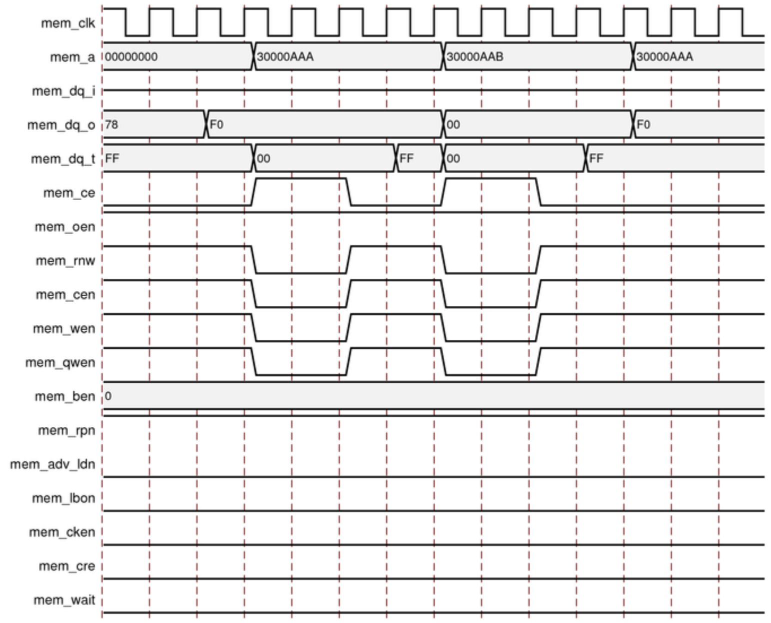


Figure 3-9: **Write Timing Representation of Linear Flash Memories**

Figure 3-10 is the read timing representation for Page Mode Flash memories.

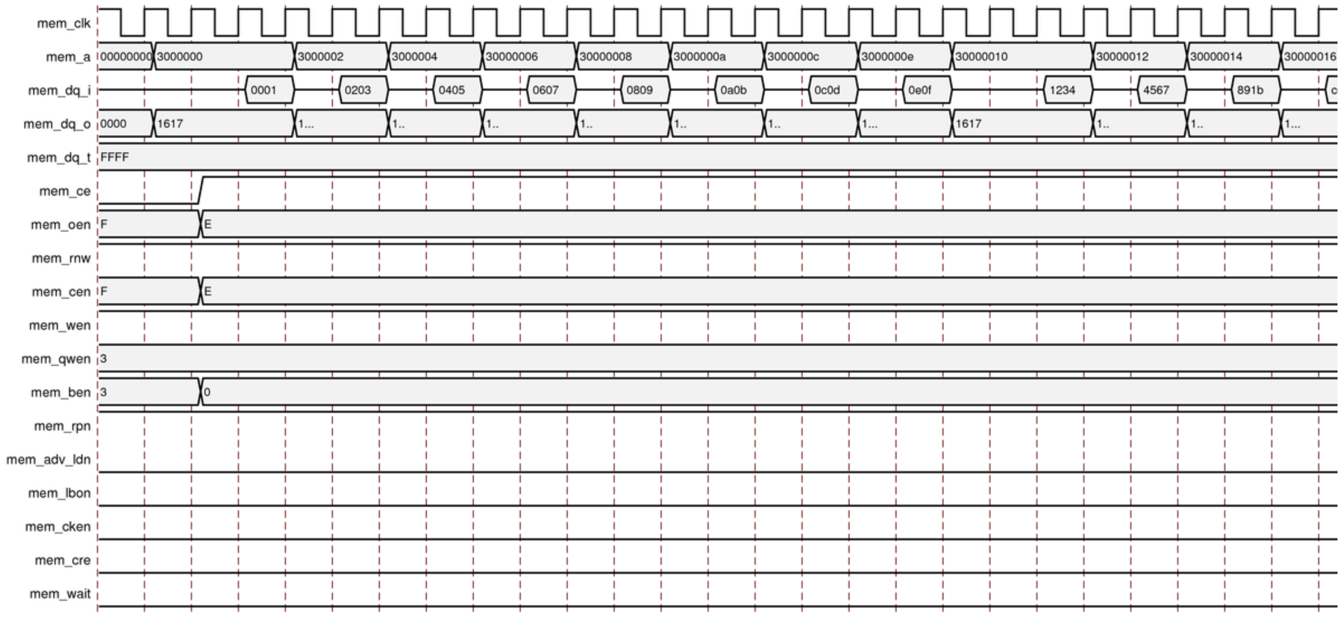


Figure 3-10: **Read Timing Representation for Page Mode Flash Memories**

Figure 3-11 is the read timing representation for Micron Flash memories.

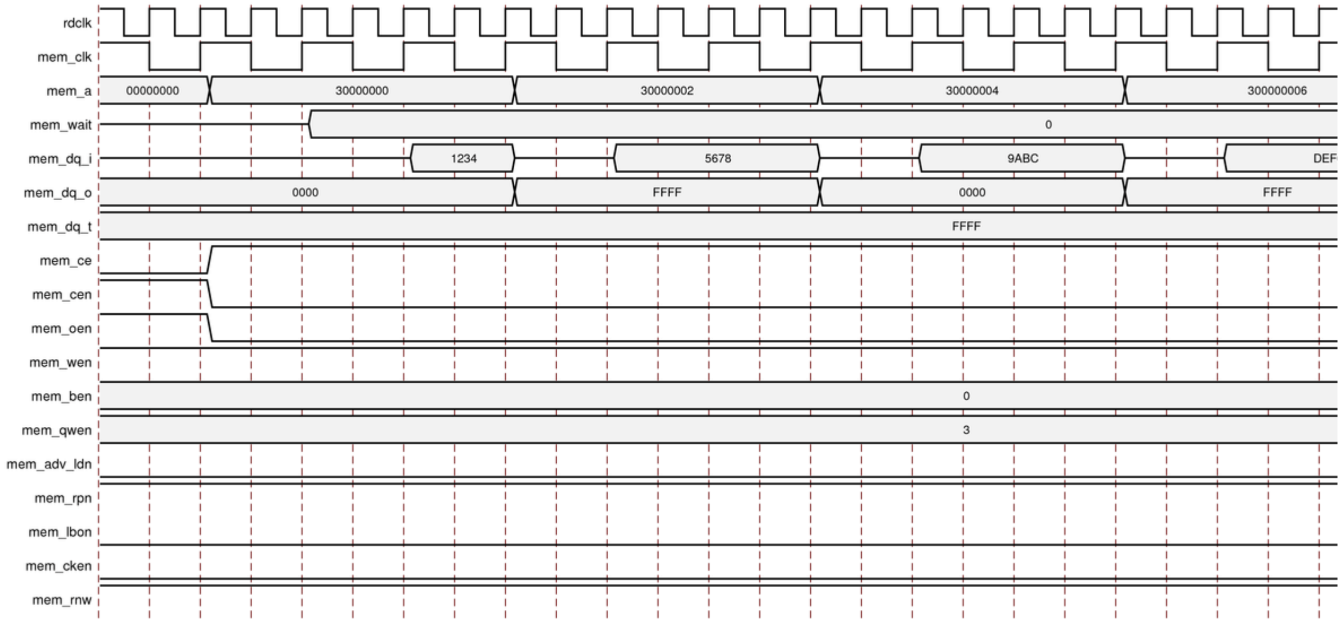


Figure 3-11: **Read Timing Representation for Micron Flash Memories**

Figure 3-12 is the write timing representation for Micron Flash memories.

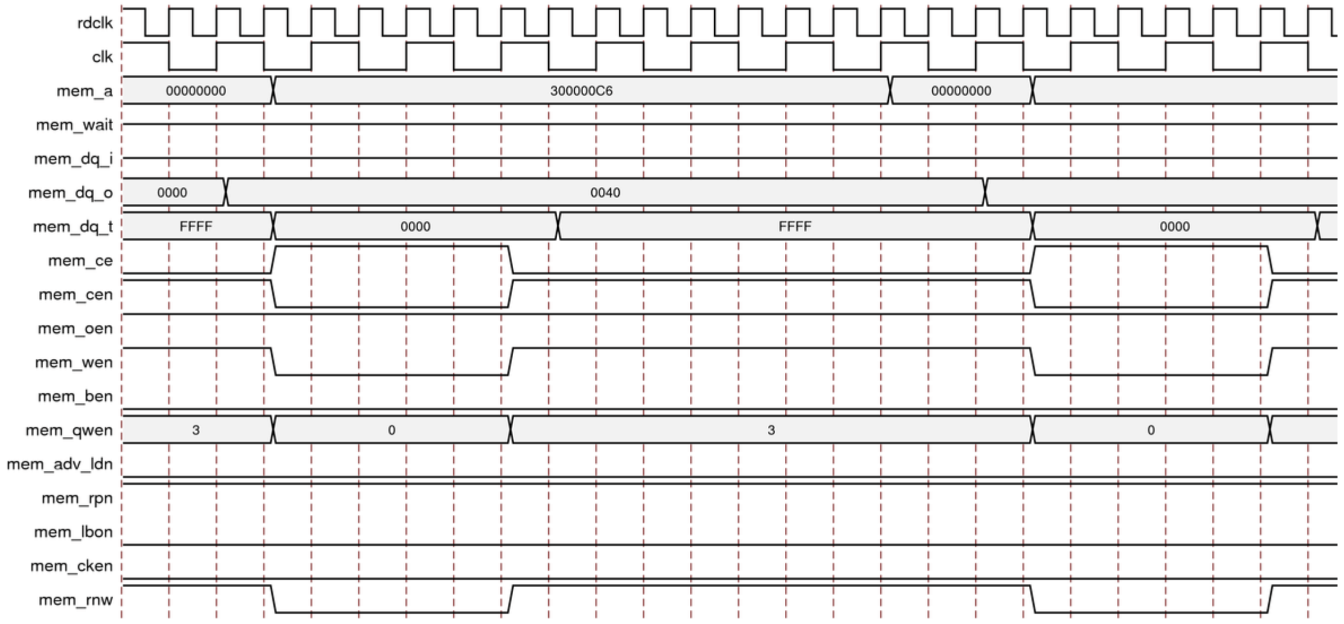


Figure 3-12: **Write Timing Representation for Micron Flash Memories**

Figure 3-13 is the Sync Read timing representation for Micron Flash memories.

