SGDMA Controller IP Core

IP Version: v2.6.0

User Guide

FPGA-IPUG-02131-1.7

December 2025

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 all associated risk is the responsibility entirely of the Buyer. The information provided herein is for informational purposes only and may contain technical inaccuracies or omissions, and may be otherwise rendered inaccurate for many reasons, and Lattice assumes no obligation to update or otherwise correct or revise this information. 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. LATTICE PRODUCTS AND SERVICES ARE NOT DESIGNED, MANUFACTURED, OR TESTED FOR USE IN LIFE OR SAFETY CRITICAL SYSTEMS, HAZARDOUS ENVIRONMENTS, OR ANY OTHER ENVIRONMENTS REQUIRING FAIL-SAFE PERFORMANCE, INCLUDING ANY APPLICATION IN WHICH THE FAILURE OF THE PRODUCT OR SERVICE COULD LEAD TO DEATH, PERSONAL INJURY, SEVERE PROPERTY DAMAGE OR ENVIRONMENTAL HARM (COLLECTIVELY, "HIGH-RISK USES"). FURTHER, BUYER MUST TAKE PRUDENT STEPS TO PROTECT AGAINST PRODUCT AND SERVICE FAILURES, INCLUDING PROVIDING APPROPRIATE REDUNDANCIES, FAIL-SAFE FEATURES, AND/OR SHUT-DOWN MECHANISMS. LATTICE EXPRESSLY DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY OF FITNESS OF THE PRODUCTS OR SERVICES FOR HIGH-RISK USES. 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.

Inclusive Language

This document was created consistent with Lattice Semiconductor's inclusive language policy. In some cases, the language in underlying tools and other items may not yet have been updated. Please refer to Lattice's inclusive language FAQ 6878 for a cross reference of terms. Note in some cases such as register names and state names it has been necessary to continue to utilize older terminology for compatibility.

Contents

Figures

Tables

Acronyms in This Document

A list of acronyms used in this document.

Acronym	Definition
AXI4-L	Advanced eXtensible Interface 4 Lite
AXI4-MM	Advanced eXtensible Interface 4 Memory Map (Same as AXI4 or AXI4-Full)
AXI4-S	Advanced eXtensible Interface 4 Stream
MM2S	Memory Map to Stream data transfer
S2MM	Stream to Memory Map data transfer
SGDMA	Scatter Gather Direct Memory Access

1. Introduction

1.1. Overview of the IP

The Scatter Gather Direct Memory Access (SGDMA) Controller IP core provides access to the main memory independent of the processor. It offloads processor intervention. The processor initiates transfer to SGDMA Controller and receives an interrupt on completion of the transfer by the DMA Engine.

The Lattice SGDMA Controller IP core implements a configurable DMA controller with scatter-gather capability. The directions for specifying the IP core configuration, including it in a user design, and directions for simulation and synthesis are provided in this user guide.

1.2. Quick Facts

Table 1.1. Summary of the SGDMA Controller IP

IP Requirements	Supported Devices	Certus™-NX, CertusPro™-NX, CrossLink™-NX, Lattice Avant™-AT-E, Lattice Avant-AT-G (G50 and G70), Lattice Avant-AT-X (X50 and X70), MachXO5™-NX, and Certus-N2
	IP Changes	Refer to the SGDMA Controller IP Release Notes (FPGA-RN-02058).
Resource Utilization	Supported User Interface	AXI-L, AXI-MM, AXI-S, APB
Design Tool Support	Lattice Implementation	IP Core v2.6.0 – Lattice Radiant™ software 2025.2 or later
	Synthesis	Synopsys Synplify® Pro for Lattice
	Simulation	Refer to the Lattice Radiant Software User Guide for the list of supported simulators.
Driver Support	API Reference	Refer to the SGDMA Driver API Reference document.

Notes:

1. In some instances, the IP may be updated without changes to the user guide. This user guide may reflect an earlier IP version but remains fully compatible with the later IP version. Refer to the IP Release Notes for the latest updates.
2. Lattice Implementation indicates the IP version release coinciding with the software version release. Check the software for IP version compatibility with earlier or later software versions.

1.3. IP Support Summary

Table 1.2. IP Support Readiness on Lattice FPGAs and Lattice Radiant Software Suite

Device Family	IP	Enable Separate Clock Domain	AXI-S Data Width	AXI-MM Data Width	AXI-MM Address Width	Radiant Timing Model
CertusPro-NX Device Family (From Speed Grade of 7 onwards)	SGDMA Controller IP Core	Yes	8, 16, 32, 64, 128	32, 64, 128	32, 64	Final
		No				
Avant-AT-E/G/X Device Family (From Speed Grade of 1 onwards)	SGDMA Controller IP Core	Yes	8, 16, 32, 64, 128	32, 64, 128	32, 64	Advance
		No				
Certus-NX Device Family (From Speed Grade of 7 onwards)	SGDMA Controller IP Core	Yes	8, 16, 32, 64, 128	32, 64, 128	32, 64	Final
		No				
MachXO5-NX Device Family (From Speed Grade of 7 onwards)	SGDMA Controller IP Core	Yes	8, 16, 32, 64, 128	32, 64, 128	32, 64	Advance
		No				
Crosslink-NX Device Family (From Speed Grade of 7 onwards)	SGDMA Controller IP Core	Yes	8, 16, 32, 64, 128	32, 64, 128	32, 64	Final
		No				

Device Family	IP	Enable Separate Clock Domain	AXI-S Data Width	AXI-MM Data Width	AXI-MM Address Width	Radiant Timing Model
Certus-N2 Device Family (From Speed Grade of 1 onwards)	SGDMA Controller IP Core	Yes	8,16,32,64,128	32, 64, 128	32, 64	Advance
		No				

1.4. Features

The key feature of the IP includes:

- Independent MM2S and S2MM data buffer
  - MM2S: AXI-MM to AXI-Stream
  - S2MM: AXI-Stream to AXI-MM
- AXI4 Protocol compliant
  - MM2S/S2MM AXI4-MM Address width support of 32 or 64 bits.
  - MM2S/S2MM AXI4-MM Data width support of 32, 64 or 128 bits.
  - MM2S/S2MM AXI4-Stream Data width support of 8, 16, 32, 64, or 128 bits.
  - Buffer Descriptor AXI4-MM Address and Data width support of 32 bits.
  - AXI4-Lite Address and Data width support of 32 bits.
- IP's Control and Status Registers access through two industry standard protocols
  - APB with Address and Data width support of 32 bits
  - AXI4-Lite with Address and Data width support of 32 bits.
- Max transfer size per descriptor support of 2^16 Bytes.
- Any-to-any AXI-MM and AXI-S Data Width streaming.
- Configurable MM2S/S2MM local FIFO Depth support of 512, 1024, 2048 or 4096.
- Byte ordering: Little Endian

AMBA Specifications:

This IP implements AXI interface compliant to the AMBA4 specifications:

- AXI4-MM
- AXI4-Stream
- AXI4-Lite
- APB3

For more information, go to http://www.arm.com/products/system-ip/amba/amba-open-specifications.php.

1.5. Licensing and Ordering Information

The SGDMA IP is provided at no additional cost with the Lattice Radiant software.

1.6. Hardware Support

Refer to the Example Design section for more information on the boards used.

1.7. Speed Grade Supported

The Lattice SGDMA IP core supported speed grade is provided in this section. Different configurations may be supported using different speed grades due to the fabric performance requirement.

- 9 – fastest speed grade for CertusPro-NX and Certus-NX
- 3 – fastest speed grade for Avant

Table 1.3. Lattice SGDMA IP Core Supported Speed Grade's Maximum Frequency for Each Individual Clock Domain during IP Standalone Compilation at the 0 Degrees Corner Scenario

Device	Enable Separate Clock Domain	AXI MM		AXI Streaming	FIFO Buffer Depth		Maximum Frequency/MHz			
		DWIDTH	AWIDTH	TDATA	MM2S	S2MM	m_axis_clk	s_axis_clk	axi_mm_clk	axil_clk
LFD2NX-40-7BG196C	✓	32	32	32	1024	1024	140.3	154.0	128.7	175.4
LFD2NX-40-8BG196C	✓	32	32	32	1024	1024	161.7	184.8	147.1	200.0
LFD2NX-40-9BG196C	✓	32	32	32	1024	1024	174.2	188.1	150.5	200.0
LFMXO5-65-7BBG400C	✓	32	32	32	1024	1024	151.8	154.0	148.4	200.0
LFMXO5-65-8BBG400C	✓	32	32	32	1024	1024	179.3	200.0	157.8	200.0
LFMXO5-65-9BBG400C	✓	32	32	32	1024	1024	184.0	184.8	179.0	200.0
LAV-AT-E70ES1-1LFG676C	✓	32	32	32	1024	1024	180.4	241.3	250.0	250.0
LAV-AT-E70ES1-2LFG676C	✓	32	32	32	1024	1024	218.5	250.0	250.0	250.0
LAV-AT-E70ES1-3LFG676C	✓	32	32	32	1024	1024	250.0	250.0	250.0	250.0
LFCPNX-100-7LFG672C	✓	32	32	32	1024	1024	143.6	161.1	133.3	180.3
LFCPNX-100-8LFG672C	✓	32	32	32	1024	1024	157.4	170.4	150.5	196.2
LFCPNX-100-9LFG672C	✓	32	32	32	1024	1024	166.4	200.0	168.2	199.9
LIFCL-40-7SG72C	✓	32	32	32	1024	1024	125.9	152.2	133.5	164.7
LIFCL-40-8SG72C	✓	32	32	32	1024	1024	149.1	189.2	149.2	200.0
LIFCL-40-9SG72C	✓	32	32	32	1024	1024	169.5	189.3	144.2	200.0
LN2-CT-20-1CBG484C	✓	32	32	32	1024	1024	231.7	242.3	250.0	250.0
LN2-CT-20-2CBG484C	✓	32	32	32	1024	1024	242.5	250.0	250.0	250.0
LN2-CT-20-3CBG484C	✓	32	32	32	1024	1024	250.0	250.0	250.0	250.0

1.8. Naming Conventions

1.8.1. Nomenclature

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

1.8.2. Signal Names

- _*n* are active low (asserted when value is logic 0)
- _*i* are input signals
- _*o* are output signals

2. Functional Description

2.1. IP Architecture Overview

The SGDMA Controller IP core provides high-bandwidth non-contiguous memory access between an AXI stream controller and an AXI memory-mapped component without a processor (system CPU) having to be involved.

SGDMA Controller operation and status registers are accessed through the AXI Lite or APB interface to initialize and kick-start the SGDMA engine for data transfer and the status register to indicate the SGDMA operation status.

The descriptor manager is responsible for reading the buffer descriptor through the AXI Memory Map and updating the buffer descriptor status field to update the operation status at the end of the data transfer.

There are two primary data stream engines, S2MM and MM2S, to transfer data from the AXI Streaming Controller to the AXI Memory Map component, or vice versa. Both S2MM and MM2S can be operated in parallel, and they are independent of each other.

