Flash Access IP Core - Lattice Radiant Software

User Guide

FPGA-IPUG-02171-1.1

May 2022

Disclaimers

Lattice makes no warranty, representation, or guarantee regarding the accuracy of information contained in this document or the suitability of its products for any particular purpose. All information herein is provided AS IS, with all faults and associated risk the responsibility entirely of the Buyer. Buyer shall not rely on any data and performance specifications or parameters provided herein. Products sold by Lattice have been subject to limited testing and it is the Buyer's responsibility to independently determine the suitability of any products and to test and verify the same. No Lattice products should be used in conjunction with mission- or safety-critical or any other application in which the failure of Lattice's product could create a situation where personal injury, death, severe property or environmental damage may occur. The information provided in this document is proprietary to Lattice Semiconductor, and Lattice reserves the right to make any changes to the information in this document or to any products at any time without notice.

Contents

Figures

Tables

Acronyms in This Document

A list of acronyms used in this document.

Acronym	Definition
FPGA	Field Programmable Gate Array
IP	Intellectual Property
LMMI	Lattice Memory Mapped Interface

1. Introduction

The Flash Access soft IP enables you to perform write and read access to the internal flash memory of LFMXO5 device. The write and read access is performed through the LMMI interface.

1.1. Features

The Flash Access soft IP has the following features:

- Supports LMMI interface
- Supports initial user data to be programmed into the flash memory
- Supports up to 133 MHz input clock frequency
- This is only supported in LFMXO5 device

1.2. Conventions

1.2.1. Nomenclature

The nomenclature used in this document is based on Verilog HDL.

2. Functional Descriptions

2.1. Overview

The Flash Access soft IP has a LMMI slave interface. This is used to perform the write and read access to the internal flash memory.

Flash access consists of two sub-blocks Flash Access Core and OSCILLATOR. Flash Access Core receives the LMMI slave interface access and performs the access to the internal flash memory.

OSCILLATOR block is required by the Flash Access Core. It utilizes the oscillator clock to control the power up reset to guarantee a valid POR pulse with and without the occurrence of the system-level POR signal.

In LFMXO5 device, only one OSCILLATOR block can be instantiated. If you need to use the OSCILLATOR block for other modules in your design, you should uncheck the *Instantiate internal oscillator* attribute. The user should instantiate an Oscillator Soft IP outside of Flash Access soft IP as described in Connecting an Instantiated Oscillator Soft IP to the Flash Access Soft IP.

When the *Instantiate Internal Oscillator* attribute is checked, Flash Access soft IP instantiates the OSCILLATOR block internally. You have two options for LMMI clock.

- If you opt to provide input to LMMI clock, the *Use HFCLK for LMMI CLK* attribute should be unchecked. The lmmi_clk_i is used directly by the Controller of Flash Access Core while this input is also passed to Oscillator block which output clock connects to flash memory as shown in Figure 2.1.
- The OSCILLATOR block generates the LMMI clock. In this case, the *Use HFCLK for LMMI CLK* attribute should be checked. The clock (HFCLK) generated by the OSCILLATOR is connected to the output port lmmi_clk_o. During this setting, lmmi_clk_o is connected to lmmi_clk_i internally. This output clock can be used for synchronization on the other design to be used with Flash Access IP. You can select the frequency of HFCLK using Frequency of the internal LMMI CLK (MHz) attribute as shown in Figure 2.2.

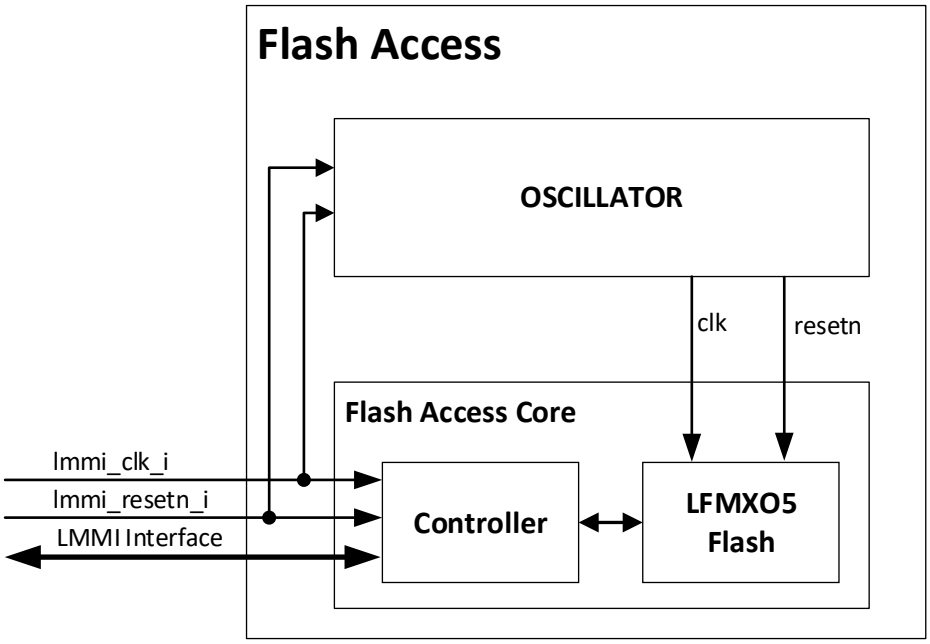

Figure 2.1. Flash Access Block Diagram (*Use HFCLK for LMMI CLK* = Unchecked)

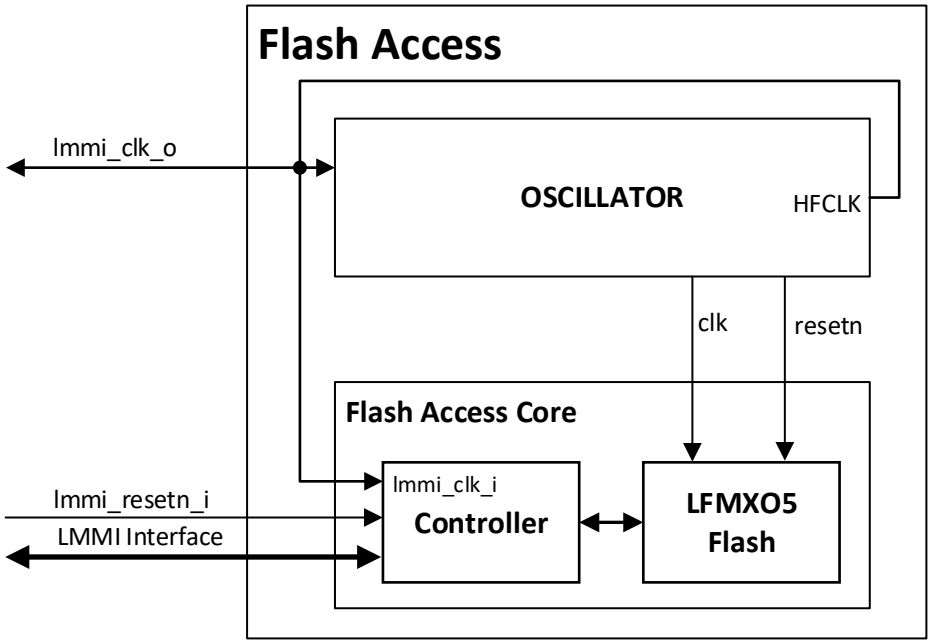

Figure 2.2. Flash Access Block Diagram (*Use HFCLK for LMMI CLK* = Checked)

2.2. Signal Description

Table 2.1 list the ports of the Flash access soft IP.

2.2.1. Flash Access Module Signal

Table 2.1. Flash Access Module Signal Description

Name	Type	Width	Description
clk_i	In	1	Input clock. Available when *Instantiate Internal Oscillator* = Unchecked
lmmi_clk_i	In	1	Clock signal Input Available when *Instantiate Internal Oscillator* = Checked and *Use HFCLK for LMMI_CLK* = Unchecked
lmmi_clk_o	Out	1	Clock signal Output Available when *Use HFCLK for LMMI CLK*= Checked
lmmi_resetn_i	In	1	Active low reset signal
lmmi_request_i	In	1	Start transaction This signal is asserted to initiate a function to be performed by the flash access IP. This signal also controls the duration of the write and read access to the flash memory
lmmi_offset_i	In	8	Address bus This bus is used to specify the type of function to be performed by the flash access IP. Refer to Table 2.9 for more details.
lmmi_wr_rdn_i	In	1	When high, indicates write transaction. When low, indicates read transaction.
lmmi_wdata_i	In	1	Serial interface for transmitting data
lmmi_rdata_o	Out	1	Serial interface for receiving data
lmmi_rdata_valid_o	Out	1	This signal indicates that lmmi_rdata_o contains valid data

Name	Type	Width	Description
lmmi_ready_o	Out	1	Ready to start a new lmmi transaction
rstn_i	In	1	Active low reset input. Available when *Instantiate Internal Oscillator* = Unchecked.