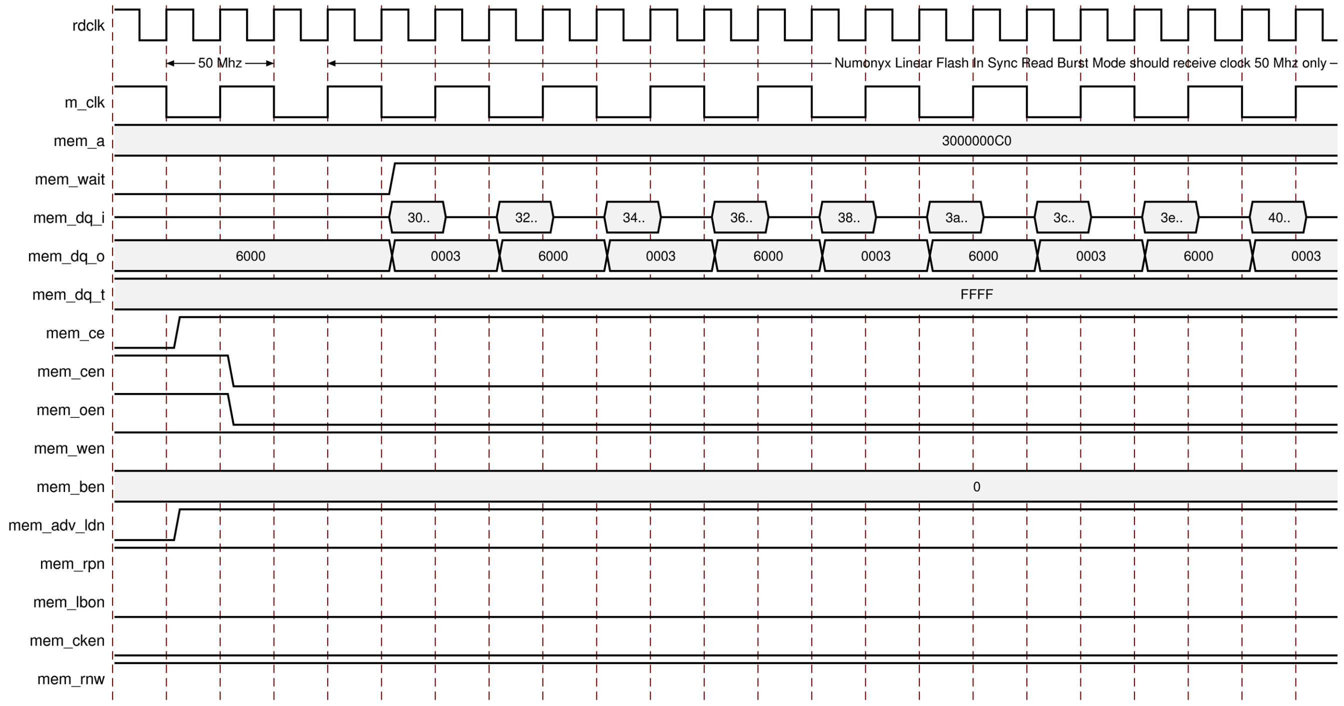


Figure 3-13: **Sync Read Timing Representation for Micron Flash Memories**

The sequence of Read/Write operations is updated to have one cycle difference between Address bus and CE/CEN and OEN/WEN both during assertion and deassertion from 2015.3.

Earlier all of these signals set asserted/deasserted at the same edge.

Design Flow Steps

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

- *Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator* (UG994) [Ref 5]
- *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 3]
- *Vivado Design Suite User Guide: Getting Started* (UG910) [Ref 6]
- *Vivado Design Suite User Guide: Logic Simulation* (UG900) [Ref 7]

Customizing and Generating the Core

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

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

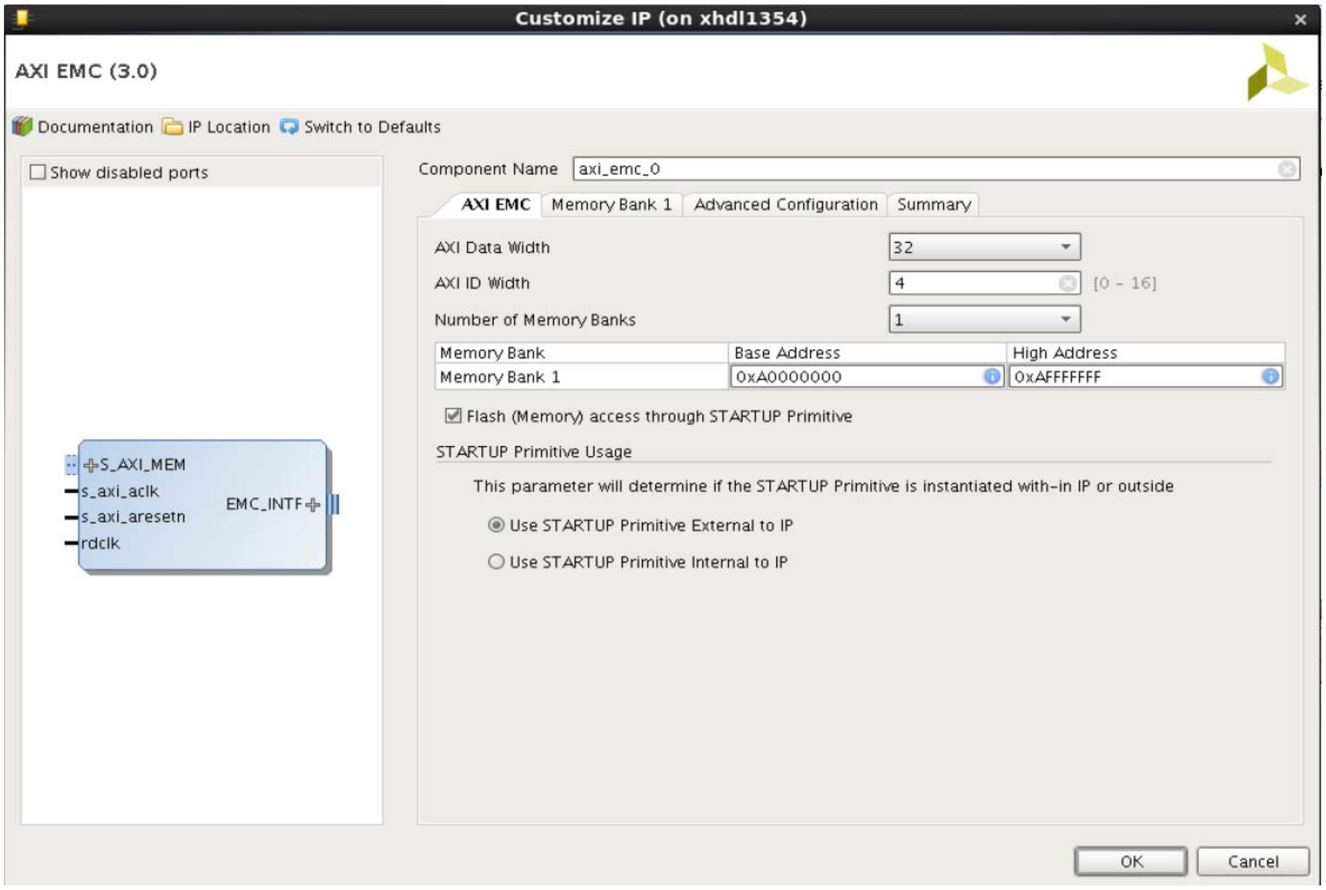
You can customize the IP for use in your design by specifying values for the various parameters associated with the IP core using the following steps:

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

For details, see the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 3] and the *Vivado Design Suite User Guide: Getting Started* (UG910) [Ref 6].

Note: Figures in this chapter are illustrations of the Vivado IDE. This layout might vary from the current version.

Figure 4-1 shows the Customize IP dialog box for AXI EMC.



Component Name: axi_emc_0

AXI Data Width: 32

AXI ID Width: 4 [0 - 16]

Number of Memory Banks: 1

Memory Bank	Base Address	High Address
Memory Bank 1	0xA0000000	0xAFFFFFFF

[✓] Flash (Memory) access through STARTUP Primitive

STARTUP Primitive Usage

This parameter will determine if the STARTUP Primitive is instantiated with-in IP or outside

[✓ Use STARTUP Primitive External to IP]

Use STARTUP Primitive Internal to IP

Figure 4-1: **AXI EMC Customize IP Dialog Box**

The AXI EMC options are as follows:

AXI Data Width

If the targeted memory is set to Micron® Flash memory and Sync burst is enabled, the AXI4 Data Bus Width must be set to 32-bit. For any other memory, the AXI4 Data Bus Width can be set to either 32 or 64 bits.

AXI ID Width

ID width of AXI interface, allowed values are 0 to 16.

Note: For IP integrator, ID Width is auto-computed which means that you are not allowed to override it.

Maximum Data Width

This option should be set to the maximum memory data width configured across all memory banks. Maximum Data Width option can be set as 8, 16, 32, or 64 bits wide.

No. of Memory Banks

Number of memory banks: The valid range for Memory Banks is from 1 to 4. For each memory bank a tab is provided for bank specific memory configurations. The tab needs to set for the memory type and for the timing parameters associated with the selected memory type.

Base Address

Per Bank Base Address for the Memory, the base address for each enabled bank must be set as per the System Address Map requirement. For more details on Memory Address Generation, see Address Map Description, page 21.