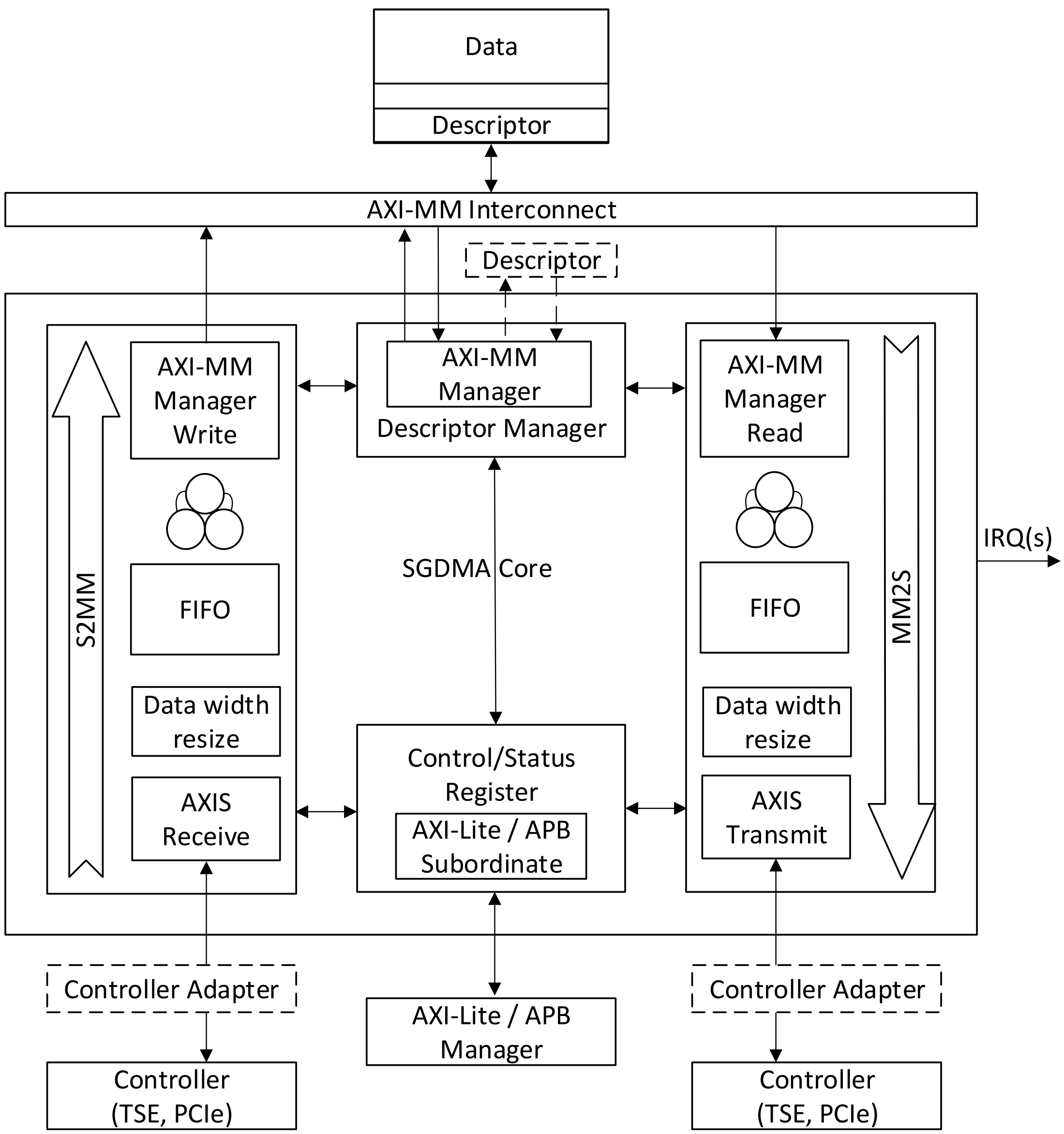

Figure 2.1. SGDMA Controller Core Block Diagram

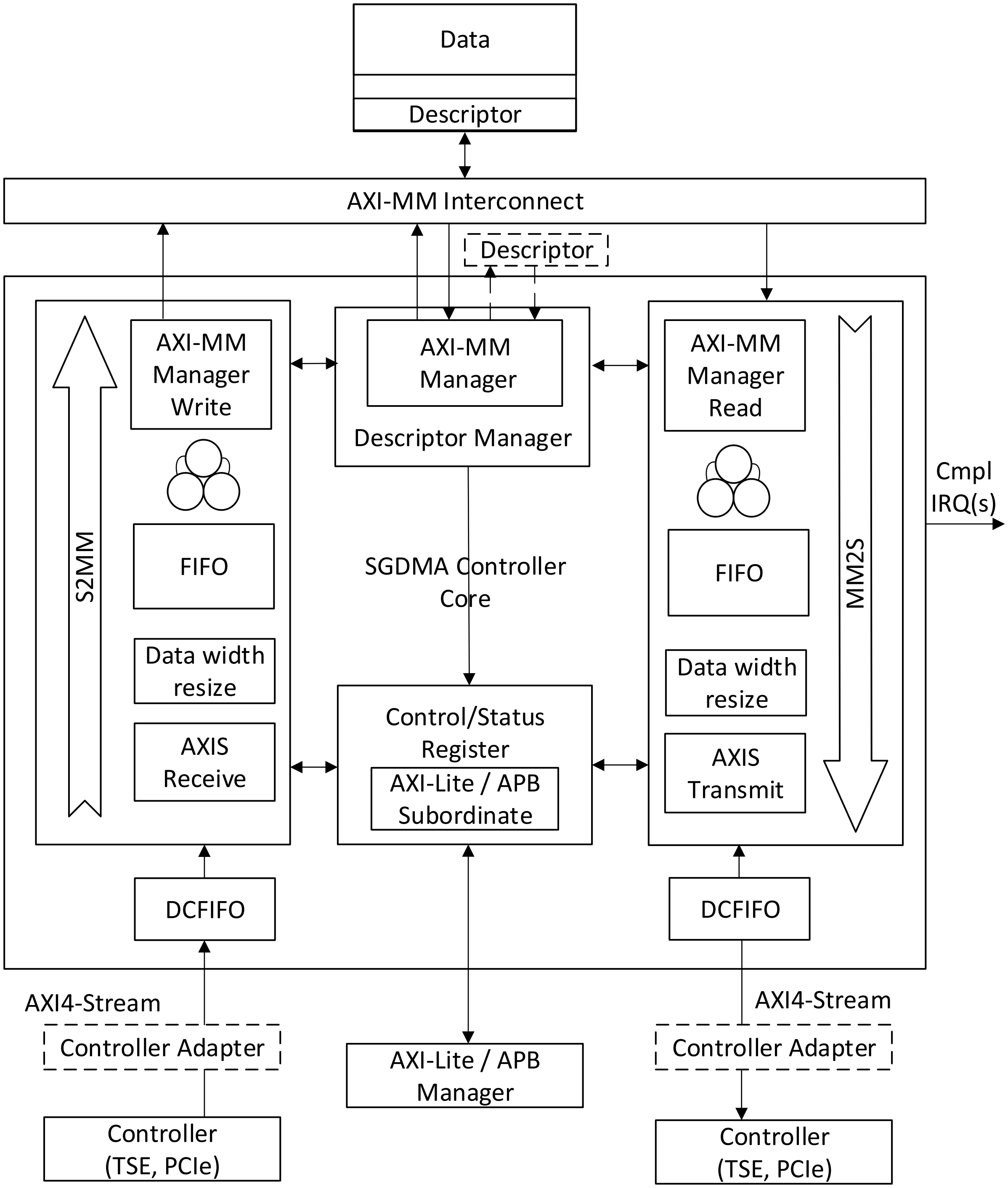

Figure 2.2. SGDMA Controller Core Block Diagram with Separate Clock Domains for AXI-S Transmitter and Receiver Enabled

2.2. Clocking

By default, there are two clock domains in the SGDMA Controller Core:

- AXI4-Lite Clock – for AXI4-Lite protocol and SGDMA Controller CSR module.
- IP Clock – for components in SGDMA Controller Core except AXI4-Lite and CSR module.

The *Enable Separate Clock Domains for AXI4-S Transmitter and Receiver* checkbox parameter enables an additional two clock domains for the AXI-S Transmitter and Receiver interfaces present on the IP.

- AXI4-Lite Clock – for AXI4-Lite protocol and SGDMA Controller CSR module.
- AXI4-MM Clock – for components in SGDMA Controller Core except AXI4-Lite, AXI-S Transmitter, AXI-S Receiver and CSR module. Replaces IP Clock.
- AXI4-S Transmitter Clock – for the AXI-S Transmitter interface of the SGDMA Controller core.
- AXI4-S Receiver Clock – for the AXI-S Receiver interface of the SGDMA Controller core.

When the *CSR Access Interface* parameter is switched to APB, the AXI4-Lite clock is replaced by the APB clock.

2.2.1. SGDMA Controller Operation Clocking Overview

Table 2.1. Two-Clock Domain Attributes

Port Name	Description
clk	SGMDA Controller Core clock (100 to 125 MHz)
axil_clk	AXI4-Lite clock (100 to 125 MHz)
s_apb_pclk_i	APB clock (100 to 125 MHz)
axi_mm_clk	AXI-MM clock (Validated 200 MHz for Lattice Avant devices and 100 MHz for the rest of Lattice FPGA devices). This replaces SGDMA Controller Core clock when additional clock domains for AXI-S Transmitter and Receiver enabled.
m_axis_clk	AXI-S Transmitter clock (100 to 125 MHz)
s_axis_clk	AXI-S Receiver clock (100 to 125 MHz)

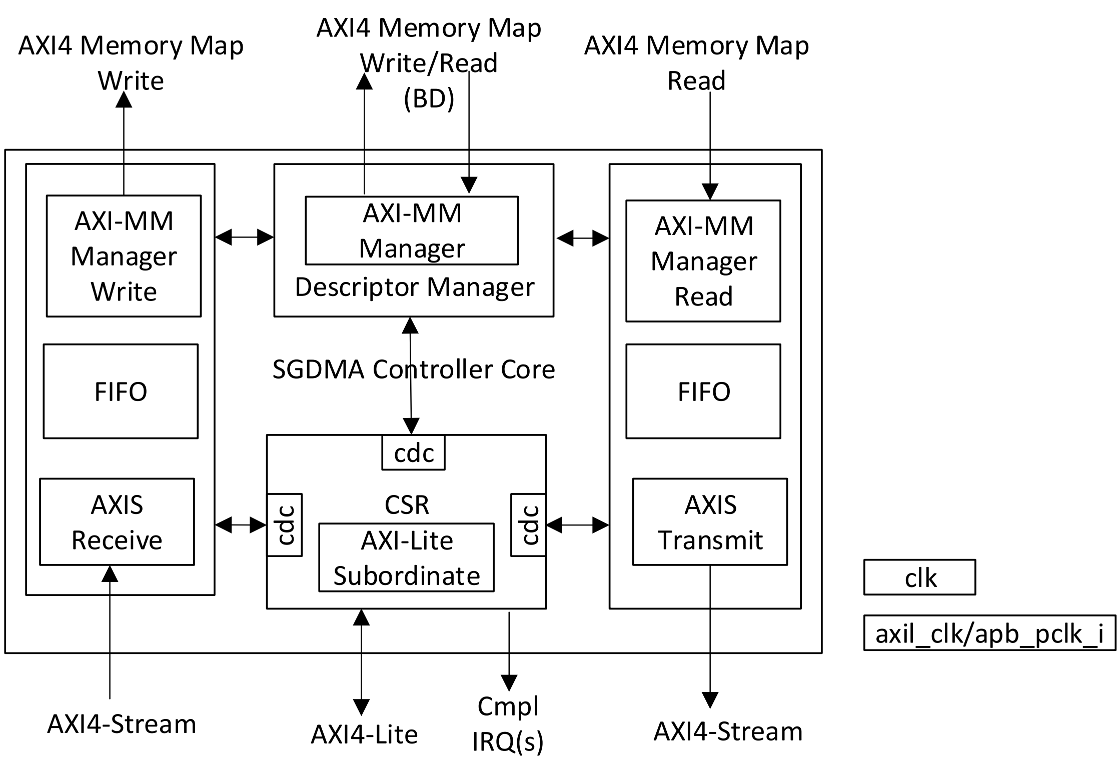

Figure 2.3. SGDMA Controller IP Clock Domain Block Diagram

2.3. Reset

By default, there are two reset ports in the SGDMA Controller Core:

- AXI4-Lite reset – Active low reset for AXI4-Lite protocol and SGDMA Controller CSR module.
- Global reset – Active low reset for components in SGDMA Controller Core except AXI4-Lite and CSR module.

The *Enable Separate Clock Domains for AXI4-S Transmitter and Receiver* checkbox parameter enables an additional two clock domains for the AXI-S Transmitter and Receiver interfaces present on the IP.

- AXI4-Lite Reset – for AXI4-Lite protocol and SGDMA Controller CSR module.
- AXI4-MM Reset – for components in SGDMA Controller Core except AXI4-Lite, AXI-S Transmitter, AXI-S Receiver and CSR module. Replaces Global reset.
- AXI4-S Transmitter Reset – for the AXI-S Transmitter interface of the SGDMA Controller core.
- AXI4-S Receiver Reset – for the AXI-S Receiver interface of the SGDMA Controller core.

When the *CSR Access Interface* parameter is switched to APB, the AXI4-Lite Reset is replaced by the APB Reset.

Other than interface-based resets, there are two register-based register bits reset to reset MM2S and S2MM Engine respectively.

- Offset 0x00 bit1 – write 1 to generate a reset pulse (sync to clk domain) to reset MM2S engine.
- Offset 0x20 bit1 – write 1 to generate a reset pulse (sync to clk domain) to reset S2MM engine

2.3.1. Reset Overview

The reset ports in the SGDMA are active low and should be synchronized to the corresponding clock domain outside the IP.

Table 2.2. Reset Attributes

Reset	Description
rstn	Active low reset synchronized to SGMDA Controller Core clock (clk)
axil_rstn	Active low reset synchronized to AXI4-Lite clock (axil_clk)
apb_presetn_i	Active low reset synchronized to APB clock (s_apb_pclk_i)
axi_mm_rstn	Active low reset synchronized to AXI-MM clock (axi_mm_clk)
m_axis_rstn	Active low reset synchronized to AXI-S Transmitter clock (m_axis_clk)
s_axis_rstn	Active low reset synchronized to AXI-S Receiver clock (s_axis_clk)

On register-based reset, writing 1 to the MM2S/S2MM_CTRL.RESET (Control and Status Registers Description section) bit resets the corresponding SGDMA engine. When this bit is written, it:

- Resets MM2S/S2MM Engine FSM.
- Resets all data and array pointers in MM2S/S2MM buffer.

When to reset:

- Before MM2S/S2MM_CTRL.REQUEST bit is set, to reset the engine.
- When MM2S/S2MM operation hangs.
- It is recommended to perform reset for all the reset ports simultaneously. For best practice, all reset ports should be connected to a common global reset signal. This issue will be fixed in future release.

2.3.1.1. Known Behavior

After a reset, the AXI-Stream interface may output unintended data. This data can be safely ignored if only one reset domain is performed, or if the S2MM/MM2S CSR reset bit is triggered.

2.4. Functional Operation

2.4.1. MM2S Operation

After the AXI4-Lite configuration completion (CSR.MM2S_CURDESC and CSR.MM2S_REQUEST), the MM2S engine initiates a buffer descriptor fetch from the memory address defined in MM2S_CURDESC. Upon buffer descriptor information being fetched and updated in MM2S Core, the MM2S engine initiates AXI4 Read to Memory with an address-defined buffer descriptor. MM2S may break the memory read into multiple transactions depending on the BUFFER_SIZE defined in the buffer descriptor, and the MM2S engine is always reading all data defined in BUFFER_SIZE.

The MM2S engine stores received data from memory into a local FIFO and starts to stream the AXI-Stream Controller whenever it is ready. There are two modes available in MM2S: the full packet and the normal model.

In a scenario where the AXI-Stream Controller does not allow TVALID to be de-asserted once TDATA streaming starts, full packet mode can be enabled in the Buffer Descriptor Control FP bit. When this bit is set, the MM2S engine stores all data ready from memory in local FIFO before it starts streaming through AXI-Stream. To prevent FIFO from overflowing, FIFO depth must be configured to a sufficient size, depending on application needs.

In normal mode, when the FP bit is not set, MM2S starts to stream data to the AXI-Stream Controller whenever the FIFO is not empty. It stops with a TVALID deassertion when FIFO is empty. MM2S cannot guarantee the TVALID to be asserted throughout the entire transaction in normal mode.

Once all data has been transferred from memory to the controller, MM2S updates the Buffer Descriptor Status DW with the total data transferred, which is the same as the BUFFER_SIZE, and asserts the CMPL bit.

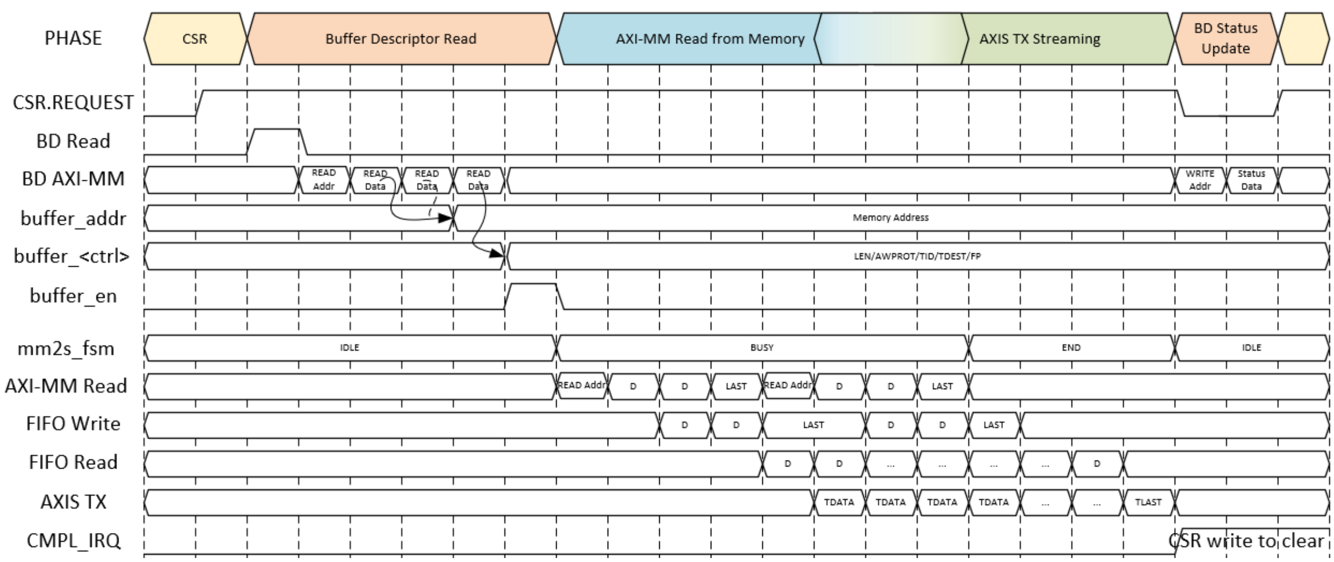

Figure 2.4. SGDMA Controller IP MM2S Operation Phase

2.4.2. S2MM Operation

After the AXI4-Lite configuration completion (CSR.S2MM_CURDESC and CSR.S2MM_REQUEST), the S2MM engine initiates a buffer descriptor fetch from the memory address defined in S2MM_CURDESC. Upon buffer descriptor information being fetched and updated in S2MM Core, it asserts TREADY to the AXI-Stream Controller, indicating DMA is ready for data streaming. The controller starts data streaming, and TLAST indicates the final transferred data.

S2MM engine stores received TDATA from AXI-Stream into local FIFO and starts to write data to memory through AXI4 Manager Write whenever FIFO is not empty. S2MM may break the data transfer into multiple AXI4 transactions, and it automatically calculates the transfer length according to FIFO availability (the maximum transfer length is 256 according to AMBA specifications).

In scenarios where AXI4 Write to Memory is slower than AXI-Stream Controller data streaming, to avoid FIFO overflow, it deasserts TREADY to backpressure AXI-Stream Controller when FIFO is almost full. Local FIFO depth can be parameterized during IP generation.