Note: Refer to LMMI User Guide for more information.

2.3. Attribute Summary

The configurable attributes of the Flash Access IP are shown in Table 2.2 and are described in Table 2.3.

The attributes can be configured through the Lattice Radiant™ software.

Table 2.2. Attributes Table

Attribute	Selectable Values	Default	Dependency on Other Attributes
CFG0/1/2 and UFM0/1/2 Settings Tab			
CFG0 and UFM0 Settings Group			
Use CFG0 with size (blocks)	0, 11, 15	11	-
CFG0 Initial Value Format	ALL 1, ALL 0, NONE, HEXFILE, BINFILE	ALL 1	Use CFG0 with size (blocks) > 0
CFG0 Initial Value File Path	—	—	CFG0 Initial Value Format = HEXFILE or BINFILE
Use UFM0 with size (blocks)	0, 4	0	Use CFG0 with size (blocks) < 15
UFM0 Initial Value Format	ALL 1, ALL 0, NONE, HEXFILE, BINFILE	—	Use UFM0 with size (blocks) > 0
UFM0 Initial Value File Path	—	—	UFM0 Initial Value Format = HEXFILE or BINFILE
CFG1 and UFM1 Settings Group			
Use CFG1 with size (blocks)	0, 11, 15	0	—
CFG1 Initial Value Format	ALL 1, ALL 0, NONE, HEXFILE, BINFILE	—	Use CFG1 with size (blocks) > 0
CFG1 Initial Value File Path	—	—	CFG1 Initial Value Format = HEXFILE or BINFILE
Use UFM1 with size (blocks)	0, 4	0	Use CFG1 with size (blocks) < 15
UFM1 Initial Value Format	ALL 1, ALL 0, NONE, HEXFILE, BINFILE	—	Use UFM1 with size (blocks) > 0
UFM1 Initial Value File Path	—	—	UFM1 Initial Value Format = HEXFILE or BINFILE
CFG2 and UFM2 Settings Group			
Use CFG2 with size (blocks)	0, 11, 15	0	-
CFG2 Initial Value Format	ALL 1, ALL 0, NONE, HEXFILE, BINFILE	—	Use CFG2 with size (blocks) > 0
CFG2 Initial Value File Path	—	—	CFG2 Initial Value Format = HEXFILE or BINFILE
Use UFM2 with size (blocks)	0, 4	0	Use CFG2 with size (blocks) < 15
UFM2 Initial Value Format	ALL 1, ALL 0, NONE, HEXFILE, BINFILE	—	Use UFM2 with size (blocks) > 0
UFM2 Initial Value File Path	—	—	UFM2 Initial Value Format = HEXFILE or BINFILE
USERDATA0 to USERDATA8 Settings Tab			
Use USERDATA0 to USERDATA8	Checked, Unchecked	Unchecked	-
USERDATA0 Settings Group			
Use USERDATA0 with size (blocks)	1–18	—	Use USERDATA0 to USERDATA8 = Checked
USERDATA0 Initial Value Format	ALL 1, ALL 0, NONE, HEXFILE, BINFILE	—	Use USERDATA0 to USERDATA8 = Checked *Refer to Table 2.3 for the rules of USERDATA*

Attribute	Selectable Values	Default	Dependency on Other Attributes
			block size
USERDATA0 Initial Value File Path	—	—	USERDATA0 Initial Value Format = HEXFILE or BINFILE
USERDATA1 Settings Group			
Use USERDATA1 with size (blocks)	1–18	—	Use USERDATA0 to USERDATA8 = Checked *Refer to Table 2.3 for the rules of USERDATA block size*
USERDATA1 Initial Value Format	ALL 1, ALL 0, NONE, HEXFILE, BINFILE	—	Use USERDATA1 with size >= 1
USERDATA1 Initial Value File Path	—	—	USERDATA1 Initial Value Format = HEXFILE or BINFILE
USERDATA2 Settings Group			
Use USERDATA2 with size (blocks)	1–18	—	Use USERDATA0 to USERDATA8 = Checked *Refer to Table 2.3 for the rules of USERDATA block size*
USERDATA2 Initial Value Format	ALL 1, ALL 0, NONE, HEXFILE, BINFILE	—	Use USERDATA2 with size >= 1
USERDATA2 Initial Value File Path	—	—	USERDATA2 Initial Value Format = HEXFILE or BINFILE
USERDATA3 Settings Group			
Use USERDATA3 with size (blocks)	1–18	—	Use USERDATA0 to USERDATA8 = Checked *Refer to Table 2.3 for the rules of USERDATA block size*
USERDATA3 Initial Value Format	ALL 1, ALL 0, NONE, HEXFILE, BINFILE	—	Use USERDATA3 with size >= 1
USERDATA3 Initial Value File Path	—	—	USERDATA3 Initial Value Format = HEXFILE or BINFILE
USERDATA4 Settings Group			
Use USERDATA4 with size (blocks)	1–18	—	Use USERDATA0 to USERDATA8 = Checked *Refer to Table 2.3 for the rules of USERDATA block size*
USERDATA4 Initial Value Format	ALL 1, ALL 0, NONE, HEXFILE, BINFILE	—	Use USERDATA4 with size >= 1
USERDATA4 Initial Value File Path	—	—	USERDATA4 Initial Value Format = HEXFILE or BINFILE
USERDATA5 Settings Group			
Use USERDATA5 with size (blocks)	1–18	—	Use USERDATA0 to USERDATA8 = Checked *Refer to Table 2.3 for the rules of USERDATA block size*
USERDATA5 Initial Value Format	ALL 1, ALL 0, NONE, HEXFILE, BINFILE	—	Use USERDATA5 with size >= 1
USERDATA5 Initial Value File Path	—	—	USERDATA5 Initial Value Format = HEXFILE or BINFILE
USERDATA6 Settings Group			
Use USERDATA6 with size (blocks)	1–18	—	Use USERDATA0 to USERDATA8 = Checked *Refer to Table 2.3 for the rules of USERDATA block size*
USERDATA6 Initial Value Format	ALL 1, ALL 0, NONE, HEXFILE, BINFILE	—	Use USERDATA6 with size >= 1
USERDATA6 Initial Value File Path	—	—	USERDATA6 Initial Value Format = HEXFILE or BINFILE

Attribute	Selectable Values	Default	Dependency on Other Attributes
USERDATA7 Settings Group			
Use USERDATA7 with size (blocks)	1–18	—	Use USERDATA0 to USERDATA8 = Checked *Refer to Table 2.3 for the rules of USERDATA block size*
USERDATA7 Initial Value Format	ALL 1, ALL 0, NONE, HEXFILE, BINFILE	—	Use USERDATA7 with size >= 1
USERDATA7 Initial Value File Path	—	—	USERDATA7 Initial Value Format = HEXFILE or BINFILE
USERDATA8 Settings Group			
Use USERDATA8 with size (blocks)	1–18	—	Use USERDATA0 to USERDATA8 = Checked *Refer to Table 2.3 for the rules of USERDATA block size*
USERDATA8 Initial Value Format	ALL 1, ALL 0, NONE, HEXFILE, BINFILE	—	Use USERDATA8 with size >= 1
USERDATA8 Initial Value File Path	—	—	USERDATA8 Initial Value Format = HEXFILE or BINFILE
Oscillator Settings Group			
Instantiate Internal Oscillator	Checked, Unchecked	Checked	—
Use HFCLK for LMMI CLK	Checked, Unchecked	Unchecked	*Instantiate Internal Oscillator* = Checked
Frequency of the internal LMMI CLK (MHz)	112.5, 90.0, 75.0, 64.2857, 56.25, 50.0, 45.0, 40.909, 30.0	90.0	*Use HFCLK for LMMI CLK* = Checked

Table 2.3. Attributes Description

Attribute	Description
Use CFG<0/1/2> with size (blocks)	Allows you to select if CFG<0/1/2> partition is used (11 or 15) or not (0); You have two options for the partition size: 11 blocks or 15 blocks. If partition size is set to 0, it should not be accessed by the user.
Use UFM<0/1/2> with size (blocks)	Allows you to select if UFM<0/1/2> partition is used (4) or not (0). • If the partition size of the corresponding CFG is set to 15, this parameter is automatically set to 0. • If the partition size is set to 0, it should not be accessed by the user.
Use USERDATA0 to USERDATA8	Allows you to select if USERDATA0 to USERDATA8 partition is used (Checked) or not (Unchecked). • Unchecked partition should not be accessed by the user. • At least USERDATA0 partition should be used if this attribute is checked.
Use USERDATA<0..8> with size (blocks)	Allows you to select the size of each USERDATA<0..8> partition in block with the following rules: [1] Selecting '0' option means that the partition is not used. *'0' option is not available for USERDATA0, USERDATA0 should be used if Use USERDATA0 to USERDATA8 = Checked* [2] If a partition is set to '0' the succeeding partitions cannot be used. *For example, if USERDATA3 size is set to '0' USERDATA4 to USERDATA8 will automatically be set to '0'.* [3] Total number of blocks should not exceed 18 blocks Available options change depending on the selected value of other USERDATAx partitions.