High Address

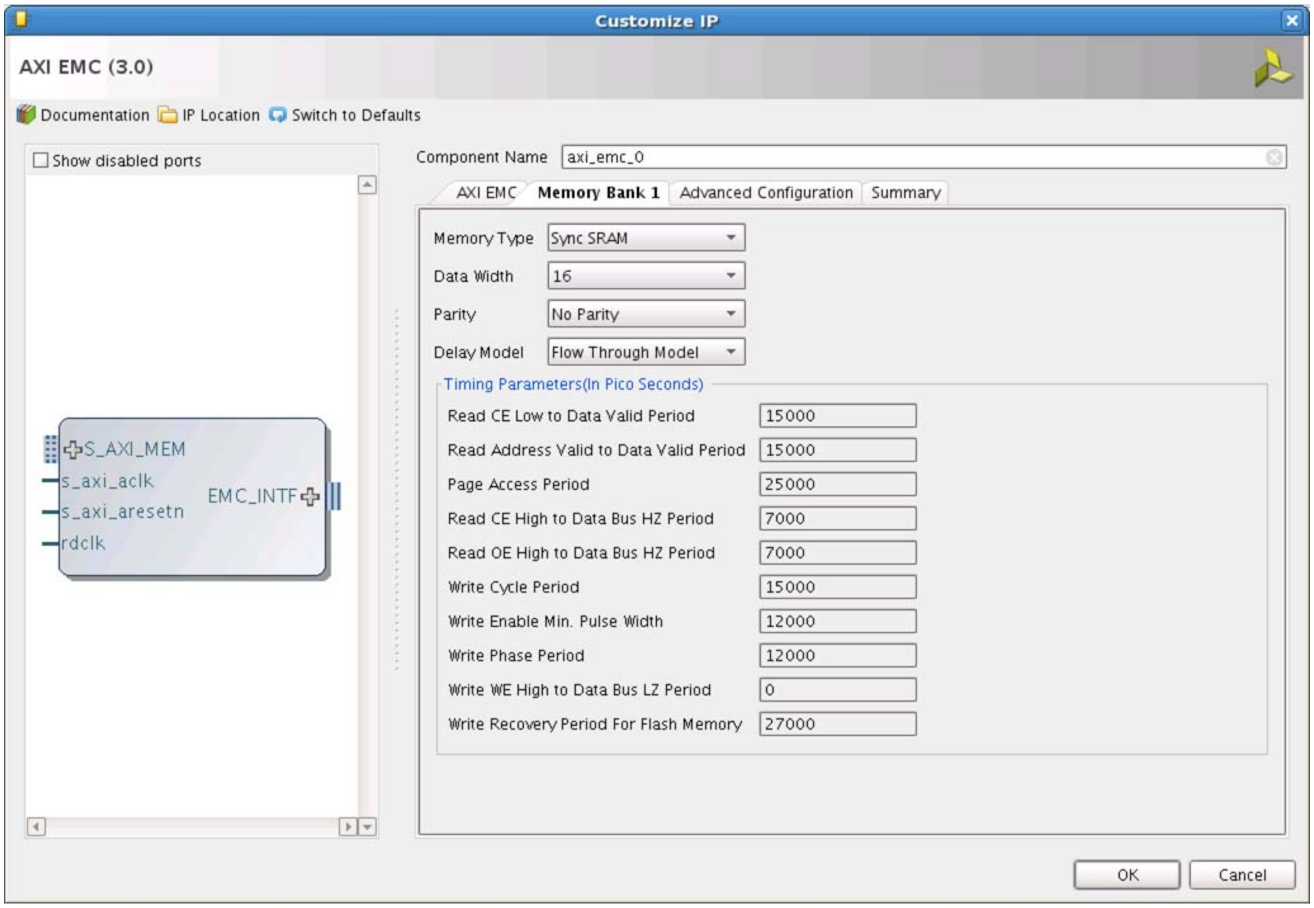
Per Bank High Address for the Memory, the High address for each enabled bank must be set as per the System Address Map requirement. For more details on Memory Address Generation, see Address Map Description, page 21.

Flash (Memory) Access through STARTUP Primitive

Includes the STARTUP primitive in the design when checked, omits the primitive when not checked. The STARTUPE3 primitive is featured in UltraScale™ architecture and later devices. It is useful in sharing the Data pins and CEN pin with external BPI flash memory. This parameter selection affects IOB attributes generated in XDC, when enabled IOB Attributes are not applied to pins directly connected to STARTUP primitive.

This parameter determines if STARTUP Primitive instantiation is within the IP or outside of it.

Figure 4-2 shows the Customize IP dialog box for each memory bank of AXI EMC core.



Component Name: axi_emc_0

Memory Type: Sync SRAM
Data Width: 16
Parity: No Parity
Delay Model: Flow Through Model

Timing Parameters(In Pico Seconds)

Read CE Low to Data Valid Period	15000
Read Address Valid to Data Valid Period	15000
Page Access Period	25000
Read CE High to Data Bus HZ Period	7000
Read OE High to Data Bus HZ Period	7000
Write Cycle Period	15000
Write Enable Min. Pulse Width	12000
Write Phase Period	12000
Write WE High to Data Bus LZ Period	0
Write Recovery Period For Flash Memory	27000

Figure 4-2: **Memory Bank of AXI EMC**

Memory Type

Memory Type for the selected bank can configured as Sync SRAM, Async SRAM, Linear Flash, Page Mode Flash, PSRAM, or Micron Flash. For the supported memory combinations across memory banks, see Allowable Memory Combination, page 22.

Data Width

Memory data width for the selected bank, Data Width of the Memory connected to the bank can be set as 8, 16, 32, or 64 bits wide.

Parity

Parity is applicable only when Memory Type selected for the bank is Sync SRAM. Parity for the memory can be set as No parity, Odd Parity, or Even Parity.

Delay Model

This configuration option is applicable only when Memory Type selected for the bank is Sync SRAM. The Delay Model for the Sync SRAM can be set as a Flow-Through model or Pipeline Model. See Timing Diagrams, page 34.

Read CE Low to Data Valid Period

Read Chip Enable to Data Valid Period specification for the memory in the selected bank. This configuration option is applicable only when memory type in the bank is asynchronous. This option is equivalent to tACE for asynchronous SRAM and tELQV for flash memory in the respective memory device data sheets.

Read Address Valid to Data Valid Period

Read Address Valid to Data Valid Period specification for the memory in the selected bank. This configuration option is applicable only when memory type in the bank is asynchronous. This option is equivalent to tAA for asynchronous SRAM and tAVQV for flash memory in the respective memory device data sheets.

IMPORTANT: *Read cycle time for AXI EMC core is determined by Read CE Low to Data Valid Period and Read Address Valid to Data Valid Period configuration settings and is the Maximum[Read CE Low to Data Valid Period, Read Address Valid to Data Valid Period].*

Page Access Period

Page Access Period specification for the asynchronous flash memory in the selected bank. This configuration option is applicable only when memory type in the bank is Page Mode Flash Memory. This option is equivalent to tPACC in the Page Mode Flash device data sheet and must be assigned only when Memory Type selected for the bank is Page Mode Flash.

Read CE High to Data Bus HZ Period

Read CE High to data bus High impedance period specification for the memory in the selected bank. This configuration option is applicable only when memory type in the bank is asynchronous. This option is equivalent to tHZCE for asynchronous SRAM and tEHQZ for flash memory in the respective memory device data sheets.

Read OE High to Data Bus HZ Period

Read OE High to data bus High impedance period specification for the memory in the selected bank. This configuration option is applicable only when memory type in the bank is asynchronous. This option is equivalent to tHZOE for asynchronous SRAM and tGHQZ for flash memory in the respective memory device data sheets.

IMPORTANT: *Read cycle recovery to write period for AXI EMC core is determined by Read CE High to Data Bus HZ Period and Read OE High to Data Bus HZ Period configuration settings and is the MAXIMUM[Read CE High to Data Bus HZ Period, Read OE High to Data Bus HZ Period].*

Write Cycle Period

Write cycle time of the memory bank. This configuration option is applicable only when memory type in the bank is asynchronous. This option is equivalent to tWC for asynchronous SRAM and tCW for flash memory in the respective memory device data sheets.

The AXI EMC core uses this parameter to hold CEN Low for each write at the memory interface.

Write Enable Minimum Pulse Width

Write enable minimum pulse width duration of the memory bank. This configuration option is applicable only when memory type in the bank is asynchronous. This option is equivalent to tWP for Asynchronous SRAM and tPWE for flash memory in the respective memory device data sheets.

The AXI EMC core uses this parameter to hold WEN Low for each write at the memory interface.

IMPORTANT: *CEN Low and WEN Low time generated by the AXI EMC core is determined by Write Cycle Period and Write Enable Minimum Pulse Width configuration settings. It is the MAXIMUM[Write Cycle Period, Write Enable Minimum Pulse Width].*

Write Phase Period

Write cycle phase time period of the memory bank. This configuration option is applicable only when memory type in the bank is asynchronous.

The AXI EMC core uses this parameter to determine the number of NOP cycles between two consecutive writes at the memory side. When this parameter is set to zero, CEN goes High for one clock cycle between writes.

Write WE High to Data Bus LZ Period

Write cycle write enable High to data bus Low impedance duration of memory bank. This configuration option is applicable only when memory type in the bank is asynchronous. This option is equivalent to tLZWE for asynchronous SRAM and tWHGL for flash memory in the respective memory device data sheets. This option is also used to meet write recovery to read time requirements of the memory.

Write Recovery Period for Flash Memory

Write recovery period for flash memories of the memory bank. This configuration option is applicable only when memory type in the bank is any flash memory or PSRAM. This option is equivalent to tWR for flash memory in the respective memory device data sheets.

The AXI EMC core uses this parameter to determine the number of NOP cycles at memory interface after each AXI WLAST.

RECOMMENDED: *You should provide the value for C_AXI_CLK_PERIOD_PS which is equivalent to AXI clock used. This value is used for internal calibration of counters when Asynchronous memories are used. By default this is set to 10,000 which is a value corresponding to 100 MHz AXI clock. If the AXI clock connected is not 100 MHz, the parameter has to be set using command* "`set_property CONFIG.C_AXI_CLK_PERIOD_PS {time period in picoseconds of AXI clock} [get_ips <component name of emc IP created>]`" *in the Tcl Console.*

Figure 4-3 to Figure 4-5 show the AXI EMC timing diagrams.