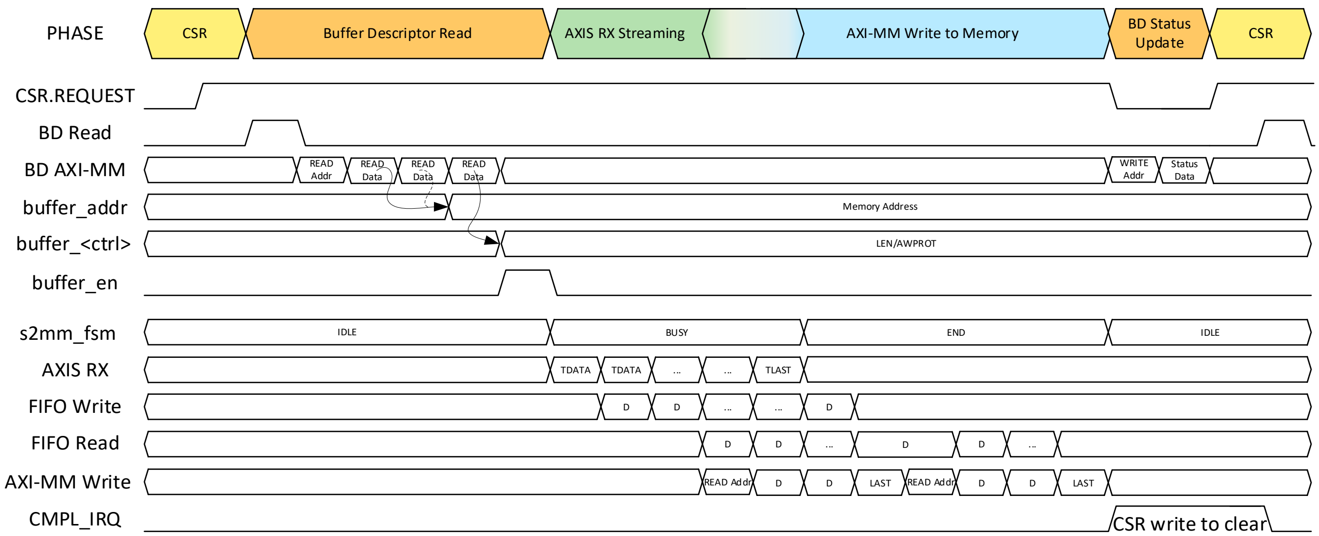

Figure 2.5 SGDMA Controller IP S2MM Operation Phase

In S2MM, the BUFFER_SIZE is allocated. However, the AXI-Stream Controller may send fewer data than the allocated size. In this scenario, the S2MM engine writes all received streamed data to memory and update Buffer Descriptor Status DW with the total actual data transferred (byte) and CMPL bit. In other scenarios, when the AXI-Stream Controller sends more data than the allocated size, the S2MM engine only transfers the maximum data defined by BUFFER_SIZE and drop the remaining incoming TDATA. It updates the Buffer Descriptor Status DW with the total actual data transferred (byte), which is the same as BUFFER_SIZE, and assert the CMPL bit together with the ERROR bit.

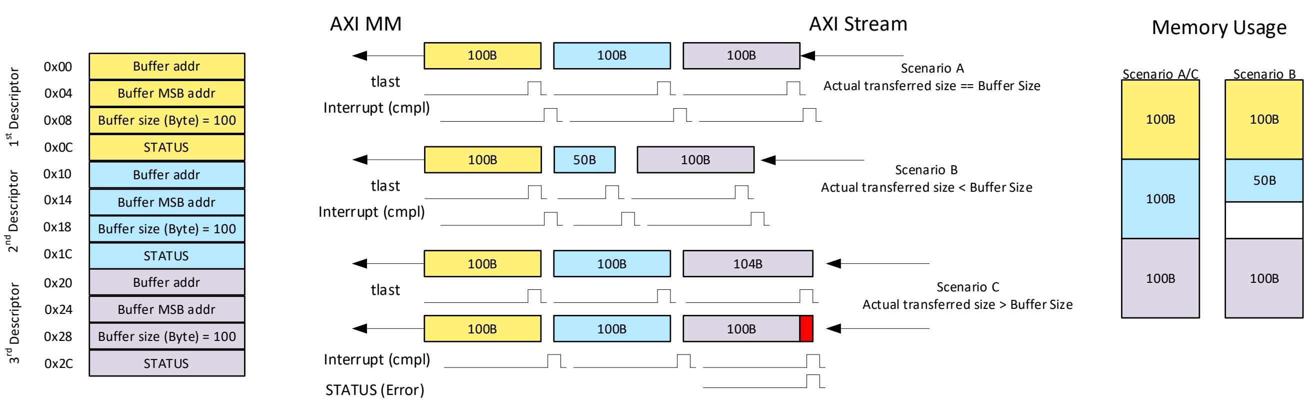

Figure 2.6. SGDMA Controller IP S2MM Buffer Size Allocation

2.4.3. Completion Interrupt Operation

Upon MMS2 or S2MM transfer operations are completed, the SGDMA Controller asserts mm2s_xfer_cmpl_irq_o and s2mm_xfer_cmpl_irq_o, respectively. The IRQ is only initiated per REQUEST operation. For a multiple buffer descriptor request operation (NXT = 1), the IRQ is only triggered at the end of all BD served to indicate the series of data transfer completions. When only a single buffer descriptor request operation (NXT=0) is executed, the IRQ is triggered at the end of the respective BD served.

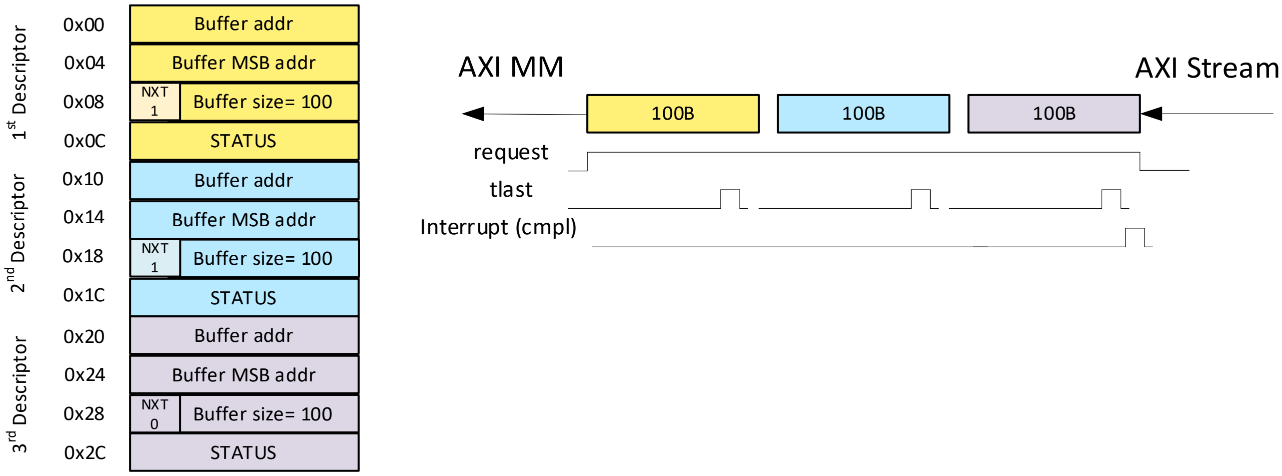

Figure 2.7. IRQ assertion per REQUEST (Multiple BD)

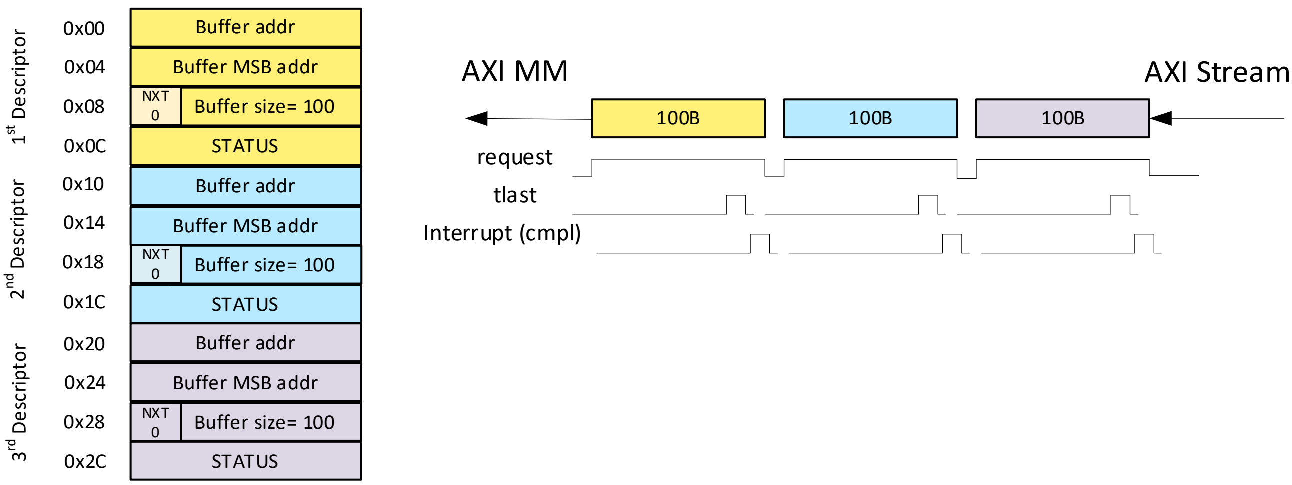

Figure 2.8. IRQ assertion per REQUEST (Single BD)

The IRQ signal is a level indication that stays asserted until the system driver reads the MM2S/S2MM_STS register to clear the register and deassert the respective IRQ pin. If there are multiple REQUEST completions and the system driver did not read the MM2S/S2MM_STS register, the IRQ pin is asserted from the first REQUEST completion and mask all the following completion indications. It is recommended for the system driver to read the status register before starting the next REQUEST.

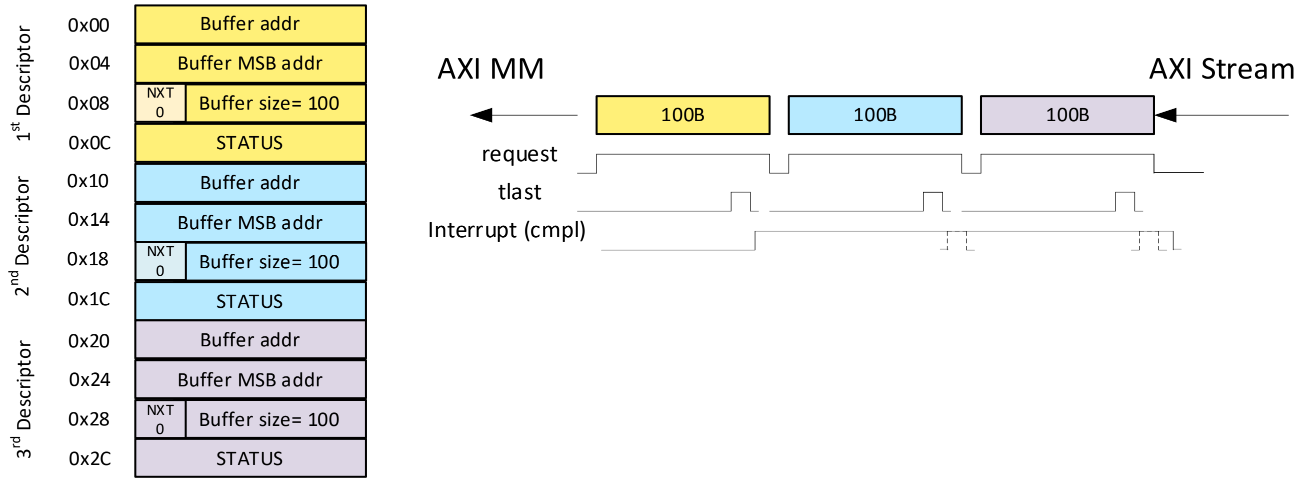

Figure 2.9. IRQ stay asserted if no STATUS bis is read after REQUEST completion

3. IP Parameter Description

The SGDMA Controller IP user interface has only one parameter tab, which allows you to customize the IP setting.

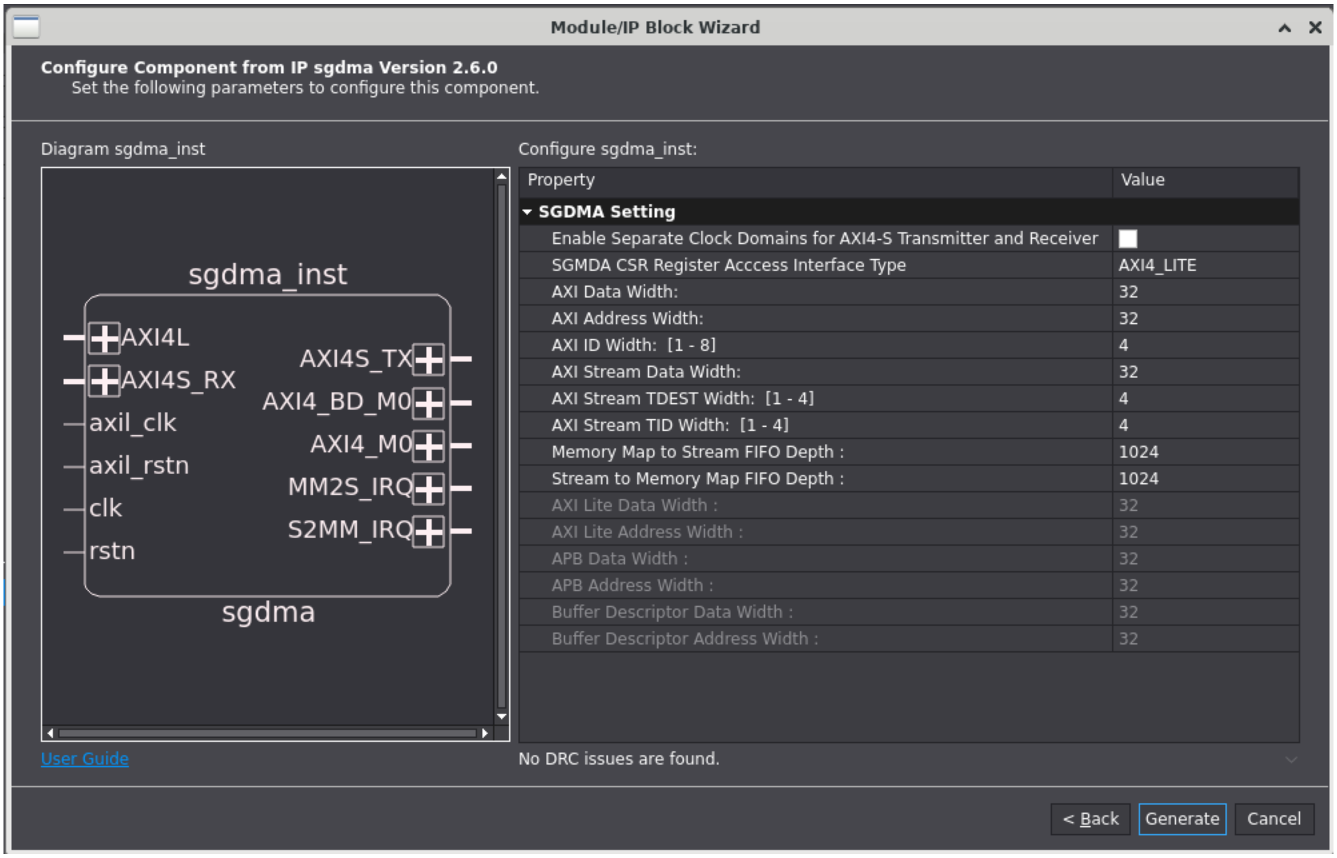

Figure 3.1. SGDMA Controller IP User Interface

3.1. General

The configurable attributes of the SGDMA Controller IP are shown in Table 3.1. You can configure the IP by setting the attributes accordingly in the IP Catalog Module/IP wizard of the Lattice Radiant software.

Where applicable, the default values are in bold.