Attribute	Description
CFG<0/1/2> / UFM<0/1/2> / USERDATA<0..8> Initial Value Format	Allows you to select how to initialize the flash memory partition Options: ALL 1 : each bit in the whole partition is set to '1' ALL 0 : each bit in the whole partition is set to '0' NONE : no initial data is set; Lattice factory value is used HEXFILE : You specify a custom memory file with HEX format BINFILE : You specify a custom memory file with BIN format
CFG<0/1/2> / UFM<0/1/2> / USERDATA<0..8> Initial Value File Path	Allows you to browse to the memory file to select the custom memory file.
Instantiate Internal Oscillator	Allows you to select instantiate Oscillator Soft IP.
Use HFCLK for LMMI CLK	Allows you to use HFCLK output of the internal oscillator as LMMI CLK. lmmi_clk_o is available when this attribute is selected. This clock output should be used to sync the LMMI transactions.
Frequency of the internal LMMI CLK (MHz)	Allows you to select the frequency of the HFCLK when *Use HFCLK for LMMI CLK*= Checked.

2.4. Connecting an Instantiated Oscillator Soft IP to the Flash Access Soft IP

In the LFMXO5 device, only one OSCILLATOR block can be instantiated. For some cases where Flash Access is used as a submodule, Oscillator block may be already used by other submodules. In this case, uncheck the *Instantiate Internal Oscillator* attribute.

Unchecking this, however, requires manual instantiation and connection of Oscillator Soft IP to Flash Access IP to enable Radiant flow and simulation.

Figure 2.3 shows a sample connection of an Oscillator Soft IP to the ports of the Flash Access IP.

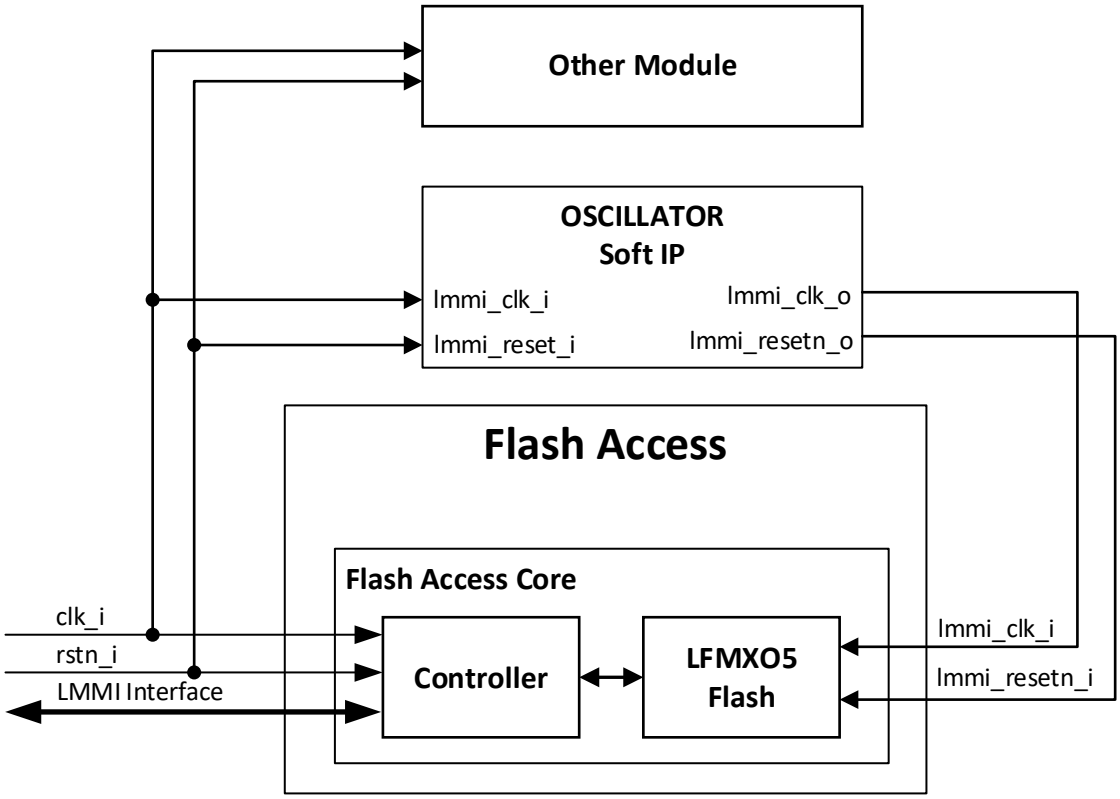

Figure 2.3. Sample Connection of Oscillator Soft IP to Flash Access Soft IP

2.5. Block Partitioning

Table 2.4 shows the whole memory map of the LFMXO5 flash memory. You can modify the partition sizes by changing the attributes prior to IP generation.

You have five options for configuring each of the three CFGx/UFMx partition sizes as shown in Table 2.5.

The rules in configuring the partition sizes of USERDATAx is described in Table 2.3.

Table 2.6 and Table 2.7 shows some sample settings for different flash partition block sizes.

Table 2.4. Flash Memory Map

Block Start Address (24-Bit Byte address)	Block End Address (24-Bit Byte Address)	Block Size	Sector Size	Contents
01 (0x01_0000)	11 (0x0B_FFFF)	11	176	CFG0
12 (0x0C_0000)	15 (0x0F_FFFF)	4	64	UFM0
16 (0x10_0000)	26 (0x1A_FFFF)	11	176	CFG1
27 (0x1B_0000)	30 (0x1E_FFFF)	4	64	UFM1
31 (0x1F_0000)	41 (0x29_FFFF)	11	176	CFG2
42 (0x2A_0000)	45 (0x2D_FFFF)	4	64	UFM2
46 (0x2E_0000)	63 (0x3F_FFFF)	Total ≤ 18	Total ≤ 288	USERDATA0
				USERDATA1
				USERDATA2
				USERDATA3
				USERDATA4
				USERDATA5
				USERDATA6
				USERDATA7
				USERDATA8

Note: One page has 256 bytes of data.

Table 2.5. Different Combinations of CFGx/UFMx Block Size

Number	CFGx Block Size	UFMx Block Size	Use Partition Setting	Description
1	15	0	Use CFGx with size = 15	Only one partition CFGx having 15 block size.
			Use UFMx with size = 0	
2	11	0	Use CFGx with size = 11	Only one partition CFGx having 11 block size; you should not access the partition for UFMx.
			Use UFMx with size = 0	
3	0	4	Use CFGx with size = 0	Only one partition UFMx having 4 block size; you should not access the partition for CFGx.
			Use UFMx with size = 4	
4	11	4	Use CFGx with size = 11	Two partitions CFGx having 11 block size and UFMx having 4 block size.
			Use UFMx with size = 4	
5	0	0	Use CFGx with size = 0	Neither CFGx or UFMx should be accessed. This combination is possible as long as either one CFGx and UFMx is used. For example, other CFGx and UFMx can be set to 0 as long as CFG0 has size greater than 0.
			Use UFMx with size = 0	

Table 2.6. Sample Settings 1

CFGx Block Size	UFMx Block Size	Use Partition Setting	Description
15	N/A	Use CFG0 with size = 15	CFG0
N/A	0	Use UFM0 with size = 0	User should not access
11	N/A	Use CFG1 with size = 11	CFG1
N/A	0	Use UFM1 with size = 0	User should not access
0	N/A	Use CFG2 with size = 0	User should not access
N/A	4	Use UFM2 with size = 4	UFM2
USERDATAx Block Size		**Use Partition Setting**	**Description**
10		USERDATA0 Size = 10	USERDATA0
8		USERDATA1 Size = 8	USERDATA1
N/A		Automatically set to '0'	User should not access
N/A		Automatically set to '0'	User should not access
N/A		Automatically set to '0'	User should not access
N/A		Automatically set to '0'	User should not access
N/A		Automatically set to '0'	User should not access
N/A		Automatically set to '0'	User should not access
N/A		Automatically set to '0'	User should not access

Table 2.7. Sample Settings 2

CFGx Block Size	UFMx Block Size	Use Partition Setting	Description
11	N/A	Use CFG0 with size = 11	CFG0
N/A	4	Use UFM0 with size = 4	UFM0
0	N/A	Use CFG1 with size = 0	User should not access
N/A	0	Use UFM1 with size = 0	User should not access
0	N/A	Use CFG2 with size = 0	User should not access
N/A	0	Use UFM2 with size = 0	User should not access
USERDATAx Block Size		**Use Partition Setting**	**Description**
8		USERDATA0 Size = 8	USERDATA0
4		USERDATA1 Size = 4	USERDATA1
2		USERDATA2 Size = 2	USERDATA2
1		USERDATA3 Size = 1	USERDATA3
2		USERDATA4 Size = 2	USERDATA4
N/A		Automatically set to '0'	User should not access
N/A		Automatically set to '0'	User should not access
N/A		Automatically set to '0'	User should not access
N/A		Automatically set to '0'	User should not access

2.6. Setting Initial Data in Flash Memory

This section provides information on how to set the initial data for the flash device using attributes.