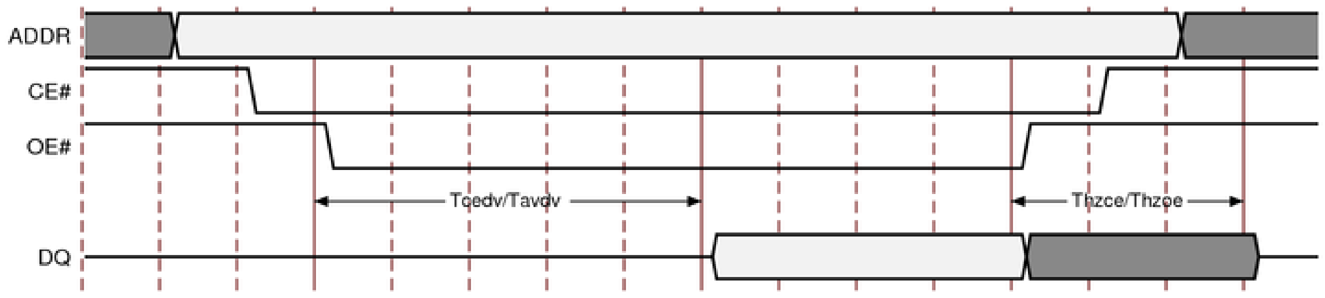


Figure 4-3: **AXI EMC Read Timing**

Notes:

1. Tcedv/Tavqv refers to Maximum of "Read CE Low to Data Valid Period" and "Read Address Valid to Data Valid Period."
2. Thzce/Thzoe refers to Maximum of "Read CE High to Data Bus HZ Period" and "Read OE High to Data Buz HZ Valid Period."

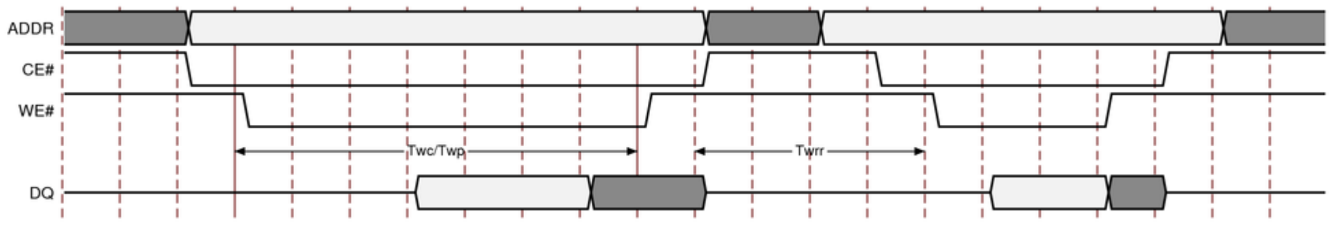


Figure 4-4: **AXI EMC Write Timing**

Notes:

1. Twc/Twp refers to Maximum of "Write Cycle Period" and "Write Enable Minimum Pulse Width."
2. Twrr refers to Maximum of "Write Recovery Period of Flash."

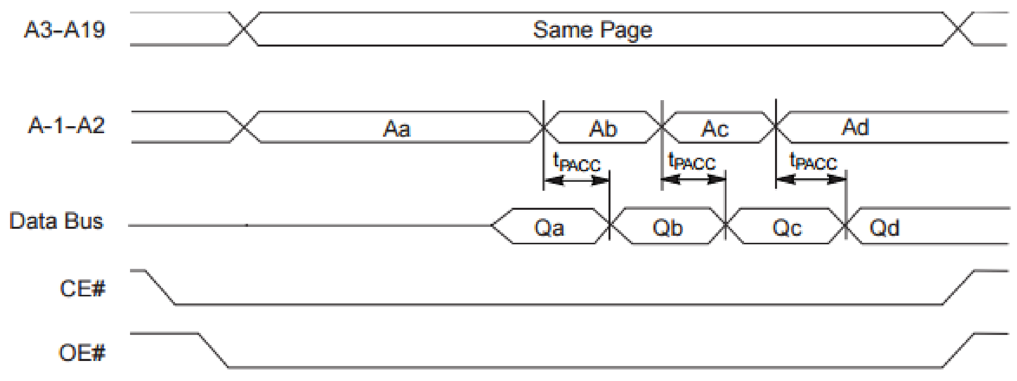
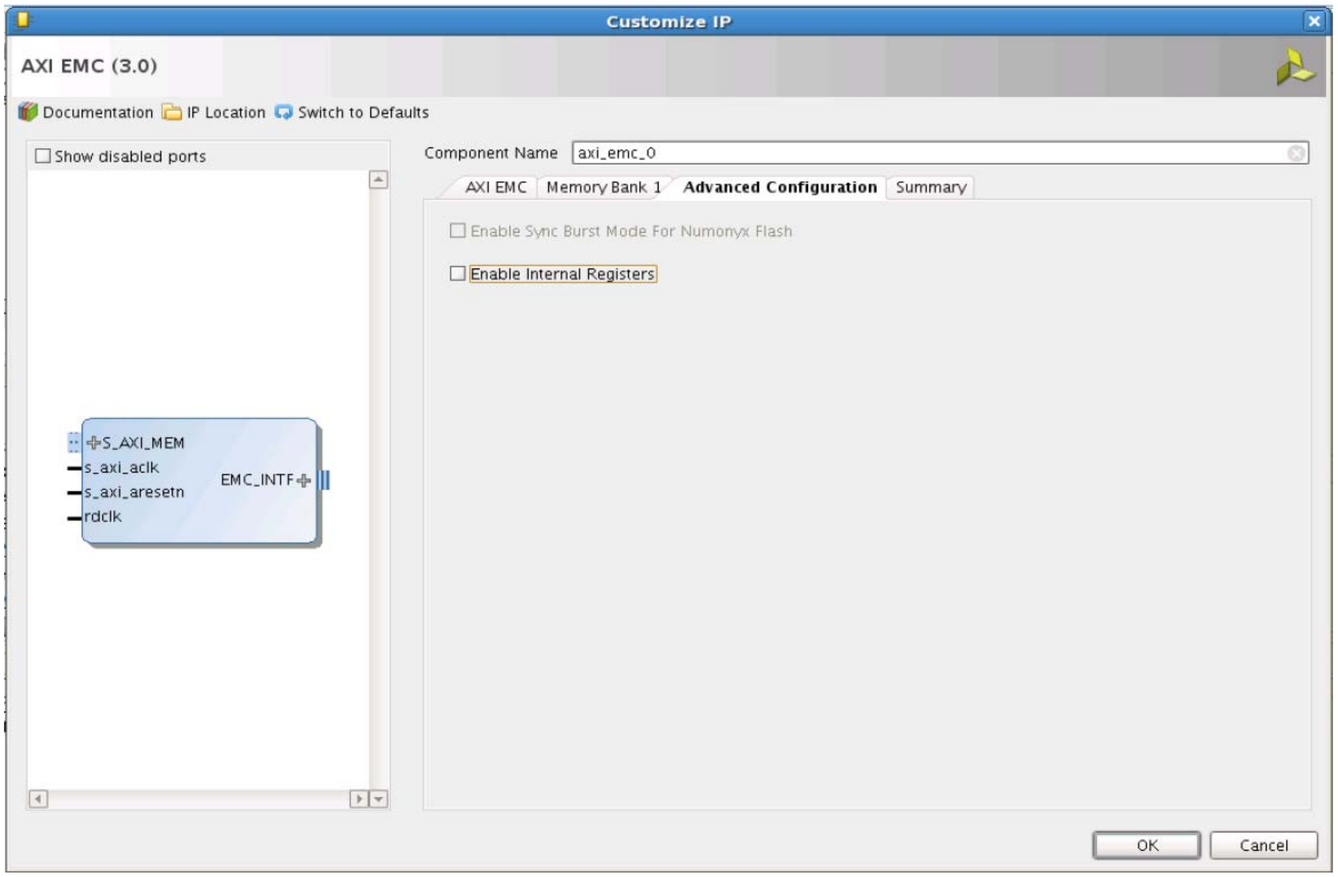


Figure 4-5: **AXI EMC Page Access Timing**

Figure 4-6 shows the Customize IP dialog box with the Advanced Configuration tab of AXI EMC core.