Table 3.1. General Attributes

Attribute Name	Selectable Value	Description
Enable separate clock domains for AXI-S Transmitter and Receiver (XCLK_ENABLE)	0, 1	SGMDA Controller Core clock domain splits into three new clock domains for greater system design flexibility.
AXI Data Width (AXI_DWIDTH)	32, 64, 128	To configure MM2S and S2MM AXI Data Width
AXI Address Width (AXI_AWIDTH)	32, 64	To configure MM2S and S2MM AXI Address Width
AXI ID Width (AXI_ID_WIDTH)	1, 2, 3, 4, 5, 6, 7, 8	To configure MM2S, S2MM and BD AXI ID Width
AXI Stream Data Width (TDATA_WIDTH)	8, 16, 32, 64, 128	To configure MM2S and S2MM AXIS TDATA Width
AXI Stream TDEST Width (TDEST_WIDTH)	1, 2, 3, 4	To configure MM2S and S2MM AXIS TID Width

Attribute Name	Selectable Value	Description
AXI Stream TID Width (TID_WIDTH)	1, 2, 3, 4	To configure MM2S and S2MM AXIS TDEST Width
Memory Map to Stream FIFO Depth (MM2S_FIFO_DEPTH)	512, 1024, 2048, 4096	To configure MM2S FIFO Depth
Stream to Memory Map FIFO Depth (S2MM_FIFO_DEPTH)	512, 1024, 2048, 4096	To configure S2MM FIFO Depth
AXI Lite Data Width (AXIL_DWIDTH)	32	Only supports 32 bits AXI Lite Data Width
AXI Lite Address Width (AXIL_AWIDTH)	32	Only supports 32 bits AXI Lite Address Width
APB Lite Data Width (APB_DWIDTH)	32	Only supports 32 bits APB Data Width
APB Lite Address Width (APB_AWIDTH)	32	Only supports 32 bits APB Address Width
Buffer Descriptor AXI Data Width (BD_DWIDTH)	32	Only supports 32 bits Buffer Descriptor AXI Data Width
Buffer Descriptor AXI Address Width (BD_AWIDTH)	32	Only supports 32 bits Buffer Descriptor AXI Address Width

4. Signal Description

This section describes the SGDMA Controller IP Core ports.

Table 4.1. Signal List

Port Name	Width	Direction	Description
Clock and Reset			
Separate Clock Domains for AXI-S Transmitter and Receiver Disabled (Default Setting)			
clk	1	I	SGMDA Controller Core clock (100 to 125 MHz)
rstn	1	I	SGMDA Controller Global reset (active low)
Separate Clock Domains for AXI-S Transmitter and Receiver Enabled			
s_axis_clk	1	I	AXI-S Receiver clock (125 MHz is validated)
s_axis_rstn	1	I	AXI-S Receiver reset (active low)
axi_mm_clk	1	I	AXI-MM clock (100 MHz and 200 MHz are validated)
axi_mm_rstn	1	I	AXI-MM reset (active low)
m_axis_clk	1	I	AXI-S Transmitter clock (125 MHz is validated)
m_axis_rstn	1	I	AXI-S Transmitter reset (active low)
Control and Status Register Access Interface			
axil_clk	1	I	AXI-L clock (100 to 125 MHz)
axil_rstn	1	I	AXI-L reset (active low)
apb_pclk_i	1	I	APB clock (100 to 125 MHz)
apb_presetn_i	1	I	APB reset (active low)
Control and Status Register Interface (Can Select Either Interface for Use)			
AXI4-Lite Interface			
s_axil_awaddr_i	AXIL_AWIDTH	I	AXI4-Lite Subordinate interface to access to SGDMA Controller Configuration and Status Register.
s_axil_awprot_i	3	I	
s_axil_awvalid_i	1	I	
s_axil_awready_o	1	O	
s_axil_wdata_i	AXIL_DWIDTH	I	
s_axil_wstrb_i	AXIL_DWIDTH/8	I	
s_axil_wvalid_i	1	I	
s_axil_wready_o	1	O	
s_axil_bresp_o	2	O	
s_axil_bvalid_o	1	O	
s_axil_bready_i	1	I	
s_axil_araddr_i	AXIL_AWIDTH	I	
s_axil_arprot_i	3	I	
s_axil_arvalid_i	1	I	
s_axil_arready_o	1	O	
s_axil_rdata_o	AXIL_DWIDTH	O	
s_axil_rresp_o	2	O	
s_axil_rvalid_o	1	O	
s_axil_rready_i	1	I	
APB Interface			
s_apb_pwdata_i	APB_DWIDTH	I	APB Subordinate interface to access to SGDMA Controller Configuration and Status Register.
s_apb_paddr_i	APB_AWIDTH	I	
s_apb_psel_i	1	I	
s_apb_penable_i	1	I	
s_apb_pwrite_i	1	I	
s_apb_prdata_o	APB_DWIDTH	O	

Port Name	Width	Direction	Description
s_apb_pready_o	1	O	
s_apb_pslverr_o	1	O	
Memory Map to Streaming (MM2S) Data Interface			
AXI4-MM Interface			
m_axi_mm2s_arready_i	1	I	MM2S AXI4-MM Manager Write interface to a AXI4-MM Subordinate Device.
m_axi_mm2s_arid_o	AXI_ID_WIDTH	O	
m_axi_mm2s_araddr_o	AXI_AWIDTH	O	
m_axi_mm2s_arregion_o	4	O	
m_axi_mm2s_arlen_o	8	O	
m_axi_mm2s_arsize_o	3	O	
m_axi_mm2s_arburst_o	2	O	
m_axi_mm2s_arlock_o	1	O	
m_axi_mm2s_arcache_o	4	O	
m_axi_mm2s_arprot_o	3	O	
m_axi_mm2s_arqos_o	4	O	
m_axi_mm2s_arvalid_o	1	O	
m_axi_mm2s_rready_o	1	O	
m_axi_mm2s_rid_i	AXI_ID_WIDTH	I	
m_axi_mm2s_rdata_i	AXI_DWIDTH	I	
m_axi_mm2s_rresp_i	2	I	
m_axi_mm2s_rlast_i	1	I	
m_axi_mm2s_rvalid_i	1	I	
AXI4-Stream Interface			
tx_axis_mm2s_tready_i	1	I	MM2S AXI4-Stream Transmitter interface to a AXI4-Stream Receiver Controller.
tx_axis_mm2s_tvalid_o	1	O	
tx_axis_mm2s_tdata_o	TDATA_WIDTH	O	
tx_axis_mm2s_tkeep_o	TDATA_WIDTH/8	O	
tx_axis_mm2s_tlast_o	1	O	
tx_axis_mm2s_tid_o	TID_WIDTH	O	
tx_axis_mm2s_tdest_o	TDEST_WIDTH	O	
Streaming to Memory Map (S2MM) Data Interface			
AXI4-MM Interface			
m_axi_s2mm_awready_i	1	I	S2MM AXI4-MM Manager Read interface to a AXI4-MM Subordinate Device.
m_axi_s2mm_awid_o	AXI_ID_WIDTH	O	
m_axi_s2mm_awaddr_o	AXI_AWIDTH	O	
m_axi_s2mm_awregion_o	4	O	
m_axi_s2mm_awlen_o	8	O	
m_axi_s2mm_awsize_o	3	O	
m_axi_s2mm_awburst_o	2	O	
m_axi_s2mm_awlock_o	1	O	
m_axi_s2mm_awcache_o	4	O	
m_axi_s2mm_awprot_o	3	O	
m_axi_s2mm_awqos_o	4	O	
m_axi_s2mm_awvalid_o	1	O	
m_axi_s2mm_wready_i	1	I	
m_axi_s2mm_wdata_o	AXI_DWIDTH	O	
m_axi_s2mm_wstrb_o	AXI_DWIDTH/8	O	
m_axi_s2mm_wlast_o	1	O	

Port Name	Width	Direction	Description
m_axi_s2mm_wvalid_o	1	O	
m_axi_s2mm_bready_o	1	O	
m_axi_s2mm_bid_i	AXI_ID_WIDTH	I	
m_axi_s2mm_bresp_i	2	I	
m_axi_s2mm_bvalid_i	1	I	
AXI4-Stream Interface			
rx_axis_s2mm_tvalid_i	1	I	S2MM AXI4-Stream Receiver interface to an AXI4-Stream Transmitter Controller. You must handle TKEEP accordingly as per the AMBA AXI-Streaming protocol specification.
rx_axis_s2mm_tdata_i	TDATA_WIDTH	I	
rx_axis_s2mm_tkeep_i	TDATA_WIDTH/8	I	
rx_axis_s2mm_tlast_i	1	I	
rx_axis_s2mm_tid_i	TID_WIDTH	I	
rx_axis_s2mm_tdest_i	TDEST_WIDTH	I	
rx_axis_s2mm_tready_o	1	O	
Buffer Descriptor Data Interface			
AXI4-MM Interface			
m_axi_bd_awready_i	1	I	SGDMA Buffer Descriptor AXI4-MM Write and Read manager to BD AXI4-MM Subordinate component.
m_axi_bd_awid_o	AXI_ID_WIDTH	O	
m_axi_bd_awaddr_o	BD_AWIDTH	O	
m_axi_bd_awregion_o	4	O	
m_axi_bd_awlen_o	8	O	
m_axi_bd_awsize_o	3	O	
m_axi_bd_awburst_o	2	O	
m_axi_bd_awlock_o	1	O	
m_axi_bd_awcache_o	4	O	
m_axi_bd_awprot_o	3	O	
m_axi_bd_awqos_o	4	O	
m_axi_bd_awvalid_o	1	O	
m_axi_bd_wready_i	1	I	
m_axi_bd_wdata_o	BD_DWIDTH	O	
m_axi_bd_wstrb_o	BD_DWIDTH/8	O	
m_axi_bd_wlast_o	1	O	
m_axi_bd_wvalid_o	1	O	
m_axi_bd_bready_o	1	O	
m_axi_bd_bid_i	AXI_ID_WIDTH	I	
m_axi_bd_bresp_i	2	I	
m_axi_bd_bvalid_i	1	I	
m_axi_bd_arready_i	1	I	
m_axi_bd_arid_o	AXI_ID_WIDTH	O	
m_axi_bd_araddr_o	BD_AWIDTH	O	
m_axi_bd_arregion_o	4	O	
m_axi_bd_arlen_o	8	O	
m_axi_bd_arsize_o	3	O	
m_axi_bd_arburst_o	2	O	
m_axi_bd_arlock_o	1	O	
m_axi_bd_arcache_o	4	O	
m_axi_bd_arprot_o	3	O	
m_axi_bd_arqos_o	4	O	
m_axi_bd_arvalid_o	1	O	

Port Name	Width	Direction	Description
m_axi_bd_rready_o	1	O	
m_axi_bd_rid_i	AXI_ID_WIDTH	I	
m_axi_bd_rdata_i	BD_DWIDTH	I	
m_axi_bd_rresp_i	2	I	
m_axi_bd_rlast_i	1	I	
m_axi_bd_rvalid_i	1	I	
Interrupt Event Interface			
mm2s_xfer_cmpl_irq_o	1	O	MM2S Buffer transfer completion interrupt signal. 1 – to indicate there was a data transfer completed. IRQ stays *1* until the driver clears the status. See Section 5.1.3 Status Register Field.
s2mm_xfer_cmpl_irq_o	1	O	S2MM Buffer transfer completion interrupt signal. 1 – to indicate there was a data transfer completed. IRQ stays *1* until the driver clears the status. See Section 5.1.6 Status Register Field.

5. Register Description

5.1. Control and Status Registers Description

All registers are accessed through the AXI4-Lite interface. Access types for each register are defined in Table 5.1. The register must be DW-aligned and in Little Endian.

5.1.1. Overview

Table 5.1. Access Types

Access Type	Behavior on Read Access	Behavior on Write Access
RO	Returns register value	Ignores write access
WO	Returns 0	Updates register value
RW	Returns register value	Updates register value
RC	Returns register value	Read to clear/reset the register bit to its default value
RSVD	Returns 0	Ignores write access

Table 5.2. Register Definition

Register Offset	Register Name	Description
0x00	MM2S_CTRL	—
0x04	MM2S_STS	—
0x08	MM2S_CURDESC	—
0x0C-0x1F	RSVD	—
0x20	S2MM_CTRL	—
0x24	S2MM_STS	—
0x28	S2MM_CURDESC	—
0x2C-0x3F	RSVD	—

5.1.2. MM2S_CTRL

Table 5.3. MM2S_CTRL – Memory Map to Streaming Control Register

Field	Name	Access	Default	Description
31:17	RSVD	RO	0	Reserved
16	CMPL_IRQ_MASK	RW	0	To mask out Interrupt event upon transfer completion. 1: To mask out Interrupt event. No IRQ is sent. 0: Enable Interrupt event (Default).
15:2	RSVD	RO	0	Reserved
1	RESET	WO	0	Soft reset to reset MM2S DMA core. Write 1 to generate 1 clock reset pulse MM2S DMA operation and flush all pending command/transfers. This is write-only bit.
0	REQUEST	RW	0	Request to start DMA operation. 1: Request to start DMA operation Upon current DMA operation completed, this bit is cleared to 0 automatically. If manually cleared REQUEST to 0 during DMA operation does not have any impact to the functionality; operation is continued until it complete.

5.1.3. MM2S_STS

Table 5.4. MM2S_STS – Memory Map to Streaming Status Register

Field	Name	Access	Default	Description
31:18	RSVD	RO	0	Reserved
17	XFER_ERR	RC	0	Current Transfer detected error. 1: An error detected during transfer between previous and current MM2S_STS read. A read cycle to this register bit clears the bit.
16	XFER_CMPL	RC	0	Current Transfer status. 1: 1 or more MM2S transfer completed between previous and current MM2S_STS read. A read cycle to this register bit clears the bit and deasserts the IRQ pin. **Note:** A MM2S Transfer is referring to 1 REQUEST bit set.
15:11	RSVD	RO	0	Reserved
10	AXI_DEC_ERR	RC	0	AXI-MM Read RESP = DEC_ERR 1: 1 or more AXI-MM Read return DEC_ERR between previous and current MM2S_STS read. 0: No Error. A read cycle to this register bit clears the bit.
9	AXI_SLV_ERR	RC	0	AXI-MM Read RESP = SLV_ERR 1: 1 or more AXI-MM Read return SLV_ERR between previous and current MM2S_STS read. 0: No Error. A read cycle to this register bit clears the bit.
8	BD_LEN_ERR	RC	0	Descriptor Buffer Length error. 1: 1 or more Descriptor's Buffer with 0 Length detected between previous and current MM2S_STS read. 0: No Error. A read cycle to this register bit clears the bit.
7:1	RSVD	RO	0	Reserved
0	STATUS	RO	0	MM2S DMA operation status 1: DMA operation in progress 0: DMA in IDLE state, no operation pending.

5.1.4. MM2S_CURDESC

Table 5.5. MM2S_CURDESC – Memory Map to Streaming Current Descriptor Pointer

Field	Name	Access	Default	Description
31:0	MM2S_CURDESC	RW	0	Address pointer to Current MM2S Descriptor

5.1.5. S2MM_CTRL

Table 5.6. S2MM_CTRL – Memory Map to Streaming Control Register

Field	Name	Access	Default	Description
31:17	RSVD	RO	0	Reserved
16	CMPL_IRQ_MASK	RW	0	To mask out Interrupt event upon transfer completion. 1: To mask out Interrupt event. No IRQ is sent. 0: Enable Interrupt event (Default).
15:2	RSVD	RO	0	Reserved

Field	Name	Access	Default	Description
1	RESET	WO	0	Soft reset to reset S2MM DMA core. Write 1 to generate 1 clock reset pulse S2MM DMA operation and flush all pending command/transfers. This is write-only bit. .
0	REQUEST	RW	0	Request to start DMA operation. 1: Request to start DMA operation Upon current DMA operation completed, this bit is cleared to 0 automatically. If manually cleared REQUEST to 0 during the DMA operation does not have any impact to the functionality, the operation is continued until it is complete.

5.1.6. S2MM_STS

Table 5.7. S2MM_STS – Memory Map to Streaming Status Register

Field	Name	Access	Default	Description
31:18	RSVD	RO	0	Reserved
17	XFER_ERR	RC	0	Current Transfer detected error. 1: An error detected during transfer between previous and current S2MM_STS read. A read cycle to this register bit clears the bit.
16	XFER_CMPL	RC	0	Current Transfer status. 1: 1 or more S2MM transfer completed between previous and current S2MM_STS read. A read cycle to this register bit clears the bit and deasserts the IRQ pin. **Note:** A S2MM Transfer is referring to 1 REQUEST bit set.
15:11	RSVD	RO	0	Reserved
10	AXI_DEC_ERR	RC	0	AXI-MM Read RESP = DEC_ERR 1: 1 or more AXI-MM Write return DEC_ERR between previous and current S2MM_STS read. 0: No Error. A read cycle to this register bit clears the bit.
9	AXI_SLV_ERR	RC	0	AXI-MM Read RESP = SLV_ERR 1: 1 or more AXI-MM Write return SLV_ERR between previous and current S2MM_STS read. 0: No Error. A read cycle to this register bit clears the bit.
8	BD_LEN_ERR	RC	0	Descriptor Buffer Length error. 1: 1 or more Descriptor's Buffer with 0 Length detected between previous and current S2MM_STS read. 0: No Error. A read cycle to this register bit clears the bit.
7:1	RSVD	RO	0	Reserved
0	STATUS	RO	0	S2MM DMA operation status 1: DMA operation in progress 0: DMA in IDLE state, no operation pending.