There are five options available to you on how to initialize each flash memory partition. Table 2.8 describes these options.

Table 2.8. Attributes Description

Partition Initial Value Format	Description
ALL 1	An init data file is generated automatically containing each bit of the whole memory partition set to '1'
ALL 0	An init data file is generated automatically containing each bit of the whole memory partition set to '0'
NONE	No init data is file is generated
BINFILE	The init data file of the memory partition is provided by the user. The init data file is in binary memory file format.
HEXFILE	The init data file of the memory partition is provided by the user. The init data file is in hex memory file format.

2.6.1. Binary File

The binary file is a text file of 0s and 1s. The number of rows is equal to the block size of the partition multiplied by 128 (block size × 128). Each row has 4,096 bits.

2.6.2. Hex File

The hex file is a text file of hexadecimal characters. The number of rows is equal to the block size of the partition multiplied by 128 (block size × 8). Each row has 1,024 hex characters.

2.7. Flash functions

This section describes the functions that the Flash Access IP supports.

Table 2.9 lists the functions supported by Flash Access IP and the succeeding subsections provide more details.

Table 2.9. Functions supported by Flash Access IP

Function Group	Function	LMMI Offset	Description
Check Status	Check Status	0x00	Allows you to check the if the Flash access is busy or not
Erase	Sector Erase (4KB)	0x10	Allows you to set 4 KB sector to erased state
	Block Erase (32KB)	0x11	Allows you to set 32 KB block to erased state
	Block Erase (64KB)	0x12	Allows you to set 64 KB block to erased state
	On-chip Flash Erase	0x13	Allows you to set the whole flash memory to erased state
Write	Write access	0x20	Allows you to perform the write access to flash memory Memory location should be in erased state prior to performing write access
Read	Read access – Low frequency	0x30	Allows you to perform the read access to flash memory Maximum LMMI clock frequency for this function is 50 MHz
	Read access – High frequency	0x31	Allows you to perform the read access to flash memory Maximum LMMI clock frequency for this function is 133 MHz

Note: All bits are set to 1 during erased state.

2.7.1. Check Status Sequence

This function is used to check if Flash access IP is busy or not. Users should use this function to ensure that the flash access IP is not busy before performing other functions.

Use the following steps to Check Status:

1. Wait for lmmi_ready_o = '1'.
2. Set lmmi_offset_i = '0x00', lmmi_wdata_i = '1', lmmi_request_i = '1' and lmmi_wr_rdn_i = '1'.
3. Set lmmi_wr_rdn_i = '0'.
4. Wait for lmmi_rdata_valid_o = '1'.
5. Wait for the 8th bit of lmmi_rdata_o, this corresponds to the BUSY flag.
6. Set lmmi_request_i = '0' to complete the function.
7. You can opt to hold the lmmi_request_i = '1', lmmi_rdata_o continuously outputs data until lmmi_request_i = '0' is set. BUSY flag can be observed every eight bit.

Step 7 can be performed if you intend to wait until the Flash access IP is not busy before performing other operation.

Figure 2.4 shows a sample waveform for performing the Check Status sequence.

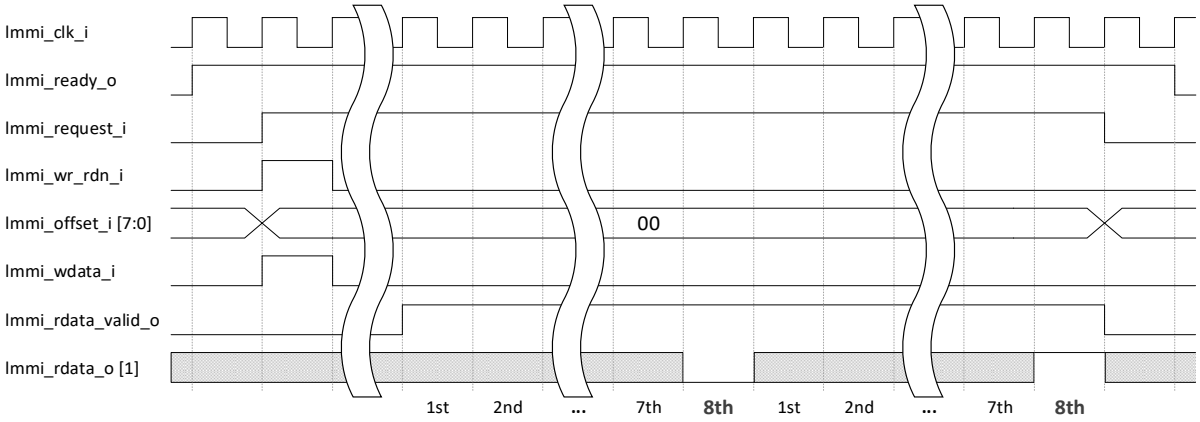

Figure 2.4. Sample Waveform for Check Status Sequence

2.7.2. Erase Sequence

This function is used to set all the memory within the specified sector/block to erased state. User should use this function to ensure that the memory location is in erased state prior to performing write access. All bits are set to 1 during erased state.

You can opt to erase a 4 KB sector, a 32 KB block or a 64 KB block by selecting the lmmi_offset_i that is sent in step 2 below. Refer to Table 2.9 for the list of lmmi_offset_i.

Use the following steps to perform an Erase function:

1. Wait for lmmi_ready_o = '1'.
2. Set lmmi_offset_i based on Table 2.9, lmmi_request_i = '1' and lmmi_wr_rdn_i = '1'.
3. Send the 24-bit address of the sector/block that is erased thru lmmi_wdata_i. The address is sent through the lmmi_wdata_i with the MSB of the address sent first (A23 – A0).
4. Set lmmi_request_i = '0' to complete the function.

The BUSY flag is set to '1' after the last step indicating that the Flash Access IP is performing the erase function on the memory sector/block. The BUSY flag is automatically set to '0' after the erase function is completed. The Flash Access IP cannot perform other functions while it is busy except for Check Status Sequence.

You must ensure that the flash access IP is not busy before performing other functions.

Figure 2.5 shows a sample waveform for performing a Sector Erase Sequence.

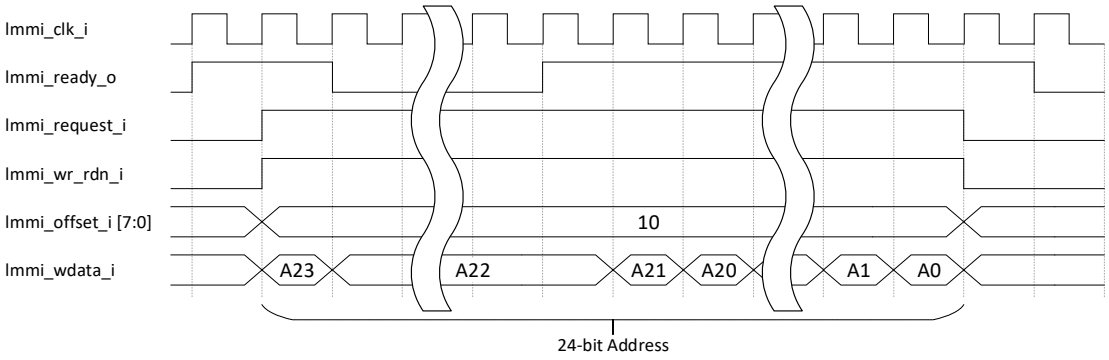

Figure 2.5. Sample Waveform for Sector Erase Sequence

2.7.3. On-chip Flash Erase Sequence

This function is used to set the whole on-chip flash to erased state.