Component Name axi_emc_0

☐ Enable Sync Burst Mode For Numonyx Flash

☐ Enable Internal Registers

Figure 4-6: **Advanced Configuration of AXI EMC**

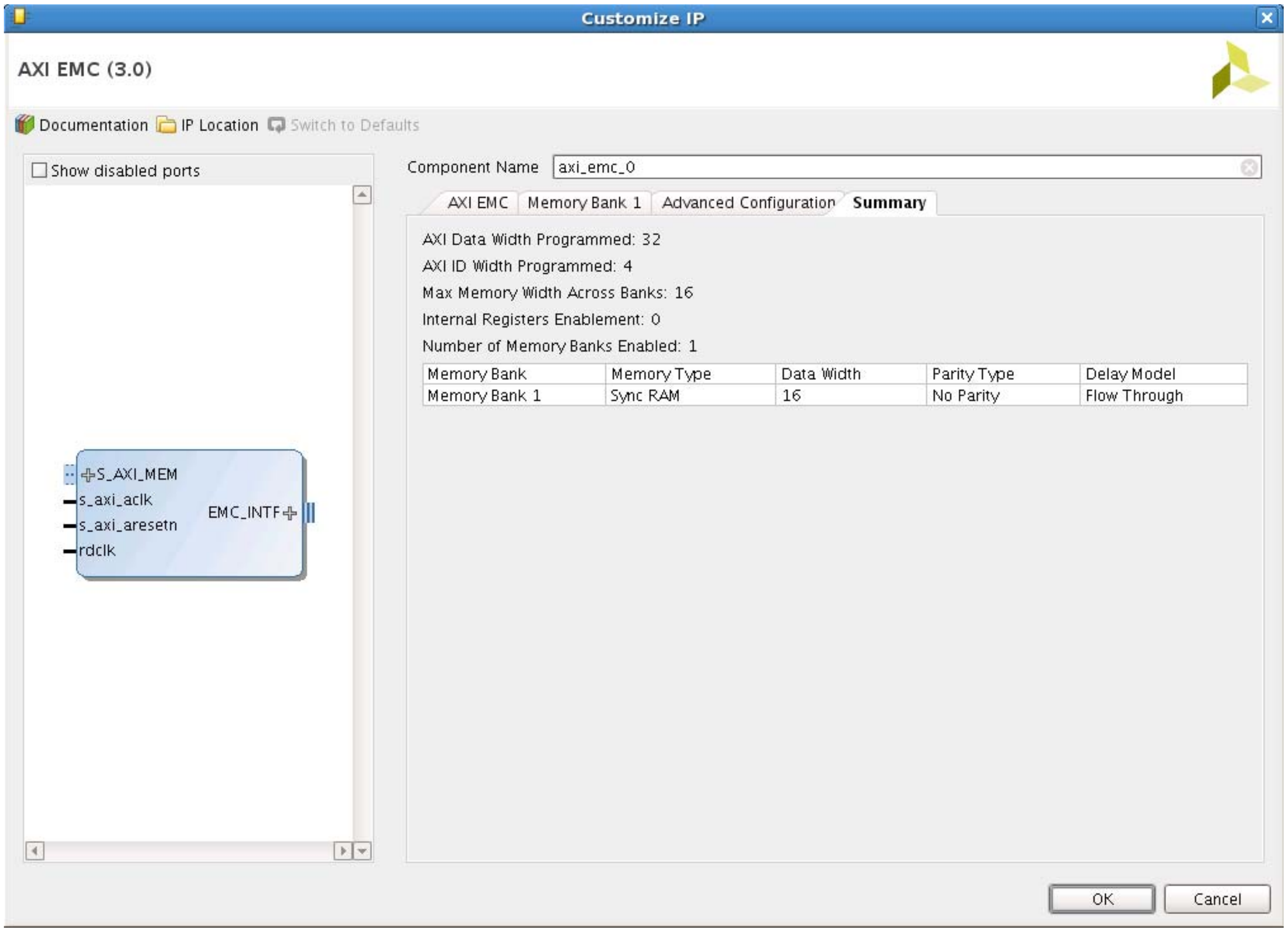
Enable Burst Mode for Micron Flash

When set, enables Burst mode for Micron Flash memory. By default Burst mode is disabled for Micron Flash memory. This option is not applicable when Micron Flash is selected with other possible Asynchronous memories.

Enable Internal Registers

When set, includes AXI4-LITE interface for accessing internal core registers. This option is applicable when Sync SRAM, PSRAM, or Micron Flash memory in Sync mode is set for any of the memory bank.

Figure 4-7 shows the Customize IP dialog box with the Summary tab of AXI EMC core.



Component Name axi_emc_0

AXI Data Width Programmed: 32
AXI ID Width Programmed: 4
Max Memory Width Across Banks: 16
Internal Registers Enablement: 0
Number of Memory Banks Enabled: 1

Memory Bank	Memory Type	Data Width	Parity Type	Delay Model
Memory Bank 1	Sync RAM	16	No Parity	Flow Through

Figure 4-7: **Summary of AXI EMC**

User Parameters

Table 4-1 shows the relationship between the fields in the Vivado IDE and the User Parameters (which can be viewed in the Tcl Console).

Table 4-1: **Vivado IDE Parameter to User Parameter Relationship**

Vivado IDE Parameter[(1)]	User Parameter[(1)]	Default Value[(1)]
AXI Data Width	C_S_AXI_MEM_DATA_WIDTH	32
AXI ID Width	C_S_AXI_MEM_ID_WIDTH	4
Number of Memory Banks	C_NUM_BANKS_MEM	1
Memory Bank Base Address Valid HEX values for number of banks selected with no overlaps.	C_S_AXI_MEM*_BASEADDR[(3)]	0x*0000000 "*" depends on bank dealing with starts from A to D. A for first bank, B for second, etc.
Memory Bank High Address Valid HEX values for number of banks selected with no overlaps.	C_S_AXI_MEM*_HIGHADDR[(3)]	0x*FFFFFFF "*" depends on bank dealing with starts from A to D. A for first bank, B for second, etc.
Memory Type Value 0 to Sync SRAM 1 to Async SRAM 2 to Linear Flash 3 to Page Mode Flash 4 to PSRAM 5 to Micron Flash	C_MEM*_TYPE[(3)]	0 This value corresponds to Sync SRAM.
Data Width 8, 16, 32, and 64 for Sync and Async SRAMs and fixed value of 16 for other memory types. 64 is allowed only if C_S_AXI_MEM_DATA_WIDTH selected is 64.	C_MEM*_WIDTH[(3)]	16
Parity Applicable only for Sync SRAM memories and should be same for all the banks enabled. Value 0 to No Parity 1 to Odd Parity 2 to Even Parity	C_PARITY_TYPE_MEM_*[(3)]	0
Delay Model Applicable only for Sync SRAM memories. Value 1 to flow through 2 to Pipeline Delay.	C_SYNCH_PIPEDELAY_*[(3)]	1
Read CE Low to Data Valid Period[(2)]	C_TCEDV_PS_MEM_*[(3)]	15000
Read Address Valid to Data Valid Period[(2)]	C_TAVDV_PS_MEM_*[(3)]	15000
Page Access Period[(2)]	C_TPACC_PS_FLASH_*[(3)]	25000

Table 4-1: **Vivado IDE Parameter to User Parameter Relationship *(Cont'd)***

Vivado IDE Parameter[1]	User Parameter[1]	Default Value[1]
Read CE High to Data Bus HZ Period[2]	C_THZCE_PS_MEM_*[3]	7000
Read OE High to Data Bus HZ Period[2]	C_THZOE_PS_MEM_*[3]	7000
Write Cycle Period[2]	C_TWC_PS_MEM_*[3]	15000
Write Enable Min. Pulse Width[2]	C_TWP_PS_MEM_*[3]	12000
Write Phase Period[2]	C_TWPH_PS_MEM_*[3]	12000
Write WE High to Data Bus LZ Period[2]	C_TLZWE_PS_MEM_*[3]	0
Write Recovery Period for Flash Memory[2]	C_WR_REC_TIME_MEM_*[3]	27000
Enable Sync Burst Mode for Micron Flash Applicable only for Micron Flash. Allowable values are 0 and 1.	C_LINEAR_FLASH_SYNC_BURST	0
Enable Internal Registers Applicable only when memory selected is Sync SRAM, Async SRAM, PSRAM, or Micron Flash. Allowable values are 0 and 1.	C_S_AXI_EN_REG	0
Flash (Memory) access through STARTUP Primitive possible values are FALSE: 0 and TRUE: 1	C_USE_STARTUP	0
STARTUP Primitive instantiation within or outside of IP STARTUP external to IP: 0 STARTUP internal to IP: 1	C_USE_STARTUP_INT	0

Notes:

1. Parameter values are listed in the table where the Vivado IDE parameter value differs from the user parameter value. Such values are shown in this table as indented below the associated parameter.
2. Applicable only for Async Memories and flashes should be bases on memory data sheet.
3. "*" depends on number of banks selected, ranges from 0 to "C_NUM_BANKS_MEM – 1."

Output Generation

For details, see the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 3].

Constraining the Core

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

Required Constraints

If external pull-ups/pull-downs are not available on the `MEM_DQ` signals, these constraints should be specified to use pull-up or pull-down resistors, as in this example:

```
set_property PULLDOWN true [get_ports MEM_DQ[0]]
set_property PULLDOWN true [get_ports MEM_DQ[1]]
set_property PULLDOWN true [get_ports MEM_DQ[2]]
...
set_property PULLDOWN true [get_ports MEM_DQ[31]]
```

STARTUP is enabled: The operation frequency of AXI EMC is affected when a STARTUP primitive is enabled. This primitive has its own delay that is not accounted in the entire implementation/timing process.

STARTUPE3 primitive is used to drive the Data pins to the flash. When using the STARTUP primitive, the clock-data relationship changes due to the extra delay added in the data pins path. This delay can vary from as low as 0.1 ns to as high as 7.5 ns based on the device and speed grade. This value can be found in the FPGA data sheet.

Because this delay is not timed by the tool, you must consider the maximum delay for all calculations. Assume this delay is called CCLK_DELAY. For K7-2 this value can range from 0.5 ns to 6.7 ns. This puts the first restriction on the frequency of the clock. The frequency of operation cannot exceed F_{Max1} = (1/CCLK_DELAY) MHz.

Constraining the IP

To understand how the constraints are added, it is important to understand the logic structure around the data pins in AXI EMC and BPI flash memory.

Figure 4-8 shows the logic structure.