5.1.7. S2MM_CURDESC

Table 5.8. S2MM_CURDESC – Memory Map to Streaming Current Descriptor Pointer

Field	Name	Access	Default	Description
31:0	S2MM_CURDESC	RW	0	Address pointer to Current S2MM Descriptor

5.2. Buffer Descriptor

The SGDMA Controller buffer descriptor is made up of four 32-bit base word fields to describe the memory/buffer address, buffer transfer size, AXI4-Stream attributes, and status field.

5.2.1. MM2S (TX) Descriptor Field

Table 5.9. MM2S – Memory Map to Streaming Descriptor Field

<table>
<tr><th>Offset Bits</th><th>31 24</th><th colspan="2">23 16</th><th>15 8</th><th>7 0</th></tr>
<tr><td>0x00 (Address)</td><td colspan="5">BUFFER_ADDR[31:0]</td></tr>
<tr><td>0x04 (Address)</td><td colspan="5">BUFFER_MSB_ADDR[63:32]</td></tr>
<tr><td>0x08 (Control)</td><td>NXT|FP|RSV|ARPROT</td><td>TID</td><td>TDEST</td><td colspan="2">BUFFER_SIZE</td></tr>
<tr><td>0x0C (Status)</td><td>CMPL| ERRs |</td><td colspan="2">RSVD</td><td colspan="2">TRANSFERRED_SIZE</td></tr>
</table>

5.2.1.1. MM2S_ADDR (Offset 0x00)

Offset 0x00-0x03 is a 32-bit buffer address. The buffer address must always be aligned with the AXI4-MM data width.

Table 5.10. MM2S_ADDR – Memory Map to Streaming Address

Field	Name	Description
31:0	BUFFER_ADDR	The BUFFER_ADDR field specifies the starting address for data transfer from the AXI-MM based memory to the AXI-Streaming interfaced output. This address must be aligned with the AXI-MM data width: • For 32-bit AXI4-MM: Bits [1:0] must be 0 • For 64-bit AXI4-MM: Bits [2:0] must be 0 • For 128-bit AXI4-MM: Bits [3:0] must be 0 You must ensure that the BUFFER_ADDR with BUFFER_SIZE selected must fall within the 4 kB address boundary as per the AMBA AXI4 protocol specification.

5.2.1.2. MM2S_MSB_ADDR (Offset 0x04)

Offset 0x04-0x07 is the MSB 32-bit buffer address if the AXI4-MM address width is 64 bits.

Table 5.11. MM2S_MSB_ADDR – Memory Map to Streaming MSB Address

Field	Name	Description
31:0	BUFFER_MSB_ADDR	Upper 32 bits Buffer Address location to transfer data from Memory Map to Streaming Controller if 64 bits addressing is supported. This DW is ignored if 32 bits addressing is selected.

5.2.1.3. MM2S_CONTROL (Offset 0x08)

This register offset to configure MM2S operation attribute.

Table 5.12. MM2S_CONTROL – Memory Map to Streaming Control

Field	Name	Description
31	NXT	Indicates there is continuous next descriptor in sequence. 1: There is next descriptor. 0: Current descriptor is the last descriptor When set to 1, the DMA fetches the next Descriptor from current descriptor address + 0x10.
30	FP	Indicates if current transfer requires to prefetch all data from Memory into MM2S Buffer before sending to AXI-Stream Controller. 1: Requires Memory data prefetch into MM2S Buffer. 0: Not requires, AXI Stream can deassert tvalid when no data. When set to 1, System application need to ensure the BUFFER_SIZE must not exceed MM2S_FIFO_SIZE
29:27	RSVD	Reserved
26:24	ARPROT	Provides AXI ARPROT information for MM2S data buffering. ARPROT shall be static for the entire packet.
23:20	TID	Provides AXIS TX TID Information for data streaming. TID must stay constant for the entire packet.
19:16	TDEST	Provides AXIS TX TDEST Information for data streaming. TDEST must stay constant for the entire packet.
12:0	BUFFER_SIZE	The BUFFER_SIZE field indicates the total number of bytes to transfer from memory to the Streaming Controller. Valid values range from 1 to 4096; a value of 0 is invalid and results in an error. The MM2S DMA completes the transfer based on the value set in this field.

5.2.1.4. MM2S_STATUS (Offset 0x0C)

This register is offset to indicate the MM2S operation status.

Table 5.13. MM2S_STATUS – Memory Map to Streaming Status

Field	Name	Description
31	CMPL	MM2S DMA Core writes to this bit to indicate Current Descriptor was processed. Processor can recycle this Memory space.
30	BD_LEN_ERR	MM2S DMA Core writes to this bit to indicate Buffer Descriptor BUFFER_SIZE set to 0 Byte.
29	TRANSFERRED_LEN_ERR	MM2S DMA Core writes to this bit to indicate actual transfer size is not aligned with Buffer Descriptor BUFFER_SIZE.
28	AXI_SLVERR	MM2S DMA Core writes to this bit to indicate AXI MM SLVERR
27	AXI_DECERR	MM2S DMA Core writes to this bit to indicate AXI MM DECERR
26:13	RSVD	Reserved
12:0	TRANSFFERED_SIZE	Indicates total data bytes being transferred from Memory to Streaming Controller. Valid range is 1–4096 bytes.

5.2.2. S2MM (RX) Descriptor Field

Table 5.14. S2MM – Streaming to Memory Map Descriptor Field

<table>
<tr><th>Offset Bits</th><th>31 24</th><th>23 16</th><th>15 8</th><th>7 0</th></tr>
<tr><td>0x00 (Address)</td><td colspan="4">BUFFER_ADDR[31:0]</td></tr>
<tr><td>0x04 (Address)</td><td colspan="4">BUFFER_MSB_ADDR[63:32]</td></tr>
<tr><td>0x08 (Control)</td><td>NXT|RSVD| AWPROT</td><td>RSVD</td><td colspan="2">BUFFER_SIZE</td></tr>
<tr><td>0x0C (Status)</td><td colspan="2">CMPL| ERRs | RSVD</td><td colspan="2">TRANSFERRED_SIZE</td></tr>
</table>

5.2.2.1. S2MM_ADDR (Offset 0x00)

Offset 0x00-0x03 is a 32-bit buffer address. The buffer address must always be aligned with the AXI4-MM data width.

Table 5.15. S2MM_ADDR – Streaming to Memory Map Address

Field	Name	Description
31:0	BUFFER_ADDR	The BUFFER_ADDR field specifies the starting address for data transfer from the AXI-S interface to the AXI-MM interface. This address must be aligned with the AXI-MM data width: • For 32-bit AXI4-MM: Bits [1:0] must be 0 • For 64-bit AXI4-MM: Bits [2:0] must be 0 • For 128-bit AXI4-MM: Bits [3:0] must be 0 Users must ensure that the BUFFER_ADDR with BUFFER_SIZE selected must fall within the 4 kB address boundary as per the AMBA AXI4 protocol specification.

5.2.2.2. S2MM_MSB_ADDR (Offset 0x04)

Offset 0x04-0x07 is the MSB 32-bit buffer address if the AXI4-MM address width is 64 bits.

Table 5.16. S2MM_MSB_ADDR – Streaming to Memory Map MSB Address

Field	Name	Description
31:0	BUFFER_MSB_ADDR	Upper 32 bits Buffer Address location to transfer data from Memory Map to Streaming Controller if 64 bits addressing is supported. This DW is ignored if 32 bits addressing is selected.

5.2.2.3. S2MM_CONTROL (Offset 0x08)

This register offset is used to configure the S2MM operation attribute.

Table 5.17. S2MM_CONTROL – Streaming to Memory Map Control

Field	Name	Description
31	NXT	Indicates there is continuous next descriptor in sequence. 1: There is next descriptor. 0: Current descriptor is the last descriptor When set to 1, DMA fetches the next Descriptor from current descriptor address + 0x10.
30:27	RSVD	Reserved
26:24	AWPROT	Provides AXI AWPROT information for S2MM data buffering. AWPROT is static for the entire packet.
23:16	RSVD	Reserved

Field	Name	Description
12:0	BUFFER_SIZE	The BUFFER_SIZE field specifies the total number of bytes to transfer from the Streaming Controller to memory. Valid values range from 1 to 4096; a value of 0 is invalid and will trigger an error condition. The actual transfer size may be less than or equal to BUFFER_SIZE. If it exceeds this value, the additional data will be dropped. To ensure no data gets dropped, the TKEEP signal (rx_axis_s2mm_tkeep_i) must correctly indicate the number of valid bytes during the final beat when TLAST signal is asserted.

5.2.2.4. S2MM_STATUS (Offset 0x0C)

This register offset is used to indicate S2MM operation status.

Table 5.18. S2MM_STATUS – Streaming to Memory Map Status

Field	Name	Description
31	CMPL	S2MM DMA Core writes to this bit to indicate Current Descriptor is processed. Processor can recycle this Memory space.
30	BD_LEN_ERR	S2MM DMA Core writes to this bit to indicate Buffer Descriptor BUFFER_SIZE set to 0 Byte.
29	TRANSFERRED_LEN_ERR	S2MM DMA Core writes to this bit to indicate AXI-S Controller stream more data bytes than defined in Buffer Descriptor BUFFER_SIZE.
28	AXI_SLVERR	S2MM DMA Core writes to this bit to indicate AXI MM SLVERR
27	AXI_DECERR	S2MM DMA Core writes to this bit to indicate AXI MM DECERR
26:13	RSVD	Reserved
12:0	TRANSFFERED_SIZE	Indicates total data bytes being transferred from Streaming Controller to Memory. Valid range is 1 to 4096 bytes.

6. Example Design

The SGDMA Controller IP Core Example Design provides a comprehensive reference implementation that demonstrates efficient data transfer operations between AXI4-Memory Mapped (AXI4-MM) and AXI4-Stream (AXI4-S) interfaces through the integrated SGDMA Controller IP. This example design includes a collection of sgdma_ed_* RTL components that emulate SGDMA driver behavior, memory modules, and AXI interface transactions for comprehensive simulation and testing purposes, though these components are intended purely for functional simulation purposes rather than production-ready modules.

The SGDMA Controller IP Core Example Design enables users to simulate and validate the functionality of various IP configuration options targeting Lattice FPGA devices before integrating the IP into larger system designs. This reference implementation serves as a proven foundation for evaluating IP functionality, ensuring confident integration into production systems.

6.1. Example Design Supported Configuration

The SGDMA Controller IP example design supports all parameter configurations available in the IP user interface.

Table 6.1. SGDMA Controller IP Configuration Supported by the Example Design

SGDMA Controller IP User Interface Parameter	Selectable Value	Supported in Example Design
XCLK_ENABLE	0,1	✓
AXI_DWIDTH	32, 64, 128	✓
AXI_AWIDTH	32, 64	✓
AXI_ID_WIDTH	1, 2, 3, 4, 5, 6, 7, 8	✓
TDATA_WIDTH	8, 16, 32, 64, 128	✓
TDEST_WIDTH	1, 2, 3, 4	✓
TID_WIDTH	1, 2, 3, 4	✓
MM2S_FIFO_DEPTH	512, 1024, 2048, 4096	✓
S2MM_FIFO_DEPTH	512, 1024, 2048, 4096	✓
AXIL_DWIDTH	32	✓
AXIL_AWIDTH	32	✓
BD_DWIDTH	32	✓
BD_AWIDTH	32	✓

6.2. Overview of Example Design and Features

Upon SGDMA Controller IP generation, the example design simulation files are generated under the project folder listed below.

Table 6.2. Example Design Component

Component	Description	File location
tb_top	SGDMA Example Design Top level file	<Project>/testbench
sgdma_ed_mem	Example AXI4-MM memory component to store data transfer and BD memory.	<Project>/testbench
sgdma_ed_axil_m	Example AXI4-Lite manager to initiate SGDMA CSR cycle.	<Project>/testbench
sgdma_ed_axis_tx	Example AXI4-S Tx to stream data	<Project>/testbench
sgdma_ed_axis_rx	Example AXI4-S Rx to receive data and compare with Tx Stream data.	<Project>/testbench
sgdma_ed_apb_m	Example APB manager to initiate SGDMA CSR cycle.	<Project>/testbench

Upon system reset release, APB or AXI4-Lite based Control and Status Register Manager module begins configuring the SGDMA IP Core by writing to the control and status registers, specifically targeting the S2MM (Stream-to-Memory-Mapped) pathway to initiate data transfer operations. The SGDMA IP core automatically retrieves Buffer Descriptors from the Example Design's dedicated memory space, where these descriptors have been pre-configured with the necessary transfer parameters including source addresses, destination addresses, and data lengths. Simultaneously, the Example Design's AXI-Stream Transmitter module begins generating and streaming test data patterns through the TDATA signal lines directly to the SGDMA IP core's stream interface. The SGDMA core then processes this incoming stream data and writes it to the Example Design's AXI Memory-Mapped interface, continuing this write process until the complete data payload has been transferred as specified by the Buffer Descriptor configuration.

When the S2MM transfer operation reaches completion, the SGDMA IP core asserts the s2mm_irq interrupt signal to notify the system that the first phase of data movement has finished successfully. At this point, Control and Status Register Manager module configures the SGDMA IP core's MM2S (Memory-Mapped-to-Stream) control registers to initiate the reverse data flow operation. The SGDMA IP core now reads the previously stored data from the exact same memory locations within the Example Design's AXI Memory-Mapped interface and streams this data out through its AXI-Stream interface to the Example Design's AXI-Stream Receiver module. The AXI-Stream Receiver performs real-time data integrity verification by comparing each incoming TDATA transaction from the SGDMA IP core against the original data patterns that were transmitted earlier during the S2MM phase, immediately asserting the compare_fail signal if any data corruption or mismatch is detected during this critical verification process. The simulation framework includes a configurable NUM_LOOP parameter that allows this complete bidirectional data transfer and verification sequence to be repeated multiple times for comprehensive stress testing and reliability validation.

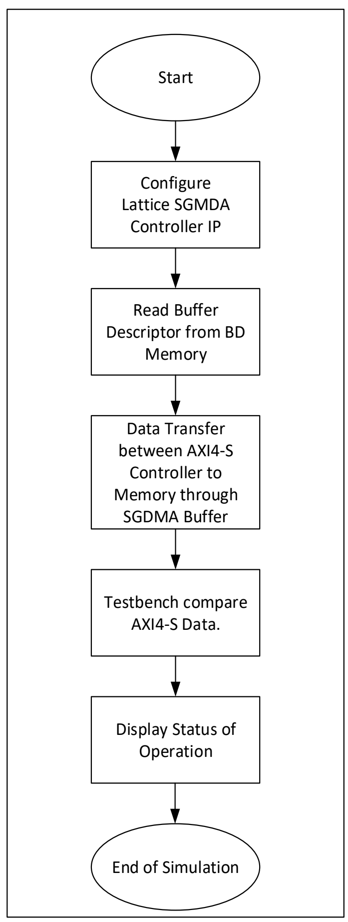

Figure 6.1. SGDMA Controller Example Design Simulation Flow

Both Control and Status Register interfaces (AXI4-Lite and APB) are fully supported within this example design. The Example Design testbench also demonstrates advanced multi-clock domain support capabilities when the XCLK_ENABLE parameter is activated, creating independent clock domains where the AXI-Stream Transmitter and AXI-Stream Receiver modules operate at 125 MHz frequency while the AXI Memory-Mapped interface runs at 200 MHz for Lattice Avant device families or 100 MHz for other supported device families, effectively showcasing the SGDMA IP core's ability to handle clock domain crossings that are common in real-world system implementations. The Control and Status Register access pathway is similar regardless of whether the XCLK_ENABLE parameter is active or inactive.