Use the following steps to perform On-chip Flash Erase function:

1. Wait for lmmi_ready_o = '1'.
2. Set lmmi_offset_i = '0x13', lmmi_wdata_i = '1', lmmi_request_i = '1' and lmmi_wr_rdn_i = '1'.
3. Set lmmi_request_i = '0' to complete the function.

The BUSY flag is set to '1' after the last step indicating that the Flash Access IP is performing the erase function on the flash memory. The BUSY flag is automatically set to '0' once the erase function is completed. Flash Access IP cannot perform other functions while BUSY flag = '1' except for Check Status Sequence.

You must ensure that the flash access IP is not busy before performing other functions.

Figure 2.6 shows a sample waveform for performing On-chip Flash Erase Function.

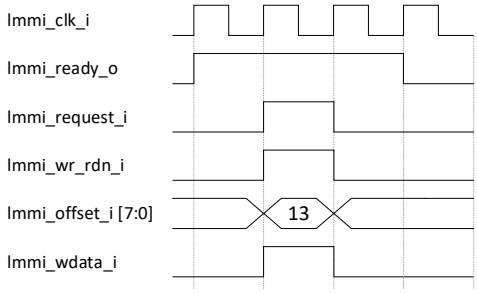

Figure 2.6. Sample Waveform for On-chip Erase Sequence

2.7.4. Write Access Sequence

This function is used to perform write access to the flash memory. Each write access can perform one byte to 256 bytes of a memory page by controlling the lmmi_request_i signal. The memory location to be written should be in erased state prior to performing write access.

Use the following steps to perform Write Access function:

1. Wait for lmmi_ready_o = '1'.
2. Set lmmi_offset_i = '0x20', lmmi_request_i = '1' and lmmi_wr_rdn_i = '1'.
3. Send the 24-bit write address using 24 transactions thru lmmi_wdata_i. The address is sent thru the lmmi_wdata_i with the MSB of the address sent first (A23 – A0).
4. Send the bytes of data to be written thru lmmi_wdata_i. The data to be written is sent through lmmi_wdata_i with the MSB of each byte sent first.

5. Set lmmi_request_i = ‘0’ once all the bytes of data to be written has been sent. If lmmi_request_i = ‘0’ is set in the middle of sending a byte (only first part of the byte has been sent), the whole write access function is not performed.

The BUSY flag is set to ‘1’ after the last step indicating that the Flash Access IP is performing the write access function on the flash memory. The BUSY flag is automatically set to ‘0’ once the write access function is completed. Flash Access IP cannot perform other functions while BUSY flag = ‘1’ except for Check Status Sequence.

You must ensure that the flash access IP is not busy before performing other functions.

The write address sent in Step 3 is used as a starting point of the write access. For every 1 byte sent in Step 4, the write address increments. If the page border is reached, the write address wraps around the start of the page as shown in Figure 2.7. You can send 1 up to 256 bytes of data to be written. However, you must ensure that the number of bytes sent are aligned to 256 bytes or a page address to avoid wrap-around of data.

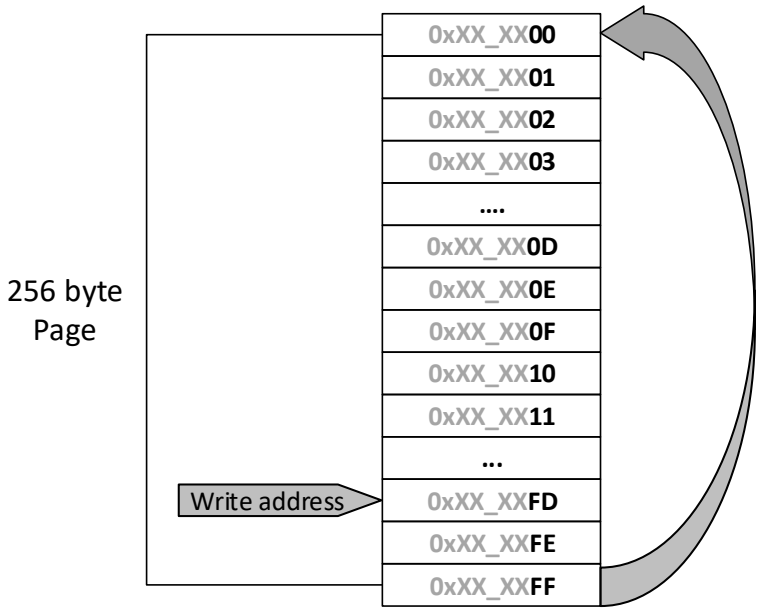

Figure 2.7. Write Data Address Pointer Behavior

Figure 2.8 shows a sample waveform for performing the Write Access function.

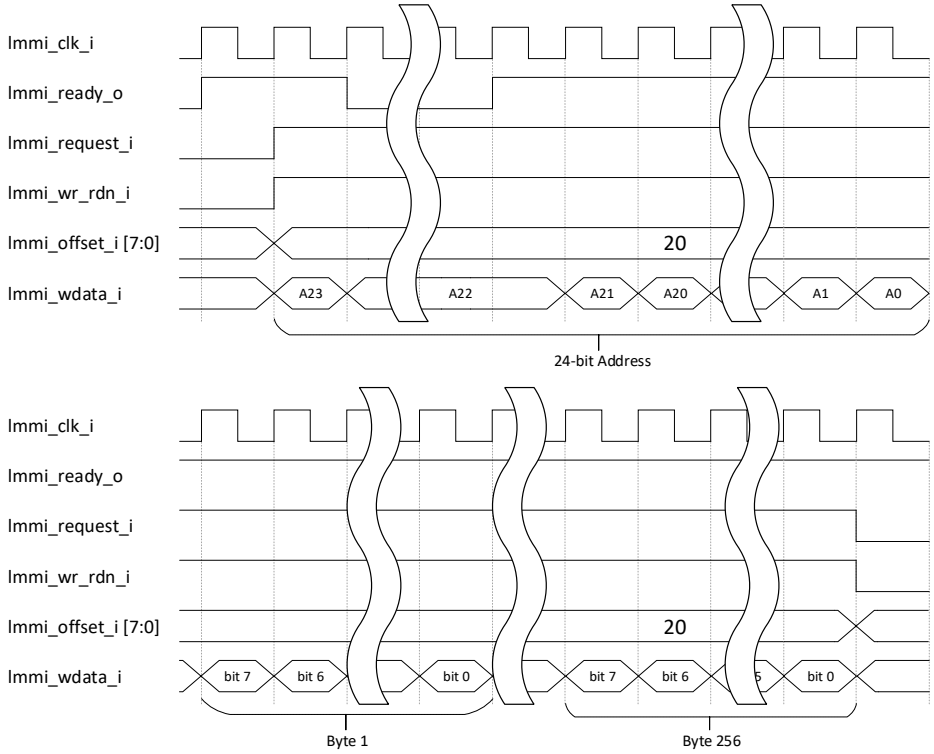

Figure 2.8. Sample Waveform for 256-byte Write Access Sequence

2.7.5. Read Access Sequence

This function is used to perform read access to the flash memory. User can read one or more data bytes by controlling the lmmi_request_i signal.

Use the following steps to perform Read Access -Serial Address function:

1. Wait for lmmi_ready_o = '1'.
2. Set lmmi_offset_i based on Table 2.9, lmmi_request_i = '1' and lmmi_wr_rdn_i = '1'.
3. Send the 24-bit read address using 24 transactions through lmmi_wdata_i. The address is sent thru the lmmi_wdata_i with the MSB of the address sent first (A23 – A0).
4. Set lmmi_wr_rdn_i = '0' to enable lmmi_rdata_o.
5. Wait for lmmi_rdata_valid_o = '1'.
6. Read data is available in lmmi_rdata_o with the MSB of each byte sent first.
7. Set lmmi_request_i = '0' once all the bytes of data to be read has been received.

The read address sent in Step 3 is used as a starting point of the read access. For every 1 byte received in Step 8, the read address increments. There is no limit on the number of bytes to be read in one read access function.

Figure 2.9 shows a sample waveform for performing Read Access-Low frequency function.

If you want to use frequency higher than 50 MHz, you can perform the same steps. You can use Read access – high frequency function and use change the lmmi_offset_i to be set in step 2.