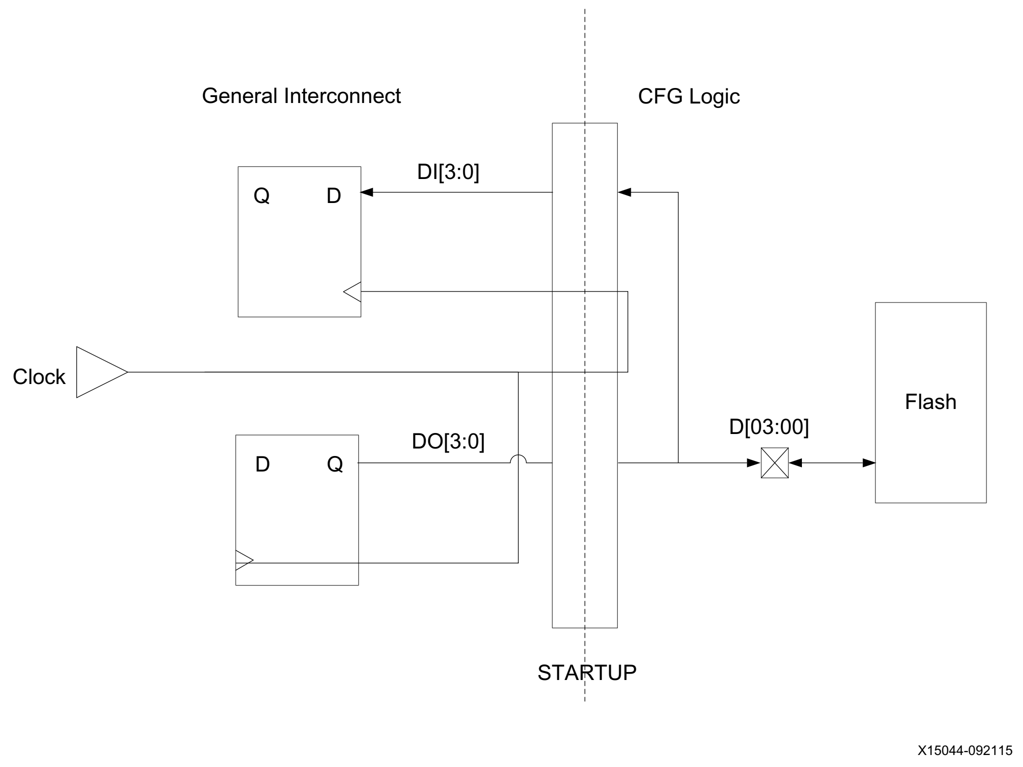


Figure 4-8: **AXI EMC Logic Structure**

The STARTUP primitive adds delay on the data[3:0] pins. This delay is unaccounted for in the tool and is not considered in the timing calculation. For the tool, the timing path ends at data[3:0] internal flops.

To obtain the required constraints you must account for the STARTUP primitive delay.

This takes into account the delay of the flip-flop that generates the data, the routing delay from that flip-flop to Data pins, and the STARTUP primitive delay. Further, to reduce the delay on data pins you must ensure that the delay from the flip-flop to pin is as low as possible. This can be constrained by using set_output_delay and set_input_delay constraints.

```
#### All the delay numbers have to be provided by the user
#### Max delay constraints are used to instruct the tool to place IP near to STARTUPE3
primitive.
#### If needed adjust the delays appropriately
set_max_delay -to [get_pins -hier {*Mem_DQ_I_v_reg[4]/D} -filter {NAME=~*EMC_CTRL_I/
IO_REGISTERS_I*}] 1.500
set_min_delay -to [get_pins -hier {*Mem_DQ_I_v_reg[4]/D} -filter {NAME=~*EMC_CTRL_I/
IO_REGISTERS_I*}] 0.100
set_max_delay -to [get_pins -hier {*Mem_DQ_I_v_reg[5]/D} -filter {NAME=~*EMC_CTRL_I/
IO_REGISTERS_I*}] 1.500
set_min_delay -to [get_pins -hier {*Mem_DQ_I_v_reg[5]/D} -filter {NAME=~*EMC_CTRL_I/
IO_REGISTERS_I*}] 0.100
```

```
set_max_delay -to [get_pins -hier {*Mem_DQ_I_v_reg[6]/D} -filter {NAME=~*EMC_CTRL_I/
IO_REGISTERS_I*}] 1.500
set_min_delay -to [get_pins -hier {*Mem_DQ_I_v_reg[6]/D} -filter {NAME=~*EMC_CTRL_I/
IO_REGISTERS_I*}] 0.100
set_max_delay -to [get_pins -hier {*Mem_DQ_I_v_reg[7]/D} -filter {NAME=~*EMC_CTRL_I/
IO_REGISTERS_I*}] 1.500
set_min_delay -to [get_pins -hier {*Mem_DQ_I_v_reg[7]/D} -filter {NAME=~*EMC_CTRL_I/
IO_REGISTERS_I*}] 0.100
set_max_delay -to [get_pins -hier {*STARTUP*_inst/DO[*]}] 1.500
set_min_delay -to [get_pins -hier {*STARTUP*_inst/DO[*]}] 0.100
set_max_delay -to [get_pins -hier {*STARTUP*_inst/DTS[*]}] 1.500
set_min_delay -to [get_pins -hier {*STARTUP*_inst/DTS[*]}] 0.100
```

Device, Package, and Speed Grade Selections

This section is not applicable for this IP core.

Clock Frequencies

The IP reads the clock frequency value in IPI. This value is then used to calculate the various timing parameters.

Clock Management

This section is not applicable for this IP core.

Clock Placement

This section is not applicable for this IP core.

Banking

This section is not applicable for this IP core.

Transceiver Placement

This section is not applicable for this IP core.

I/O Standard and Placement

This section is not applicable for this IP core.

Simulation

For comprehensive information about Vivado simulation components, as well as information about using supported third-party tools, see the *Vivado Design Suite User Guide: Logic Simulation* (UG900) [Ref 7].

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

Synthesis and Implementation

For details about synthesis and implementation, see the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 3].

Example Design

This chapter contains information about the example design provided in the Vivado® Design Suite.

The top-level example design for the AXI EMC core is `<component_name>_exdes.vhd`.

The complete example design works as a mini-system to show the AXI EMC functionality by performing read and write transactions to memory model (Figure 5-1).

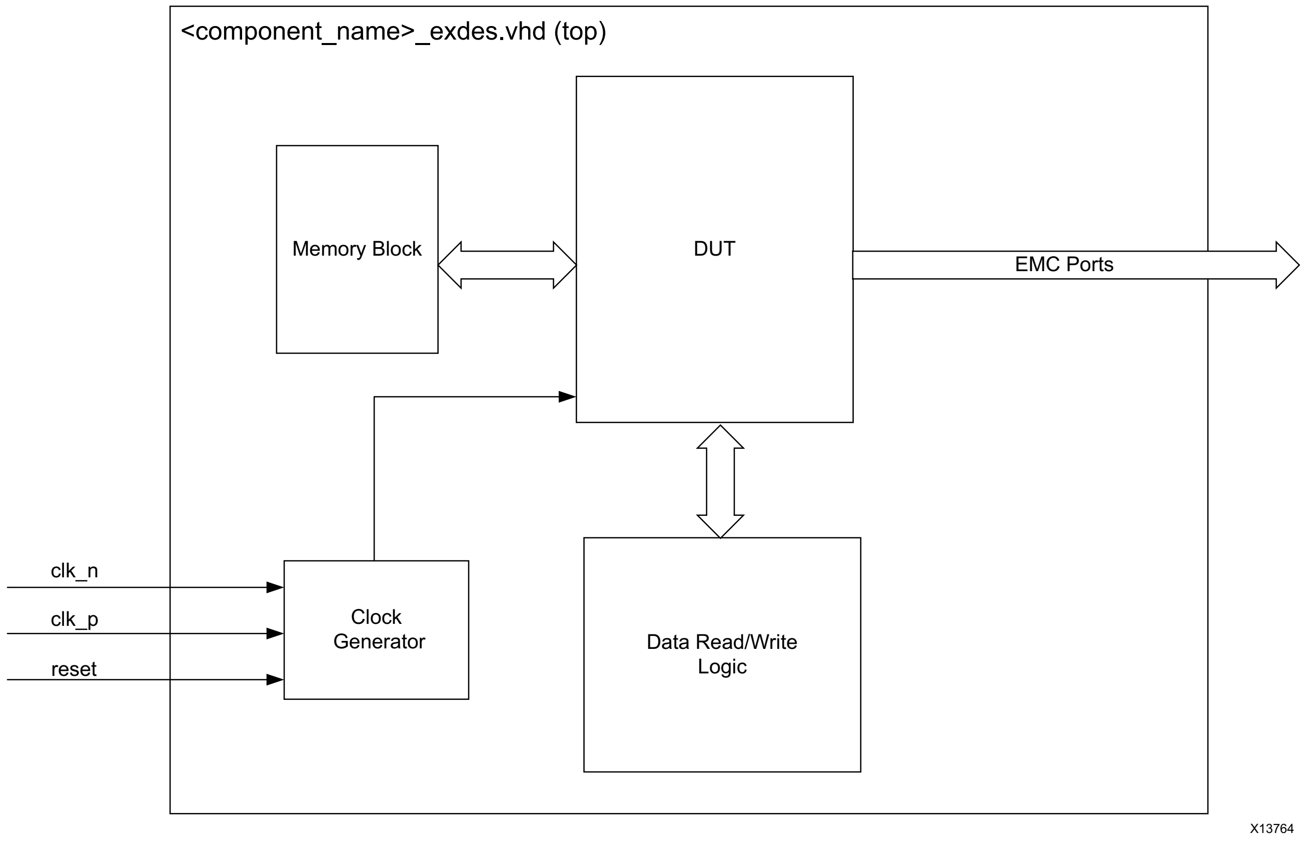


Figure 5-1: **AXI EMC Example Design Block Diagram**

The following is the brief description of individual blocks:

- **Clock Generator** – EMC example design makes use of the clocking wizard to supply clocks and resets to all other blocks in the design. The clocking wizard is configured to provide 100 MHz clock to all the blocks, and the locked signal from the wizard is appropriately used as Resets.
- **Memory** – Block Memory Generator (BMG) core is used to serve for memory storage in the example design. It is configured for 32 × 1,024 memory whose first 256 locations are initialized using a coefficient file. The example design uses the first half of the BMG as the source of data to write data to the memory model through EMC, and it uses the second half to store data read from model through EMC.
- **Data Read/Write Path** – Example design uses the Central Direct Memory Access (CDMA) core to generate the required AXI instructions for data flow between BMG and EMC in both directions. The design uses two instances of CDMA:
  - One instance that reads the channel connected to BMG, and writes the channel connected to EMC. This is called as Write CDMA in design and is used to write data from BMG to EMC.
  - Another CDMA instance that reads the channel connected to EMC and writes the channel connected to BMG. This is called Read CDMA to read data from EMC and write to BMG.
- **AXI Traffic Generator** – Example design makes use of the AXI Traffic Generator (ATG) core in system test mode to control the transaction sequence. ATG is connected to both CDMAs to program them for the source address, destination address, and bytes to transfer registers based on which AXI instructions are generated by the CDMA. The ATG takes four COE files as input, and gives `done` and `status` pins as outputs which determines the result of the tests. It instantiates IOBUF to take in I, O, and T pins of the EMC, and gives out a single I/O pin that communicates with the memory model.