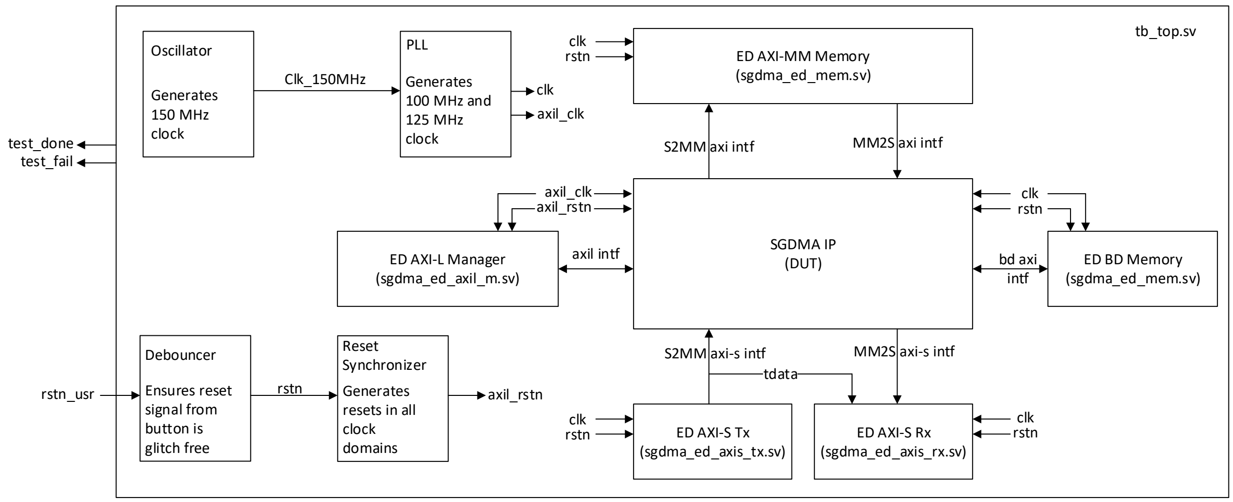

Figure 6.2. SGDMA Controller Example Design Block Diagram with Separate AXI-S Transmitter and Receiver Clock Domains Disabled and AXI-4Lite CSR Access Interface Selected

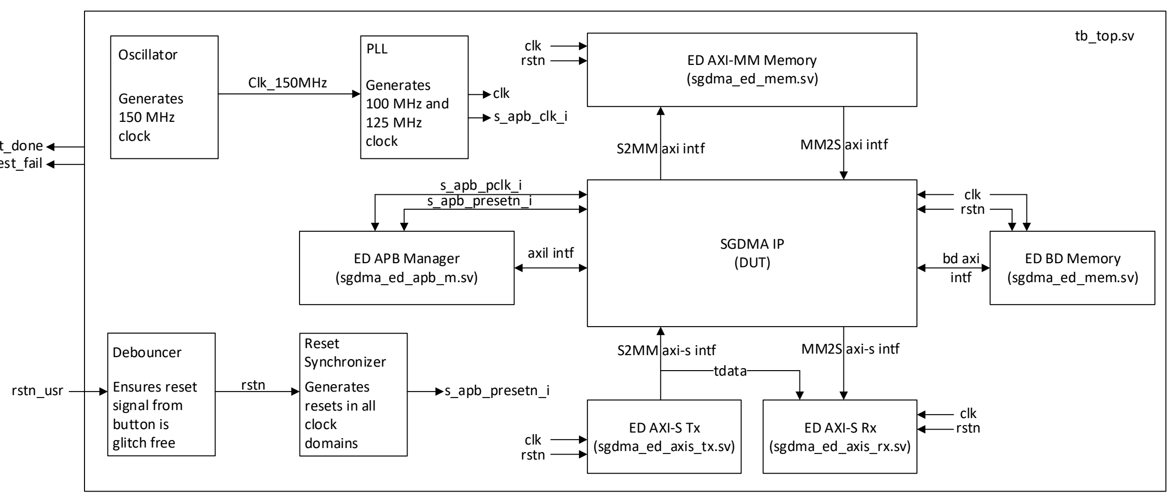

Figure 6.3. SGDMA Controller Example Design Block Diagram with Separate AXI-S Transmitter and Receiver Clock Domains Disabled and APB CSR Access Interface Selected

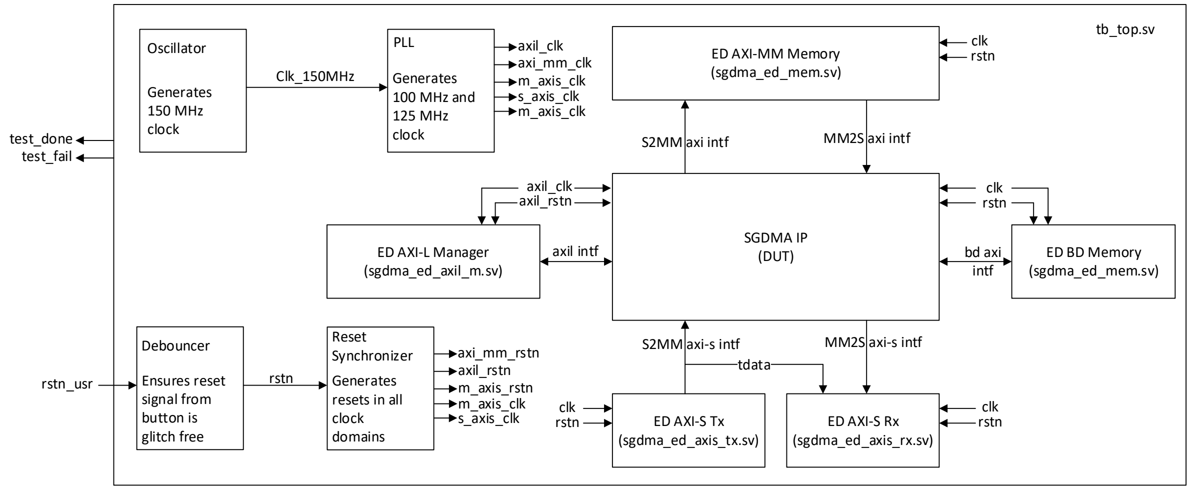

Figure 6.4. SGDMA Controller Example Design Block Diagram with Separate AXI-S Transmitter and Receiver Clock Domains Enabled and AXI-4Lite CSR Access Interface Selected

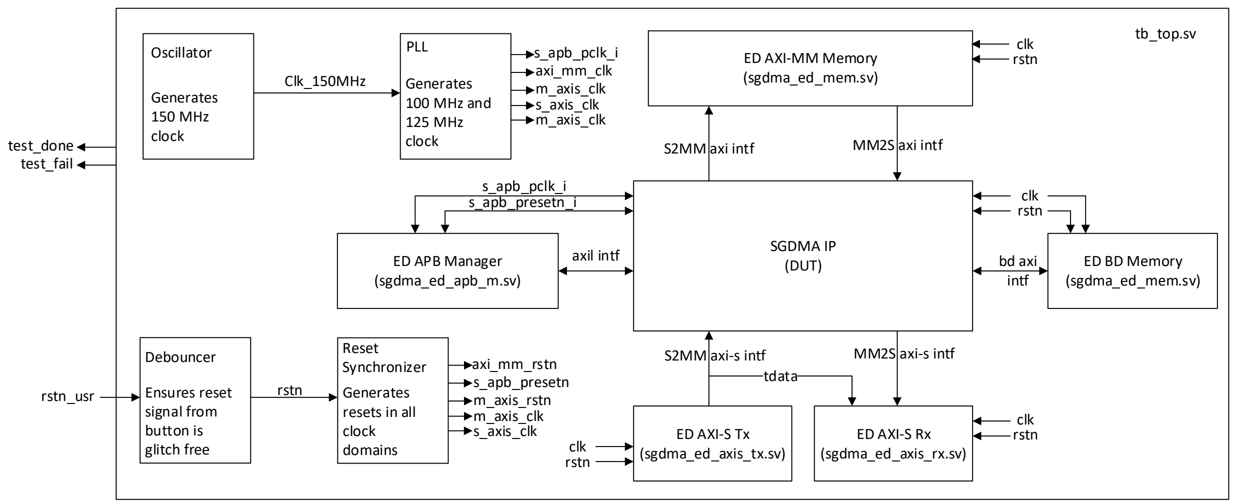

Figure 6.5. SGDMA Controller Example Design Block Diagram with Separate AXI-S Transmitter and Receiver Clock Domains Enabled and APB CSR Access Interface Selected

6.3. Simulating the Example Design

6.3.1. Running Functional Simulation

You can run functional simulations after the IP is generated. QuestaSim OEM simulator is supported starting Radiant 2024.1. Modelsim OEM simulator is supported by older Radiant (2023.2 and earlier).

To run the functional simulations, perform the following:

1. Click the  button located on the **Toolbar** to initiate the **Simulation Wizard** as shown in Figure 6.6.

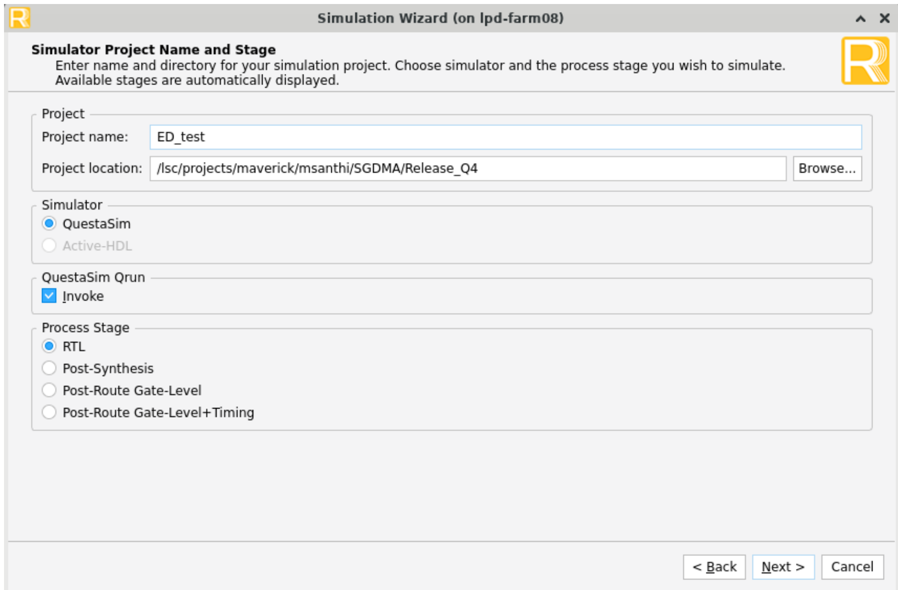

Figure 6.6. Simulation Wizard

2. Click **Next** to open the **Add and Reorder Source** window as shown in Figure 6.7.

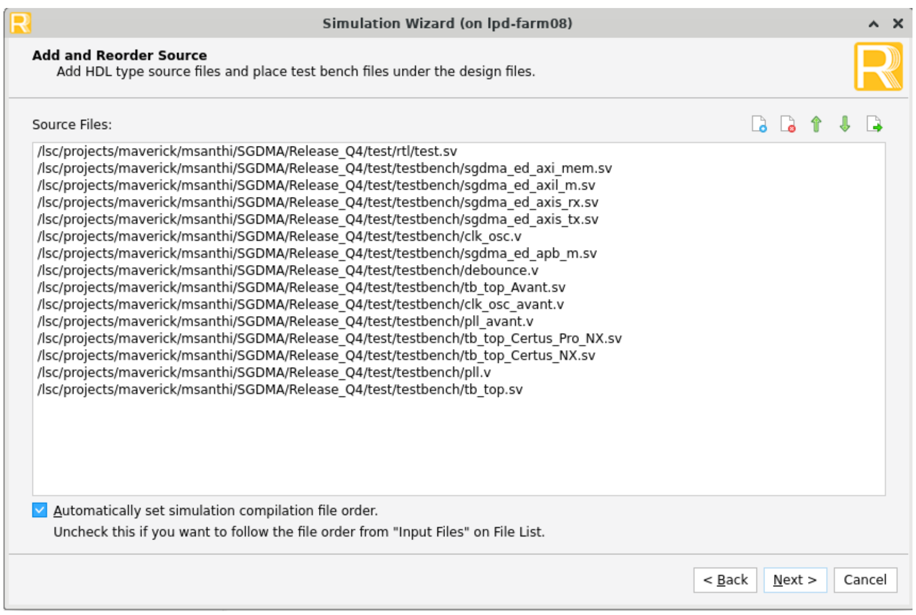

Figure 6.7. Add and Reorder Source

3. Click **Next** to open **Parse HDL files for simulation**. Select **tb_top** as the Simulation Top module.

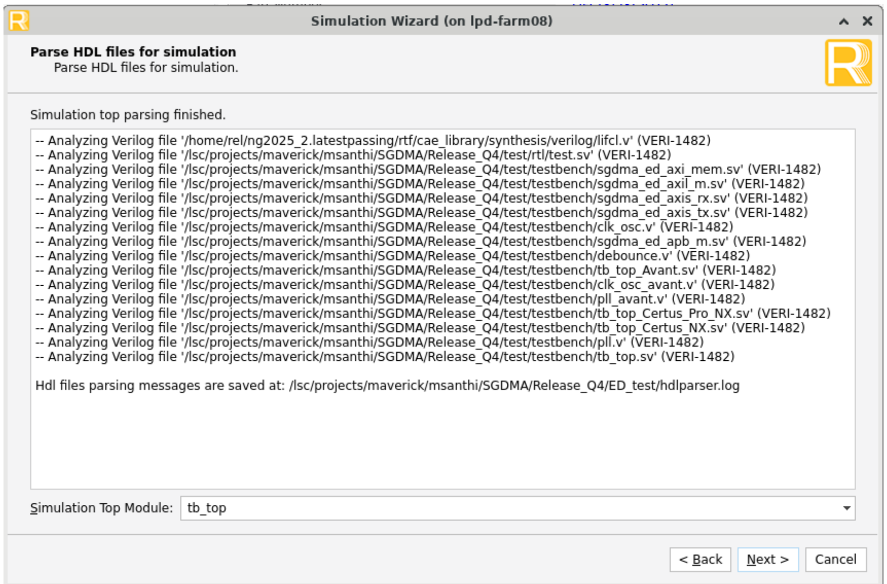

Figure 6.8. Select Simulation Top Module

4. Click **Next**. The **Summary** window is shown. You can set the Default run to 0ns if you'd like the simulation to run until it achieves its break condition.