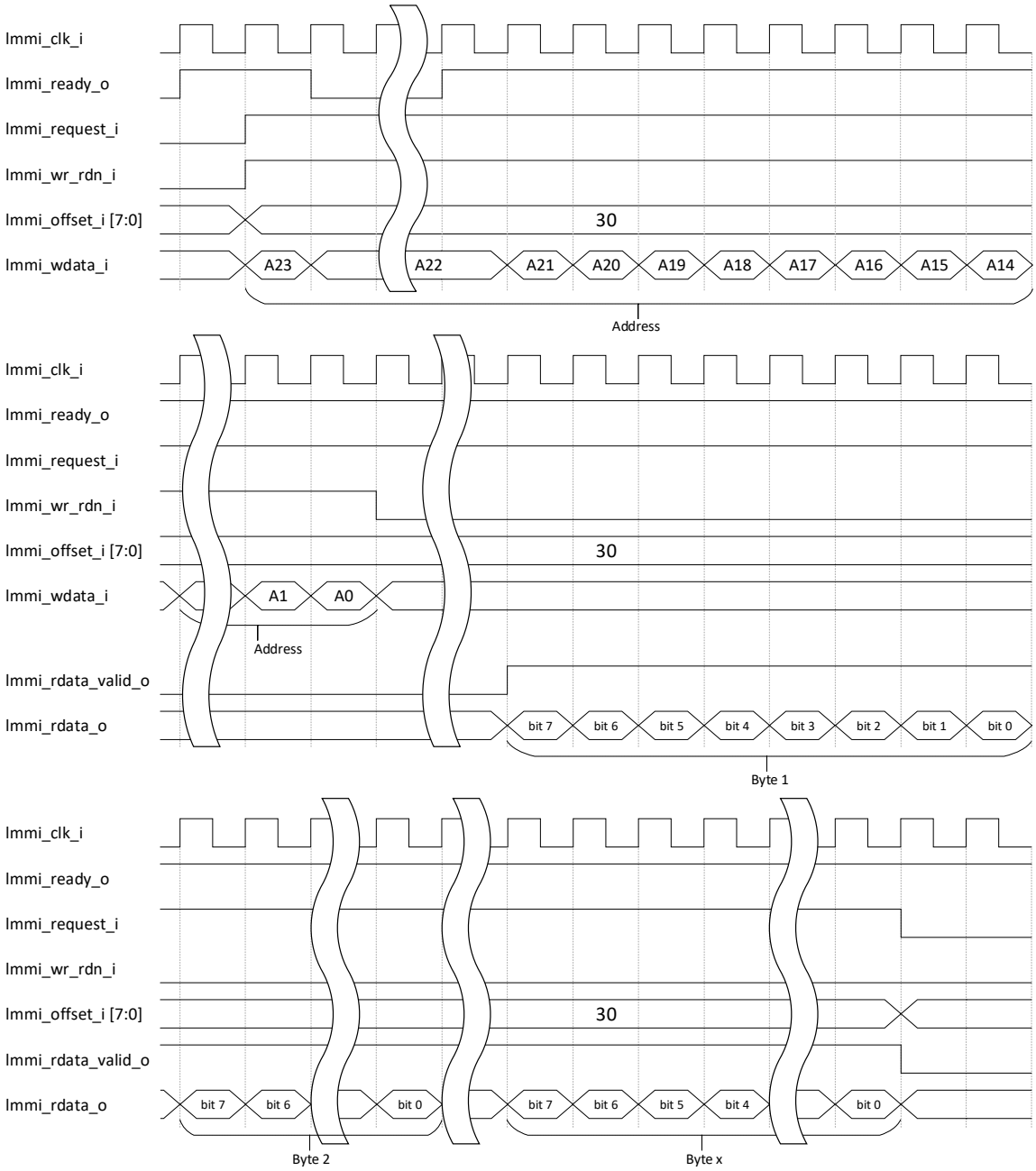

Figure 2.9. Sample Waveform for Read Access – Low frequency Sequence

3. Flash Access IP Generation and Simulation

3.1. IP Generation on Radiant Software

This section provides information on how to generate the Flash access soft IP module using the Lattice Radiant™ software.

To generate the Flash access soft IP module:

1. Double-click Flash Access under Architecture_Modules This opens the Module/IP Block Wizard.
2. Fill out the following information. Click **Next** as shown in Figure 3.1.

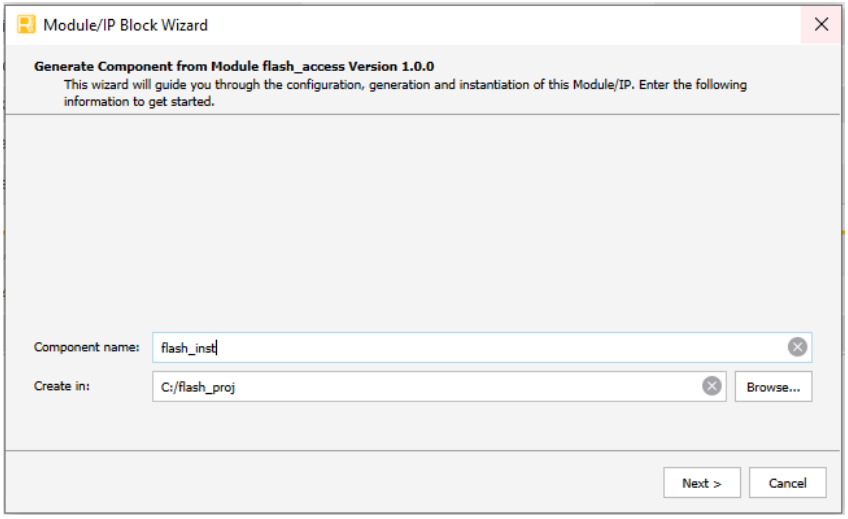

Figure 3.1. Example: Generating Flash Access Using Module/IP Block Wizard

3. Choose the partition settings and click **Next.** Figure 3.2, Figure 3.3, and Figure 3.4 show the tabs of the attributes.