IMPORTANT: *The example design is not supported when the start-up block is enabled*

Implementing the Example Design

After following the steps described in Customizing and Generating the Core, page 41 to generate the core, implement the example design as follows:

1. Right-click the core in the Hierarchy window, and select **Open IP Example Design**.
2. A new window pops up, asking you to specify a directory for the example design. Select a new directory or keep the default directory.
3. A new project is automatically created in the selected directory and it is opened in a new Vivado window.

4. In the Flow Navigator (left-side pane), click **Run Implementation** and follow the directions.

Example Design Directory Structure

This directory and its subdirectories contain all the source files that are required to create the AXI EMC example design.

Table 5-1 shows the files delivered in the example design directory. It contains the generated example design top files.

Table 5-1: **Example Design Directory**

Name	Description
topdirectory	Top-level project directory; the name is user-defined.
<component_name>/example design	The VHDL example design, test bench and COE files.

Table 5-2 shows the files delivered in the `<project_name>/<project_name>.srcs/sources_1/ip/` directory.

Table 5-2: **Project Files**

Name	Description
Synth/<component_name>.v|vhd	Synthesis wrapper generated by the Vivado tool.
Sim/<component_name>.v|vhd	Simulation wrapper generated by the Vivado tool.
<component_name>.xci	Vivado tools project-specific option file; can be used as an input to the Vivado tools.
<component_name>.vho|veo	VHDL or Verilog instantiation template.
<component_name>_ooc.xdc	Out of Context constraints for IP.
COE Files	These files are intended for the usage of example design, and after open example design is performed all these are copied to new project created.

Table 5-3 shows the files delivered in the `<component_name>/example design` provided with the core.

Table 5-3: **Example Design Files**

Name	Description
<component_name>_exdes.vhd	Example design top file for synthesis.
<component_name>_exdes_tb.vhd	Example design top file for simulation.
<memory_model>.vhd	Memory model to mimic the behavior in simulation.
Exdes.xdc	Constraints for the example design

Test Bench

This chapter contains information about the test bench provided in the Vivado® Design Suite.

Figure 6-1 shows a top-level view of the example design test bench.

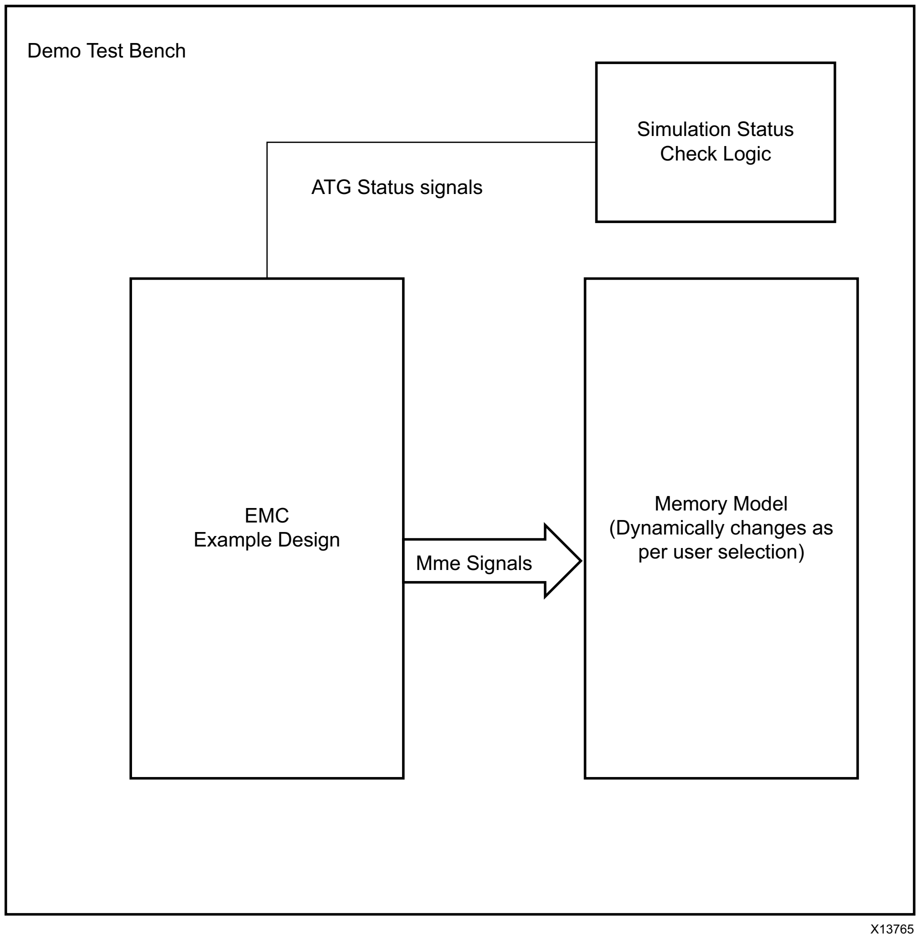

Demo Test Bench
Simulation Status Check Logic
ATG Status signals
EMC Example Design
Mme Signals
Memory Model (Dynamically changes as per user selection)
X13765

Figure 6-1: **AXI EMC Example Design Test Bench**

The demonstration test bench performs the following tasks:

- Generates the clock and reset inputs for the clocking wizard.
- Instantiates the memory model which communicates with EMC. The memory model used depends on your selection in the Vivado Integrated Design Environment (IDE).
- Instantiates IOBUF to take in I, O, and T pins of the memory model, and give a single I/O port to connect with EMC.
- Considers the `done` and `status` pins from ATG and determines the simulation status.

Simulating the Example Design

Using the AXI EMC example design (delivered as part of the AXI EMC), you can quickly simulate and observe the behavior of the AXI EMC.

Setting Up the Simulation

The Xilinx® simulation libraries must be mapped into the simulator. If the libraries are not set for your environment, see the *Vivado Design Suite User Guide: Logic Simulation* (UG900) [Ref 7] for assistance compiling Xilinx simulation models and setting up the simulator environment. To switch simulators, click **Simulation Settings** in the Flow Navigator (left pane). In the Simulation options list, change **Target Simulator**.

Simulation Results

The simulation script compiles the AXI EMC example design and supporting simulation files. It then runs the simulation and checks to ensure that it completed successfully.

If the test passes, then the following message is displayed:

```
Test Completed Successfully
```

If the test fails or does not complete, then the following message is displayed:

```
Test Hanged
```

This is the message displayed when example design simulation fails.

Verification, Compliance, and Interoperability

This appendix provides details about how this IP core was tested for compliance.

Simulation

The AXI External Memory Controller (EMC) core has been tested with Xilinx® Vivado® Design Suite and the Mentor Graphics QuestaSim simulator. For the supported versions of these tools, see the *Xilinx Design Tools: Release Notes Guide (UG631)*[3].

The IP is tested using Xilinx proprietary standard AXI Memory Mapped OVM Verification Components (OVCs).

Hardware Testing

Figure A-1 shows the hardware testing setup for the AXI EMC core.

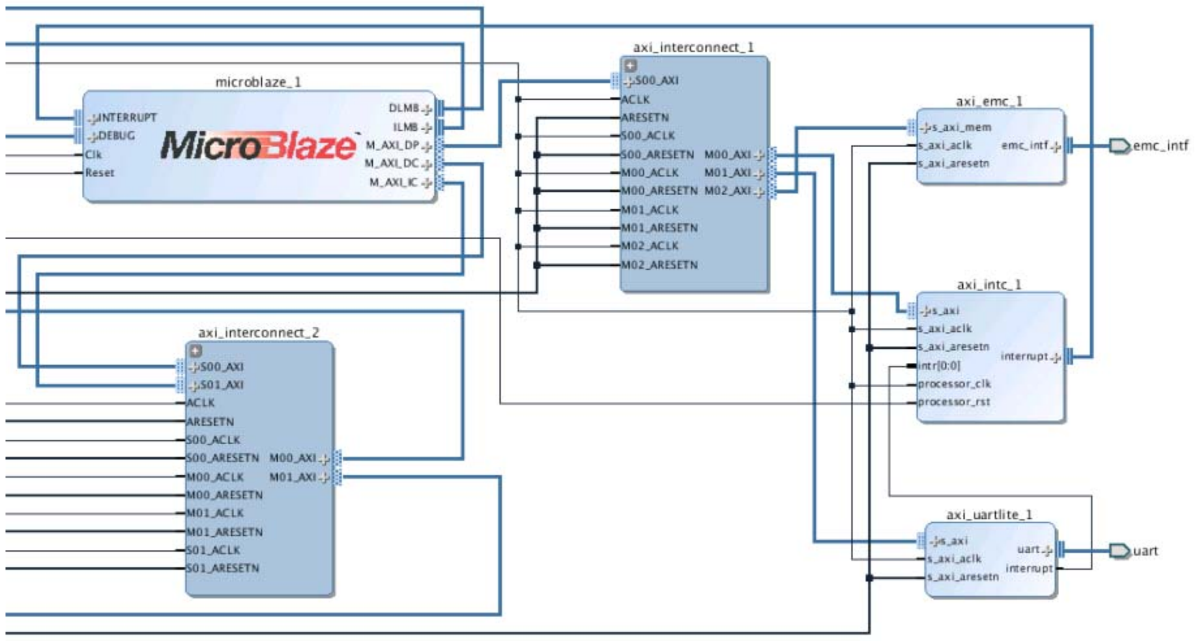
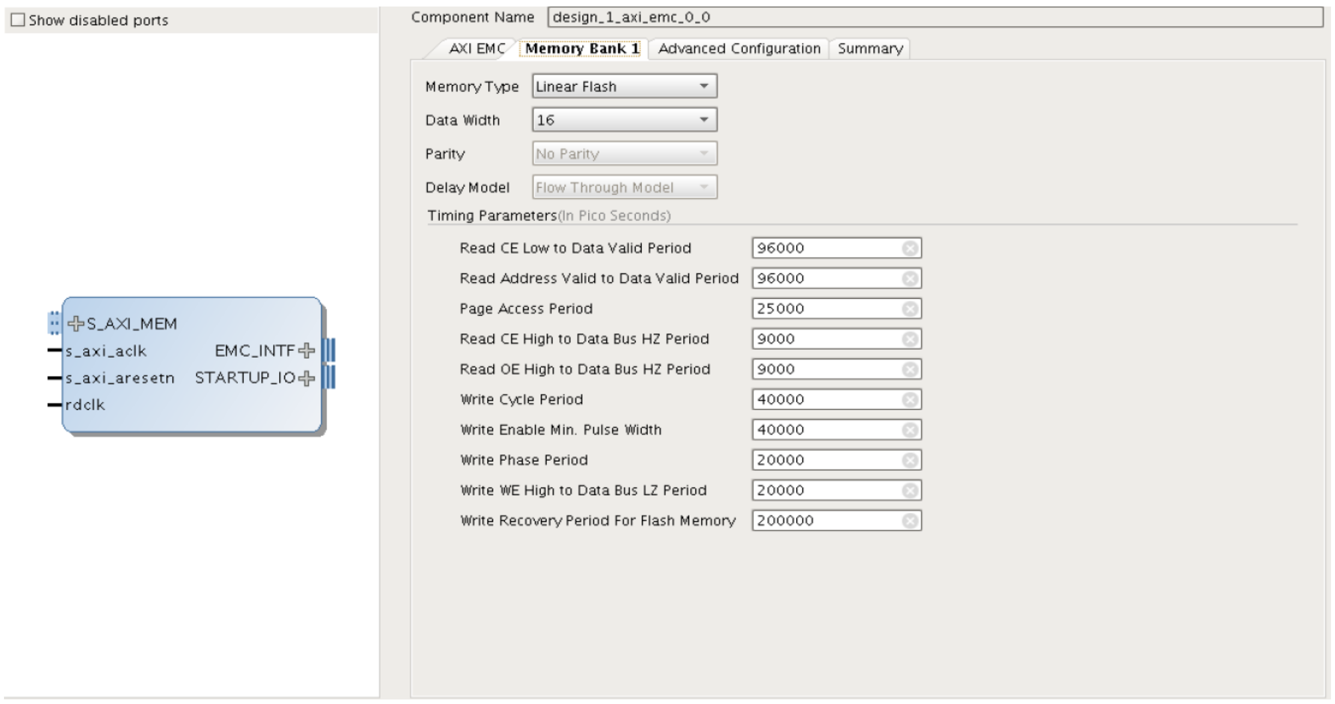