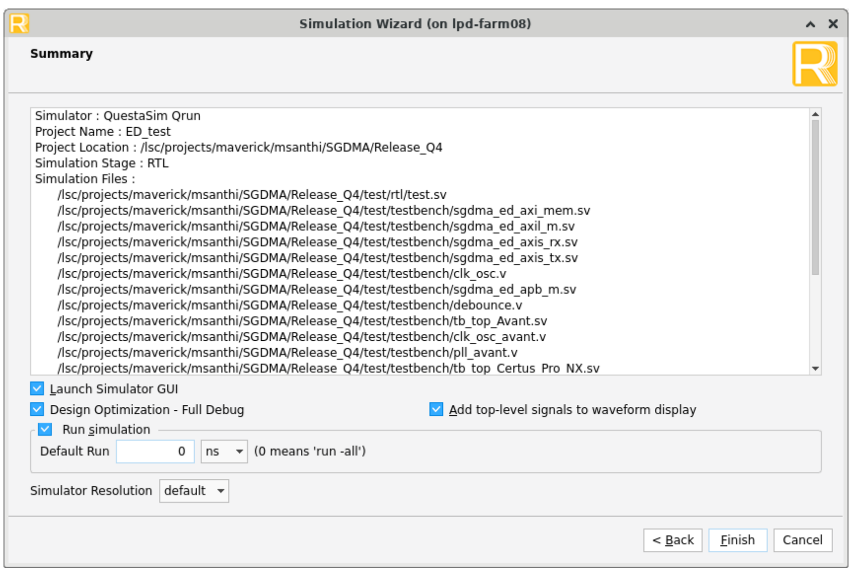

Figure 6.9. Summary Window for Example Design Simulation

5. Click **Finish** to run the simulation. Figure 6.10 shows an example simulation result.

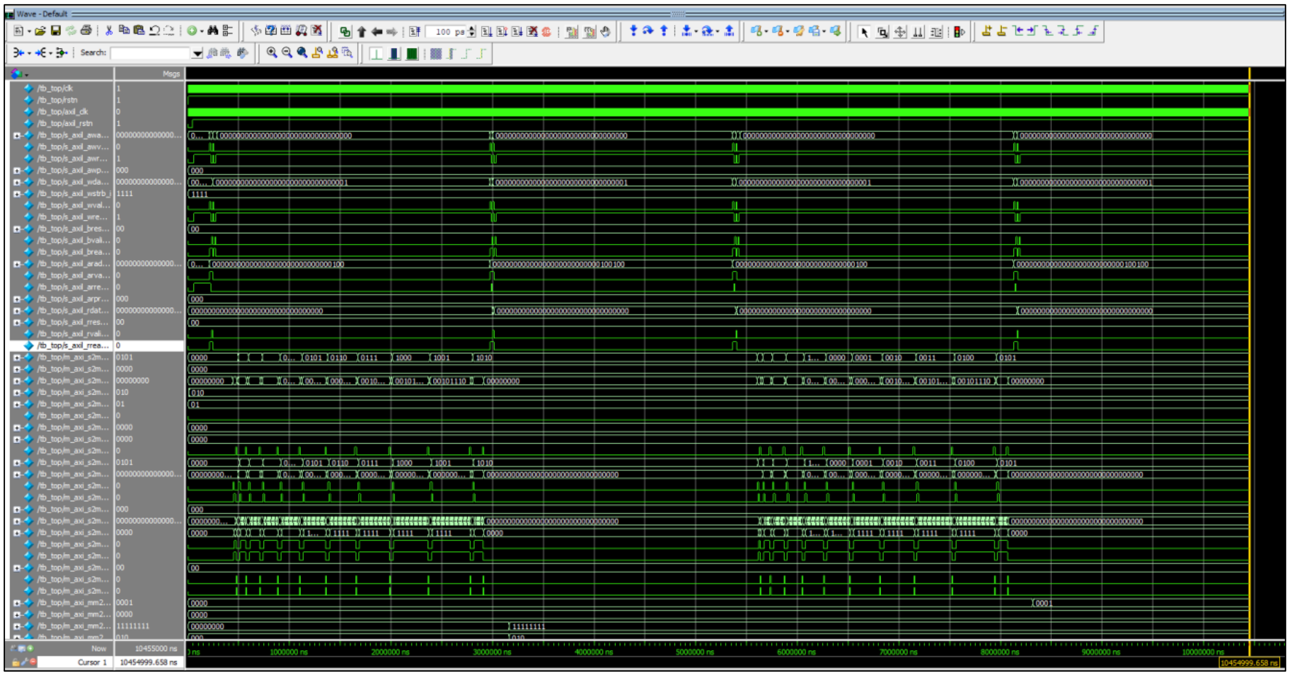

Figure 6.10. Simulation Waveform

6.3.2. Simulation Results

6.3.2.1. Successful Test Run

During a successful test execution, the simulation completes without triggering either the data_mismatch or test_fail output signals, indicating that all data integrity verification checks have passed successfully. The test_count output pin displays the total number of complete S2MM-to-MM2S data transfer cycles that have been executed during the simulation run, with this example showing two complete cycles since test_count operates as a zero-based counter signal. You can customize the number of transfer cycles by modifying the NUM_LOOP localparam setting within the tb_top.sv testbench file, which supports configuration values ranging from 1 to 15 cycles to accommodate different testing requirements and thoroughness levels.

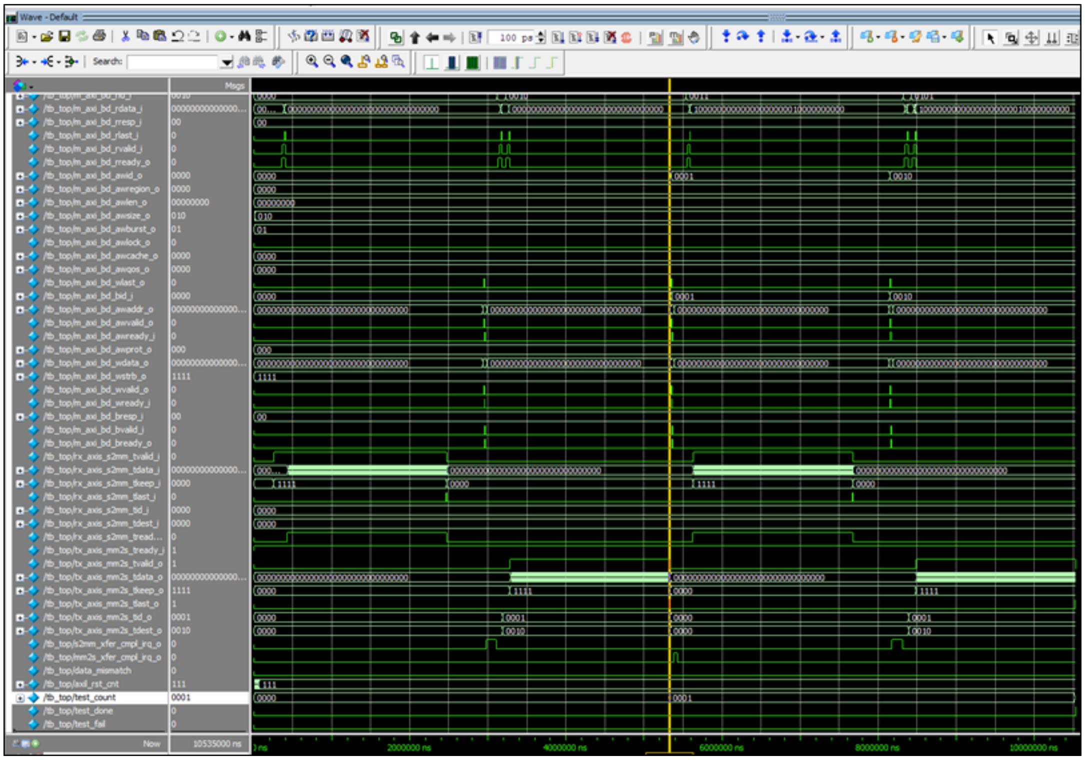

Figure 6.11 Passing Simulation Waveform

The simulation message box showed *Simulation Passed*.

```
# ----------------------------------------
# ------------- SIMULATION PASSED -------------
# ----------------------------------------
# ** Note: $stop    : /lsc/scratch/sdc/kcheah/my_design/sgdma200/sgdma200/testbench/tb_top.sv(482)
#    Time: 10535 us  Iteration: 2  Instance: /tb_top
# Break in Module tb_top at /lsc/scratch/sdc/kcheah/my_design/sgdma200/sgdma200/testbench/tb_top.sv line 482

VSIM 6>
```

Figure 6.12. Passing Simulation Log

6.3.2.2. Failed Test Run

In a failing scenario where AXI4-S Transmit TDATA mismatches with AXI4-S Receive TDATA, the simulation flags through data_mismatch and test_fail pin.

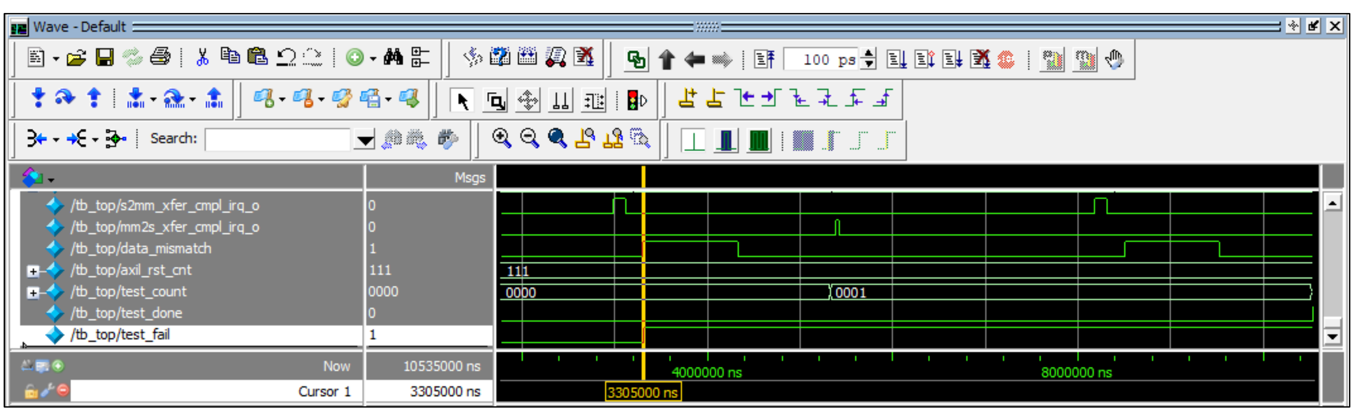

Figure 6.13. Failing Simulation Waveform

The simulation message box showed *Test Failed* and data mismatched time.

```
# [Error] Data Mismatch at Time:          3300000000
# [Error] Data Mismatch at Time:          8500000000
# ---------------------------------------
# ------------- Test Failed -------------
# ---------------------------------------
# ** Note: $stop    : C:/Users/wng/my_designs/sgdma_ed/sgdmal/testbench/tb_top.sv(476)
#    Time: 10535 us  Iteration: 2  Instance: /tb top
```

Figure 6.14. Failing Simulation Log

7. Designing with the IP

This section provides information on how to generate the IP Core using the Lattice Radiant software and how to run simulation and synthesis. For more details on the Lattice Radiant software, refer to the Lattice Radiant Software User Guide.

Note: The screenshots provided are for reference only. Details may vary depending on the version of the IP or software being used. If there have been no significant changes to the GUI, a screenshot may reflect an earlier version of the IP.

7.1. Generating and Instantiating the IP

You can use the Lattice Radiant software to generate IP modules and integrate them into the device architecture. The steps below describe how to generate the SGDMA Controller IP in the Lattice Radiant software.

To generate the SGDMA Controller IP:

1. Create a new Lattice Radiant software project or open an existing project.
2. In the **IP Catalog** tab, double-click **SGDMA Controller** under the **IP, Processors_Controllers_and_Peripherals** category. The **Module/IP Block Wizard** opens as shown in Figure 7.1. Enter values in the **Component name** and the **Create in** fields and click **Next**.

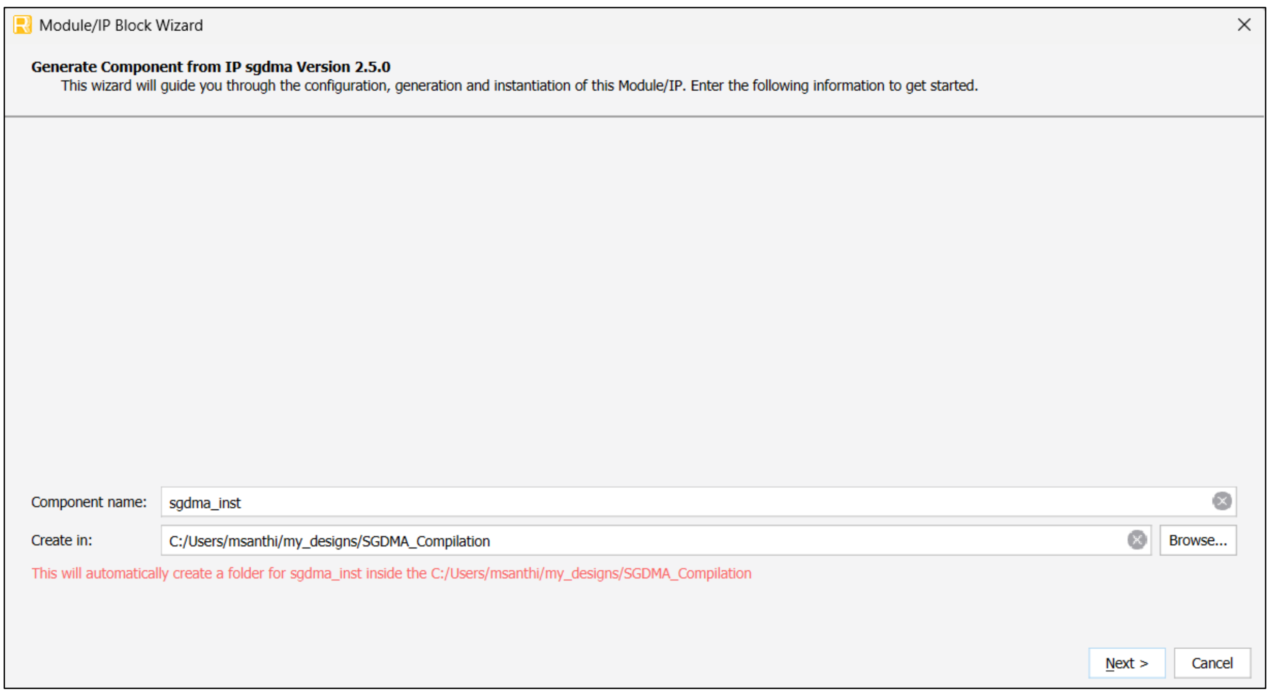

Figure 7.1. Module/IP Block Wizard

3. In the next **Module/IP Block Wizard** window, customize the selected SGDMA Controller IP using drop-down lists and check boxes. Figure 7.2. IP Configuration shows an example configuration of the SGDMA Controller IP. For details on the configuration options, refer to the IP Parameter Description section.

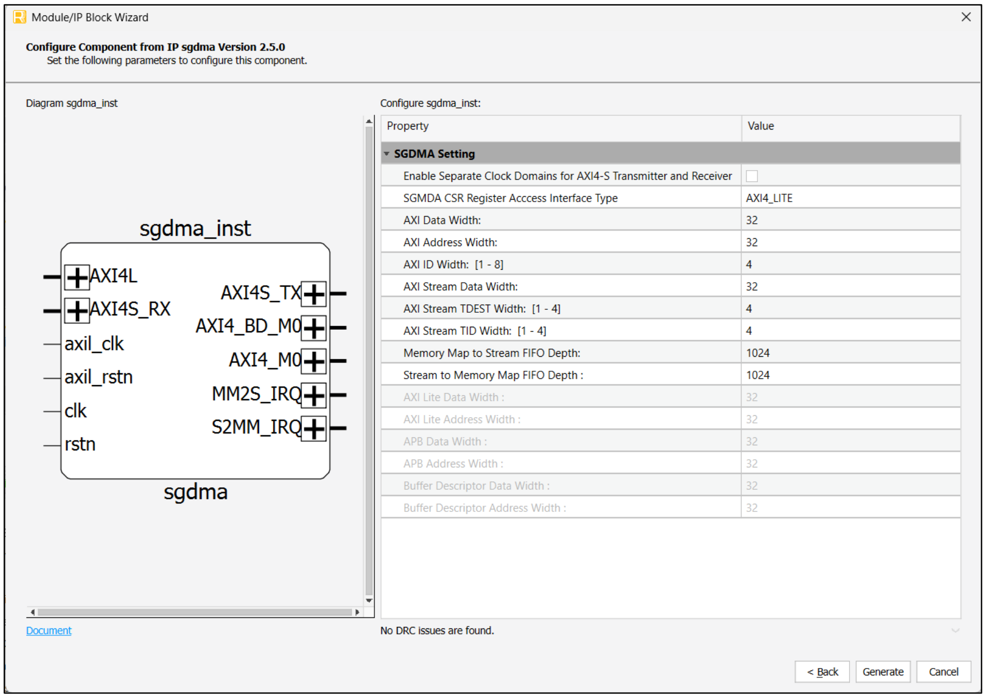

Figure 7.2. IP Configuration

4. Click **Generate**. The **Check Generating Result** dialog box opens, showing design block messages and results as shown in Figure 7.3.