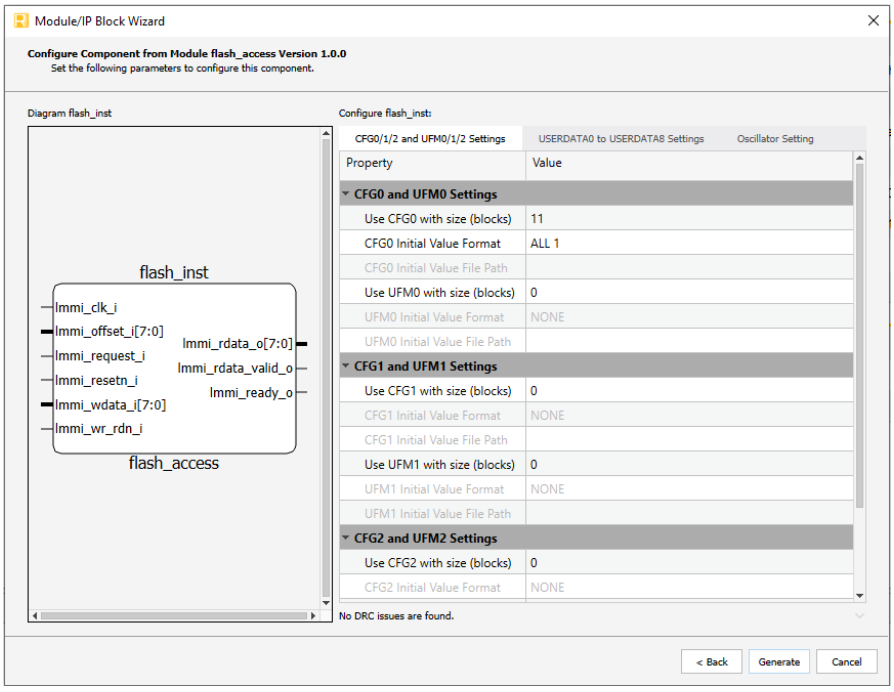

Figure 3.2. Example: Generating Flash Access Using Module/IP Block Wizard

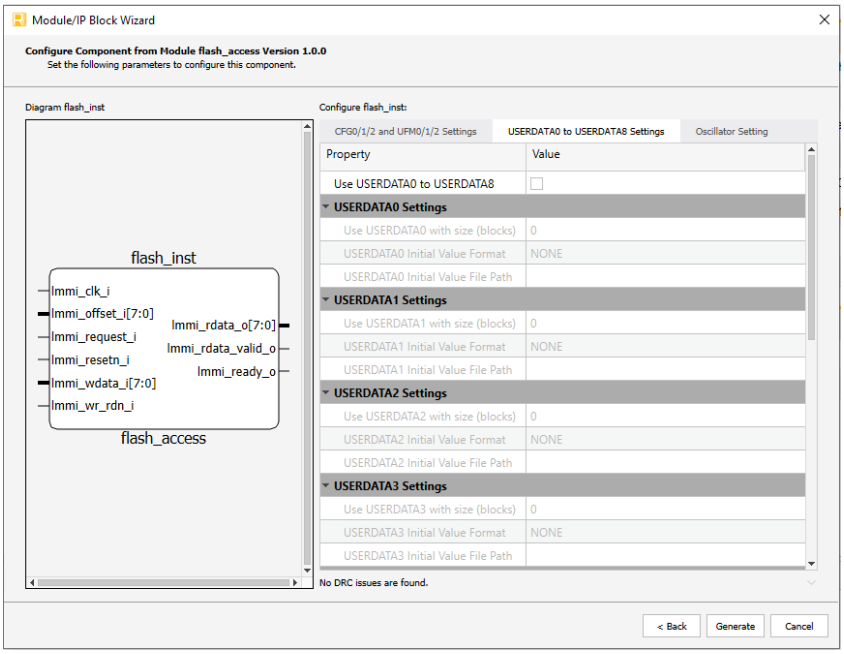

Figure 3.3. Example: Generating Flash Access Using Module/IP Block Wizard

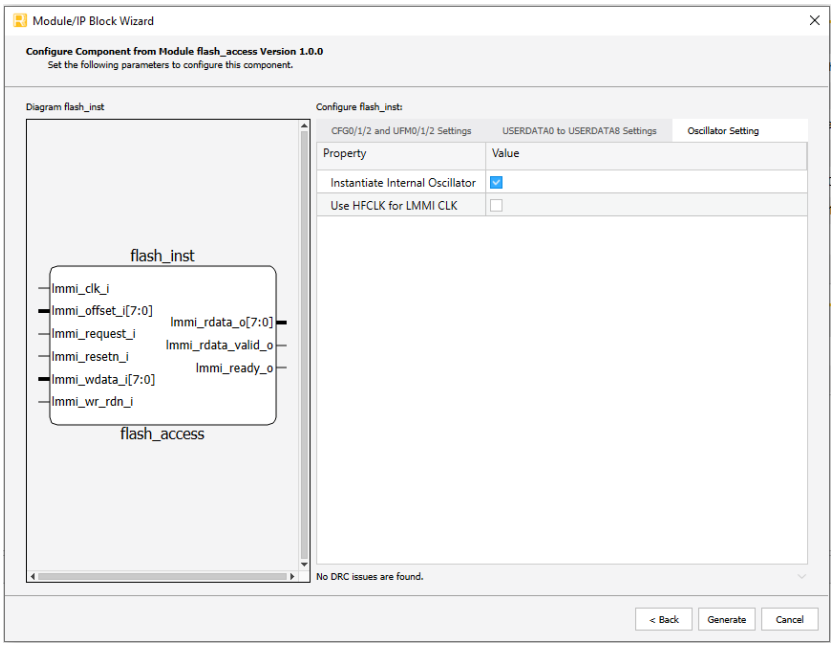

Figure 3.4. Example: Generating Flash Access Using Module/IP Block Wizard

4. When all options are set, click **Generate.**
5. Click **Finish**.

Once this module is in the Lattice Radiant software project, it can be instantiated in other modules within the project. You can view the files and instance/s added in the **File List** tab.

3.2. Running Functional Simulation

3.2.1. Running Default Test Sequence

1. Click the  button located on the **Toolbar** to initiate the **Simulation Wizard** shown in Figure 3.5**.**

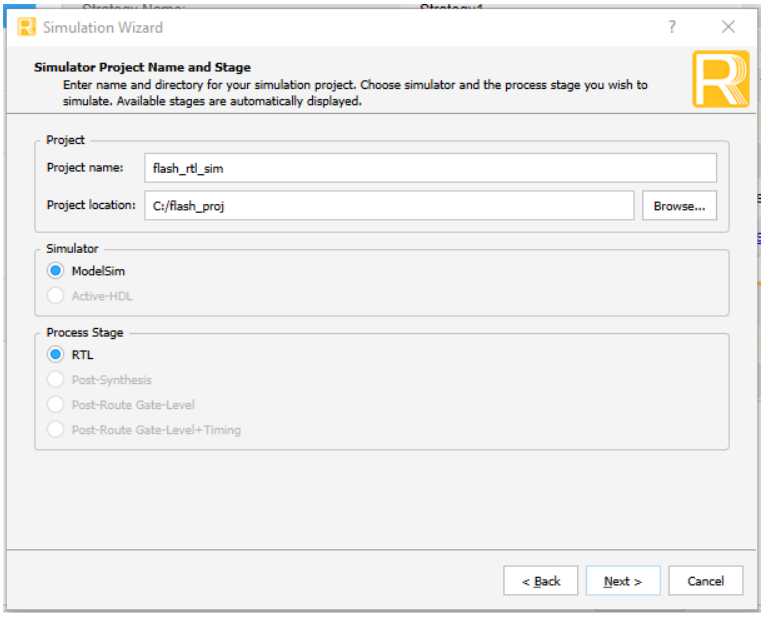

Figure 3.5. Simulation Wizard

2. Click **Next** to open the **Add and Reorder Source** window as shown in Figure 3.6.

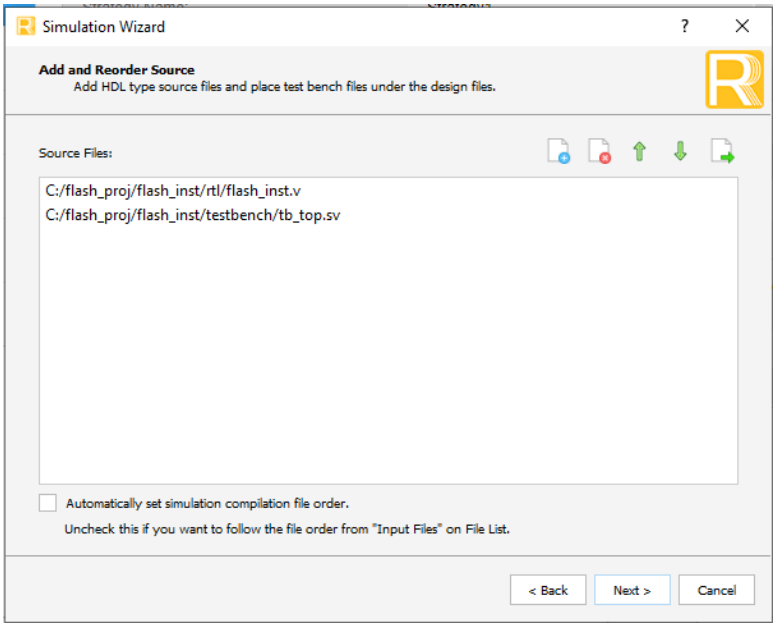

Figure 3.6. Adding and Reordering Source

3. Click **Next** to display the Summary window.
4. To run the simulation, click **Finish**.

3.2.2. Running Other Test Sequence

After running the default test sequence, you can run other test sequences.

The available tests are shown in Table 3.1.

Table 3.1. Other Test Sequences

Test Name	Test Function
check_status_test	Checks the current status of the flash
flash_erase_test	Tests the Erase Function
flash_write_read_test	Test write to and read from flash functions

To run the test sequence:

1. Run the Default Test Sequence as described in Running Default Test Sequence.
2. Go to the Project Simulation directory chosen in the Default Test Sequence, then copy the mdo and TXT files.