Figure A-1: **AXI EMC Core Hardware Testing**

The AXI EMC core has been hardware validated on a KC705 board using a Kintex®-7 FPGA with –1 speed grade (325T). The setup uses a AXI4 system implementing a MicroBlaze™ processor, AXI4 interconnects, AXI Interrupt Controller, AXI block RAM and UART peripheral, in addition to the AXI EMC core. The AXI EMC core in this setup is connected to Linear Flash Memory and is used to load the device boot code from the flash memory to an internal AXI block RAM memory. AXI EMC core operation in this setup is set to 100 MHz and Flash Memory is operated at a frequency of 50 MHz.

Also, the AXI EMC core has been tested on a VCU108 board having a STARTUP block inside the IP, The following is a snapshot of the timing parameters of the MT28GU01GAAA1EGC-0SIT flash present on the board.



Component Name design_1_axi_emc_0_0

Memory Type	Linear Flash
Data Width	16
Parity	No Parity
Delay Model	Flow Through Model

Timing Parameters(In Pico Seconds)

Read CE Low to Data Valid Period	96000
Read Address Valid to Data Valid Period	96000
Page Access Period	25000
Read CE High to Data Bus HZ Period	9000
Read OE High to Data Bus HZ Period	9000
Write Cycle Period	40000
Write Enable Min. Pulse Width	40000
Write Phase Period	20000
Write WE High to Data Bus LZ Period	20000
Write Recovery Period For Flash Memory	200000

Figure A-2: **AXI EMC Core Tested on a VCU108 Board**

Migrating and Upgrading

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

Migrating to the Vivado Design Suite

For information on migrating to the Vivado Design Suite, see the *Vivado Design Suite Migration Methodology Guide* (UG911) [Ref 8].

Upgrading in the Vivado Design Suite

There are no port or parameter changes.

Debugging

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

Finding Help on Xilinx.com

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

Documentation

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

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

Answer Records

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

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

- Product name
- Tool messages
- Summary of the issue encountered

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

Master Answer Record for the AXI EMC

AR: 54429

Technical Support

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

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

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

Debug Tools

Vivado Design Suite Debug Feature

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

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

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

See the *Vivado Design Suite User Guide: Programming and Debugging* (UG908) [Ref 9].

Interface Debug

AXI4-Lite Interfaces

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

- The `s_axi_aclk` and `aclk` inputs are connected and toggling.
- The interface is not being held in reset, and `s_axi_areset` is an active-Low reset.
- The interface is enabled, and `s_axi_aclk` is active-High (if used).
- The main core clocks are toggling and that the enables are also asserted.
- If the simulation has been run, verify in simulation and/or a Vivado Lab tool capture that the waveform is correct for accessing the AXI4-Lite interface.
- Program the necessary registers for PSRAM and Sync Mode Linear Flash memory as per guidelines given in Register Description in Chapter 2.
- If any non-existent register is accessed either through read or write AXI transaction, the core sends ACK and completes the transaction.

AXI4-Memory Mapped Interfaces

- The core supports all AXI transactions, except FIXED transactions.
- The core responds to read and write transactions in round robin fashion. This ensures that none of the AXI channels are kept in wait mode.
- The core supports single memory access at a given time.
- The AXI read transactions are completed by the core with data and read response.
- In case of any read data error, the particular data response from the core is slave error.
- The AXI write transactions are completed by the core with proper response at the acceptance of the last AXI transaction (after the actual write transactions to the memory is over).

- To access the Async Mode memories (and flash memories), add timing parameters after referring to the data sheets.
- In case of Sync Mode Linear Flash memory access, core support fixed relation is between the AXI clock (100 MHz) and the Sync Mode Linear Flash memory clock (50 MHz).
- Observe the memory interface signals activity when the AXI transactions are accepted by the core.

Additional Resources and Legal Notices

Xilinx Resources

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

References

These documents provide supplemental material useful with this product guide:

1. ARM AMBA AXI and ACE Specification (ARM IHI 0022D)
2. *UltraScale Architecture Libraries Guide* (UG974)
3. *Vivado Design Suite User Guide: Designing with IP* (UG896)
4. *Vivado AXI Reference Guide* (UG1037)
5. *Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator* (UG994)
6. *Vivado Design Suite User Guide: Getting Started* (UG910)
7. *Vivado Design Suite User Guide: Logic Simulation* (UG900)
8. *ISE to Vivado Design Suite Migration Methodology Guide* (UG911)
9. *Vivado Design Suite User Guide: Programming and Debugging* (UG908)
10. *LogiCORE IP AXI Interconnect Product Guide* (PG059)

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
04/05/2017	3.0	Added AXI4 restriction when using AXI EMC with a flash device.
11/18/2015	3.0	Added support for UltraScale+ families.
09/30/2015	3.0	• Added Flash (Memory) Access through STARTUPE Primitive section. • Added note #6 and 7 in I/O Signal Descriptions table. • Added STARTUPE Primitive section. • Updated figures and added timing diagrams in Customizing and Generating the Core section. • Added Flash (Memory) Access through STARTUP Primitive section. • Updated User Parameters table. • Added Constraining the IP section. • Added important note in Example Design.
04/01/2015	3.0	• Updated description in Configuring the Numonyx Flash in Sync Burst Read Mode section. • Updated Fig. 3-2: Synchronous Memory Bank Clocked by FPGA Output with Feedback and added constraints in Clocking Synchronous Memory section. • Updated Software Considerations section. • Added UNISIM important note in Simulation section.
11/19/2014	3.0	• Document updates only for revision change. • Updated Bits[31, 6] description in Table 2-8: PSRAM/Numonyx Flash Configuration Register (PSRAM_FLASH_CONFIG_REG_X). • Updated description in Configuring the Numonyx Flash in Sync Burst Read Mode section.
06/04/2014	3.0	• Updated Note in I/O Signal Descriptions table. • Updated Note in Allowable Memory Combination section. • Updated Fig. 4-4: Summary of AXI EMC in Enable Internal Registers section. • Updated description in Enable Burst Mode for Numonyx Flash. • Added Recommended Note in Write Recovery Period for Flash Memory section. • Added User Parameters in Design Flow Steps chapter.

Date	Version	Revision
04/02/2014	3.0	• Updated IP Facts table. • Added UltraScale support in Maximum Frequencies section. • Updated Table 2-1: Maximum Frequencies. • Added data width description in Memory Controller Unit section. • Disabled parity feature in Byte Parity Logic section. • Updated ports, ID width description, and memory parity notes in Table 2-5 I/O Signal Descriptions. • Added Numonyx Flash description to Clocking Synchronous Memory section. • Updated description in Configuring the Numonyx Flash in Sync Burst Read Mode section. • Updated description in Allowable Memory Combination section. • Updated figures and descriptions in Customizing and Generating the Core chapter. • Updated Constraining the Core chapter. • Updated to Test Hanged in Simulation Results section. • Added Important note in Example Design chapter. • Updated description in Test Bench chapter.
12/18/2013	2.0	Added UltraScale support.
10/02/2013	2.0	Re-release of document for core v2.0, with the following changes made: • Added information related to the core design example, and test bench. • Added Vivado IP integrator support. • Added the Simulation, and Synthesis and Implementation chapters. • Added AXI4-memory mapped interface debug information.
03/20/2013	2.0	• Updated to core v2.0 and Vivado Design Suite-only support. • Updates to modules, and timing diagrams. • Added Appendix A, Verification, Compliance, and Interoperability. • Major revisions to Appendix C, Debugging.
12/18/2012	1.0	Initial Xilinx release as a product guide. Replaces *LogiCORE IP AXI EMC Data Sheet*, DS762.

Please Read: Important Legal Notices