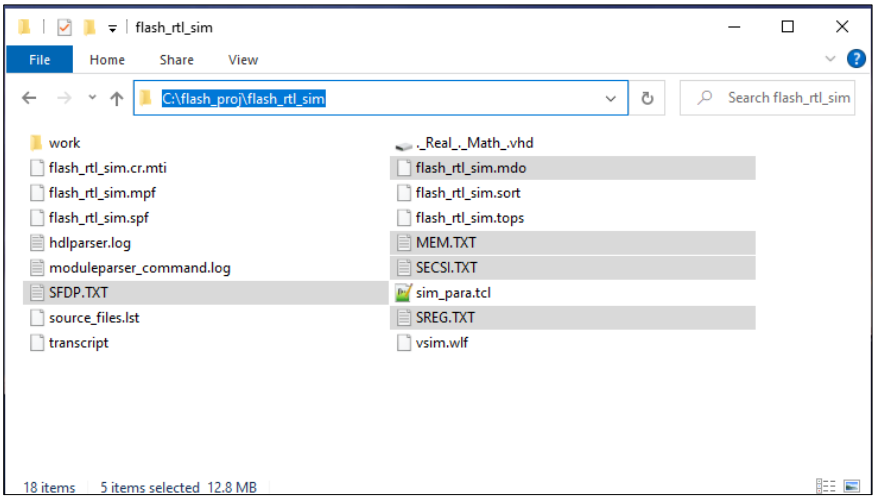

Figure 3.7. Copy from Project Simulation Directory

3. Paste the files on a new folder, *other_tests* under the **Project Location**.

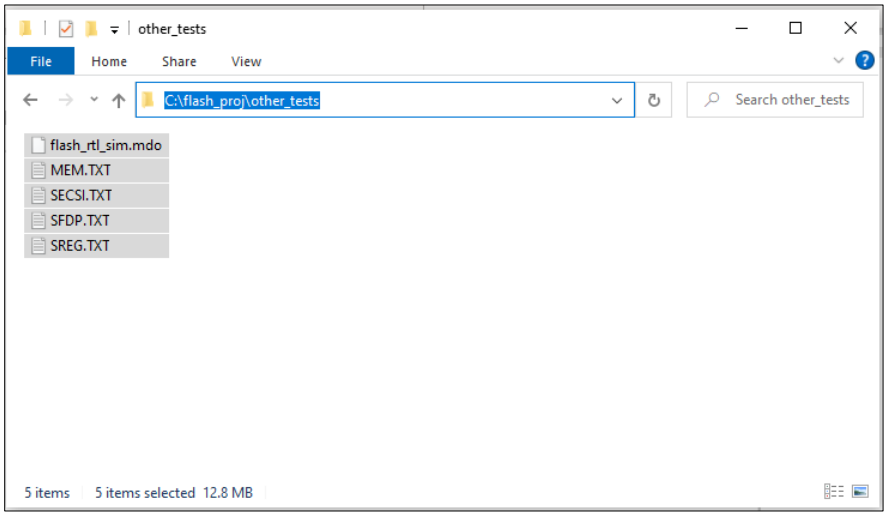

Figure 3.8. Paste to New Directory

4. Open the mdo file for editing. Replace <project simulation name> to *other_tests*.
5. Delete the first line of the mdo file.
6. Add the following options on line 29 of the mdo file:

   +<test name in Table 3.1>

   +iter=<# of iterations>

Figure 3.9 shows the difference between the edited the default mdo file.

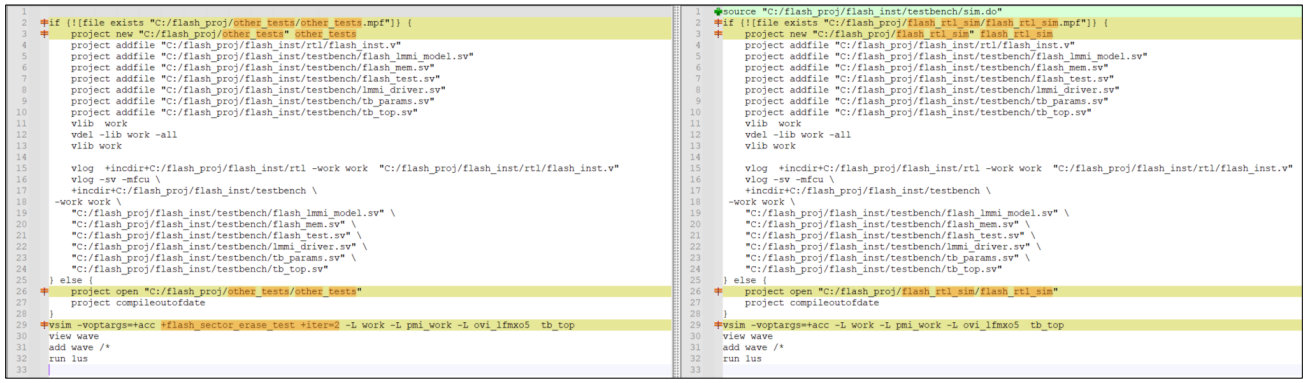

Figure 3.9. Difference of mdo Files

7. Save the edited mdo file.
8. Go back to the Radiant Software, Click the button located on the **Toolbar** to initiate the **ModelSim** software.
9. After opening the **ModelSim** software, type the following in the command line:

```
cd other_tests
do <project simulation name>.mdo
```

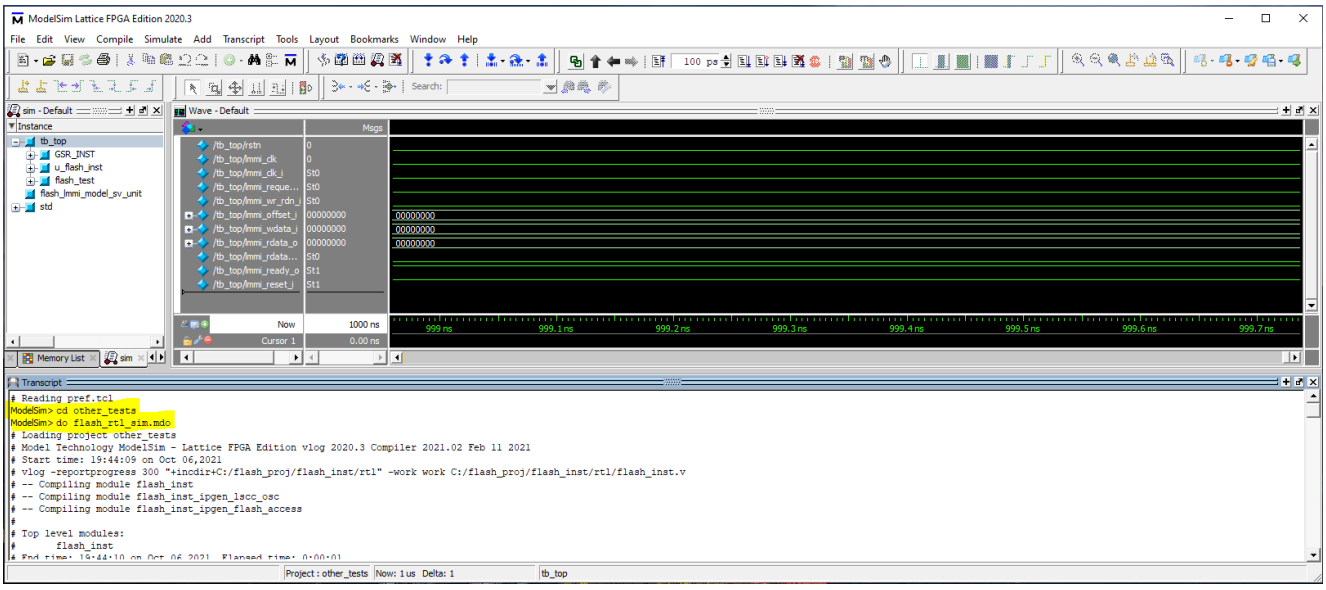

Figure 3.10. Running Other Tests

References

- AMBA 3 AHB-Lite Protocol V1.0
- Lattice Memory Mapped Interface (LMMI) and Lattice Interrupt Interface (LINTR) User Guide (FPGA-UG-02039)

Technical Support Assistance

Submit a technical support case through www.latticesemi.com/techsupport.

Revision History

Document Revision 1.1, Lattice Radiant SW version 3.2, May 2022

Section	Change Summary
Functional Descriptions	• Editorial changes in Section 2.1 Overview. • Updated Figure 2.1. Flash Access Block Diagram (Use HFCLK for LMMI CLK = Unchecked), Figure 2.2. Flash Access Block Diagram (Use HFCLK for LMMI CLK = Checked), Figure 2.3. Sample Connection of Oscillator Soft IP to Flash Access Soft IP, Figure 2.6. Sample Waveform for On-chip Erase Sequence • Updated Table 2.1. Flash Access Module Signal Description. Added rows *clk_i* and *rstn_i*. Table 2.3. Attributes Description, Table 2.4. Flash Memory Map, Table 2.5. Different Combinations of CFGx/UFMx Block Size, Table 2.6. Sample Settings 1, Table 2.7. Sample Settings 2 • Updated description of Section 2.4 Connecting an Instantiated Oscillator Soft IP to the Flash Access Soft IP. • Updated Section 2.7.3 On-chip Flash Erase Sequence. Updated the value of *lmmi_offset_i* from 0x14 to 0x13. • Updated description of Section 2.7.4 Write Access Sequence.

Flash Access IP Generation and Simulation	• Updated Table 3.1. Other Test Sequences.
	  • Removed row flash_reset_test.
	  • Added rows check_status_test and flash_write_read_test.
	  • Changed row flash_sector_erase_test to flash_erase_test.

Document Revision 1.0, Lattice Radiant SW version 3.1, January 2022

Section	Change Summary
All	Initial release.

www.latticesemi.com