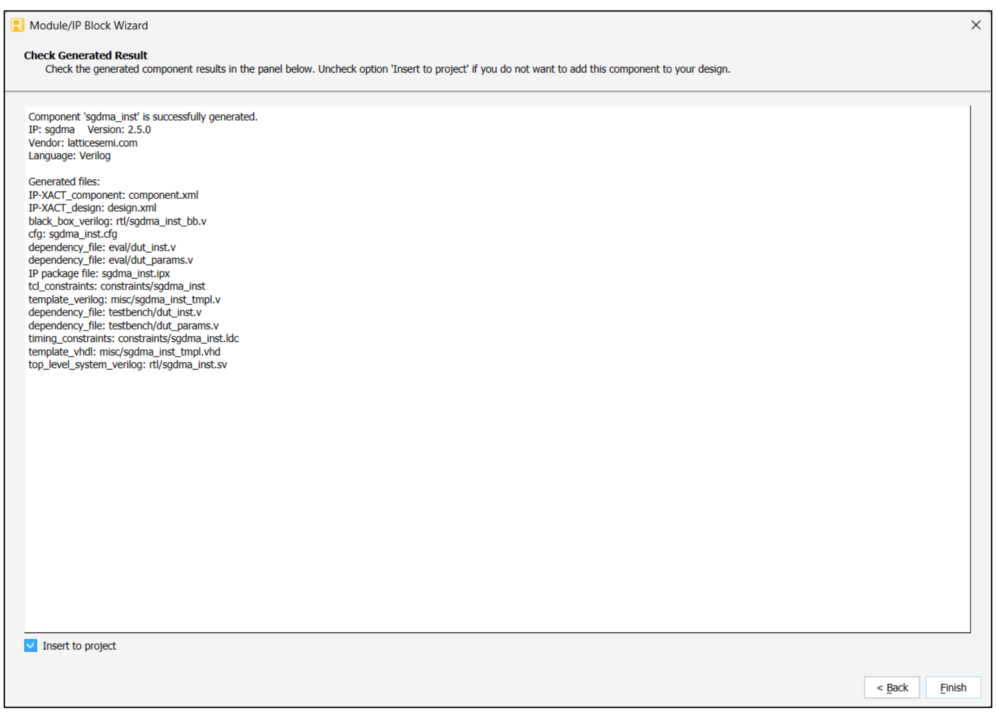

Figure 7.3. Check Generated Result

5. Click **Finish**. All the generated files are placed under the directory paths in the **Create in** and the **Component name** fields shown in Figure 7.1.

7.1.1. Generated Files and File Structure

The generated SGDMA Controller module package includes the black box (<Component name>_bb.v) and instance templates (<Component name>_tmpl.v/vhd) that can be used to instantiate the core in a top-level design. An example RTL top-level reference source file (<Component name>.v) that can be used as an instantiation template for the module is also provided. You may also use this top-level reference as the starting template for their complete design. The generated files are listed in Table 7.1.

Table 7.1. Generated File List

Attribute	Description
<Component name>.ipx	This file contains the information on the files associated to the generated IP.
<Component name>.cfg	This file contains the parameter values used in IP configuration.
component.xml	Contains the ipxact:component information of the IP.
design.xml	Documents the configuration parameters of the IP in IP-XACT 2014 format.
rtl/<Component name>.v	This file provides an example RTL top file that instantiates the module.
rtl/<Component name>_bb.v	This file provides the synthesis black box.
misc/<Component name>_tmpl.v misc /<Component name>_tmpl.vhd	These files provide instance templates for the module.

7.2. Design Implementation

Completing your design includes additional steps to specify analog properties, pin assignments, timing, and physical constraints. You can add and edit the constraints using the Device Constraint Editor or by manually creating a PDC file.

Post-Synthesis constraint files (.pdc) contain both timing and non-timing constraint .pdc source files for storing logical timing and physical constraints. Constraints that are added using the Device Constraint Editor are saved to the active .pdc file. The active post-synthesis design constraint file is then used as input for post-synthesis processes.

Refer to the relevant sections in the latest Lattice Radiant User Guide in the Lattice Radiant web page for more information on how to create or edit constraints and how to use the Device Constraint Editor.

7.3. Timing Constraints

The timing constraints are based on the clock frequency used. The timing constraints for the IP are defined in the relevant constraint files. The example below shows the IP timing constraints generated for the SGDMA Controller IP. The constraint file is available in *<Project>/constraints/constraint.sdc* and *<Project>/ldc/constraint.ldc*. Both files are automatically added during IP generation.

```
if { $family == "LAV-AT" && $XCLK_ENABLE == "ENABLE"} {

    create_clock -name {axi_mm_clk} -period 5 [get_ports axi_mm_clk]
    create_clock -name {m_axis_clk} -period 8 [get_ports m_axis_clk]
    create_clock -name {s_axis_clk} -period 8 [get_ports s_axis_clk]
    create_clock -name {axil_clk} -period 10 [get_ports axil_clk]
    set_clock_groups -group [get_clocks axi_mm_clk] \
                 -group [get_clocks m_axis_clk] \
                 -group [get_clocks s_axis_clk] \
         -group [get_clocks axil_clk] \
                 -asynchronous


} elseif { $family != "LAV-AT" && $XCLK_ENABLE == "ENABLE"} {

    create_clock -name {axi_mm_clk} -period 10 [get_ports axi_mm_clk]
    create_clock -name {m_axis_clk} -period 8 [get_ports m_axis_clk]
    create_clock -name {s_axis_clk} -period 8 [get_ports s_axis_clk]
    create_clock -name {axil_clk} -period 10 [get_ports axil_clk]
    set_clock_groups -group [get_clocks axi_mm_clk] \
                 -group [get_clocks m_axis_clk] \
                 -group [get_clocks s_axis_clk] \
         -group [get_clocks axil_clk] \
                 -asynchronous

} else {

    create_clock -name {clk} -period 8 [get_ports clk]
    create_clock -name {axil_clk} -period 10 [get_ports axil_clk]
    set_clock_groups -group [get_clocks clk] \
         -group [get_clocks axil_clk] \
                 -asynchronous

}
```

Figure 7.4. Excerpt from Lattice Design Constraints File (.ldc) for the SGDMA Controller IP

```
set_max_delay -from [get_pins */sgdmac_core_inst/sgdma_csr/genblk1.cc_s2mm_cfg_addr/genblk1.din[*].ff_inst/Q] \
              -to [get_pins */sgdmac_core_inst/sgdma_csr/genblk1.cc_s2mm_cfg_addr/genblk1.dout[*].ff_inst/DF] -datapath_only 5
set_max_delay -from [get_pins */sgdmac_core_inst/sgdma_csr/genblk1.cc_mm2s_cfg_addr/genblk1.din[*].ff_inst/Q] \
              -to [get_pins */sgdmac_core_inst/sgdma_csr/genblk1.cc_mm2s_cfg_addr/genblk1.dout[*].ff_inst/DF] -datapath_only 5
```

Figure 7.5. Excerpt from SDC Constraints File (.sdc) for the SGDMA Controller IP

7.4. Physical Constraints

7.4.1. Compiling Standalone IP only

When compiling SGDMA Controller IP Core standalone without connecting to other IPs, setting SGDMA Controller I/O Ports to virtual ports is required to avoid place and route errors. The constraint file is available in <Project>/eval/constraint.pdc,

```
ldc_set_attribute {VIRTUAL_IO=TRUE} [get_ports {rx_axis_s2mm_*_i[*]}]
ldc_set_attribute {VIRTUAL_IO=TRUE} [get_ports {rx_axis_s2mm_tkeep_i[*]}]
ldc_set_attribute {VIRTUAL_IO=TRUE} [get_ports {rx_axis_s2mm_*_i}]
ldc_set_attribute {VIRTUAL_IO=TRUE} [get_ports {m_axi_s2mm_*_i[*]}]
ldc_set_attribute {VIRTUAL_IO=TRUE} [get_ports {m_axi_s2mm_*_i}]
ldc_set_attribute {VIRTUAL_IO=TRUE} [get_ports {tx_axis_mm2s_*_i}]
ldc_set_attribute {VIRTUAL_IO=TRUE} [get_ports {tx_axis_mm2s_*_o[*]}]
ldc_set_attribute {VIRTUAL_IO=TRUE} [get_ports {m_axi_mm2s_*_o[*]}]
ldc_set_attribute {VIRTUAL_IO=TRUE} [get_ports {m_axi_mm2s_*_i[*]}]
ldc_set_attribute {VIRTUAL_IO=TRUE} [get_ports {m_axi_mm2s_*_i}]
ldc_set_attribute {VIRTUAL_IO=TRUE} [get_ports {s_axil_*_i[*]}]
ldc_set_attribute {VIRTUAL_IO=TRUE} [get_ports {s_axil_*_i}]
ldc_set_attribute {VIRTUAL_IO=TRUE} [get_ports {m_axi_bd_*_i[*]}]
ldc_set_attribute {VIRTUAL_IO=TRUE} [get_ports {m_axi_bd_*_i}]
ldc_set_attribute {VIRTUAL_IO=TRUE} [get_ports {m_axi_bd_araddr_o[*]}]
ldc_set_attribute {VIRTUAL_IO=TRUE} [get_ports {m_axi_bd_arlen_o[*]}]
ldc_set_attribute {VIRTUAL_IO=TRUE} [get_ports {m_axi_bd_a*_o[*]}]
ldc_set_attribute {VIRTUAL_IO=TRUE} [get_ports {m_axi_bd_wdata_o[*]}]
ldc_set_attribute {VIRTUAL_IO=TRUE} [get_ports {m_axi_bd_wstrb_o[*]}]
```

Figure 7.6. Physical Constraint File (.pdc) for the SGDMA Controller IP

7.5. Specifying the Strategy

The Lattice Radiant software provides two predefined strategies: area and timing. It also enables you to create customized strategies. For details on how to create a new strategy, refer to the Strategies section of the Lattice Radiant Software user guide.

8. Programming Model

The reference driver is available in the *Driver* folder post IP generation. Refer to SGDMA Driver API Reference (FPGA-TN-02340) for more information.

8.1. MM2S DMA Programming

8.1.1. Single Buffer Descriptor per Request (NXT=0)

1. Configure MM2S_CURDESC register in offset 0x08 with the Buffer Descriptor Address Pointer.
2. Configure MM2S_CTRL.REQUEST register bit 0 in offset 0x00 to trigger the start of the MM2S operation.
3. DMA Core updates the Buffer Descriptor Status DW with CMPL bit set and total transferred size. Error bits are set according to the actual scenario.
4. DMA Core asserts the mm2s_xfer_cmpl_irq_o port to indicate data transfer completion for the current Buffer Descriptor.
5. Upon CMPL IRQ detection by the CPU, the system driver performs the following:
    a. Read MM2S_STS.XFER_CMPL register bit 16 in offset 0x04 to deassert the IRQ and other status fields in the same offset.
    b. Read the Buffer Descriptor status described in the Buffer Descriptor section and the CPU can repurpose the Buffer Descriptor memory as necessary.

8.1.2. Multiple Buffer Descriptor per Request (NXT=1)

1. Configure MM2S_CURDESC register in offset 0x08 with the Buffer Descriptor Address Pointer.
2. Configure MM2S_CTRL.REQUEST register bit 0 in offset 0x00 to trigger the start of the MM2S operation.
3. DMA Core updates the Buffer Descriptor Status DW with CMPL bit set and total transferred size. Error bits are set according to the actual scenario.
4. DMA Core asserts the mm2s_xfer_cmpl_irq_o port to indicate data transfer completion after SGDMA services all BD per REQUEST for multiple BD operations.
5. DMA core fetches the next Buffer Descriptor in the sequence, start the next MM2S operation automatically, and update statuses like (3) and (4).
6. Upon CMPL IRQ detection by the CPU, the system driver performs the following:
    a. Read MM2S_STS.XFER_CMPL register bit 16 in offset 0x04 to deassert the IRQ and other status fields in the same offset.
    b. Read the Buffer Descriptor status described in the Buffer Descriptor section and the CPU can repurpose the Buffer Descriptor memory as necessary.

8.2. S2MM DMA Programming

8.2.1. Single Buffer Descriptor per Request (NXT=0)

1. Configure S2MM_CURDESC register in offset 0x28 with the Buffer Descriptor Address Pointer.
2. Configure S2MM_CTRL.REQUEST register bit 0 in offset 0x20 to trigger the start of the S2MM operation.
3. DMA Core updates the Buffer Descriptor Status DW with CMPL bit set and total transferred size. Error bits are set according to the actual scenario.
4. DMA Core asserts the s2mm_xfer_cmpl_irq_o port to indicate data transfer completion for the current Buffer Descriptor.
5. Upon CMPL IRQ detection by the CPU, the system driver performs the following:
    a. Read MM2S_STS.XFER_CMPL register bit 16 in offset 0x04 to deassert the IRQ and other status fields in the same offset.

b. Read Buffer Descriptor status describes in the Buffer Descriptor section and the CPU can repurpose the Buffer Descriptor memory as necessary.

8.2.2. Multiple Buffer Descriptor per Request (NXT=1)

1. Configure S2MM_CURDESC register in offset 0x28 with the Buffer Descriptor Address Pointer.
2. Configure S2MM_CTRL.REQUEST register bit 0 in offset 0x20 to trigger the start of the S2MM operation.
3. DMA Core updates the Buffer Descriptor Status DW with CMPL bit set and total transferred size. Error bits are set according to the actual scenario.
4. DMA Core fetches the next Buffer Descriptor in the sequence, start the next S2MM operation automatically, and update statuses like (3) and (4).
5. DMA core asserts s2mm_xfer_cmpl_irq_o port to indicate data transfer completion after SGDMA services all BD per REQUEST for multiple BD operations.
6. Upon CMPL IRQ detection by the CPU, the system driver performs the following:
   a. Read MM2S_STS.XFER_CMPL register bit 16 in offset 0x04 to deassert the IRQ and other status fields in the same offset.
   b. Read Buffer Descriptor status describes in the Buffer Descriptor section and the CPU can repurpose the Buffer Descriptor memory as necessary.

8.3. MM2S/S2MM DMA Reset

During DMA in IDLE or Normal Operation, system driver can halt and reset the MM2S or S2MM operation.

1. Configure MM2S/S2MM_CTRL.RESET (bit 1 in offset 0x00/0x20).
2. Respective DMA halts operation, reset the state machine, flush all pending commands and data, reset all MM2S/S2MM configuration, and status registers and deassert the IRQ port.
3. The Buffer Descriptor status is not updated.
4. Refer to the Reset Overview section for more information.

8.4. Error Scenarios

8.4.1. MM2S Errors

- BD_LEN_ERR, AXI_SLV_ERR, AXI_DEC_ERR
  - The first encountered error is reflected in the status register until the user reads the register to clear it.
  - DMA writes back the first encountered BD_LEN_ERR to the Buffer Descriptor. Subsequent BD_LEN_ERR update on the Buffer Descriptor table are not guaranteed by DMA due to invalid buffer length can cause the DMA to enter an undefined behavior.
  - After an error is reported, you must program RESET bit to flush the DMA and then reprogram the DMA to process a new descriptor table.
- XFER_ERR
  - The first encountered error will be reflected in the status register until you read the register to clear it.
  - The user should poll the status register bit to detect any XFER_ERR.
  - After the first XFER_ERR is logged in the DMA register, you must program RESET bit to flush the DMA and then reprogram the DMA to process a new descriptor table.
  - In certain scenarios, this error may not be reflected consistently in the Buffer Descriptor.

8.4.2. S2MM Errors

- BD_LEN_ERR, AXI_SLV_ERR, AXI_DEC_ERR
    - The first encountered error is reflected in the status register until the user reads the register to clear it.
    - DMA writes back the first encountered BD_LEN_ERR to the Buffer Descriptor. Subsequent BD_LEN_ERR update on the Buffer Descriptor table are not guaranteed by DMA due to invalid buffer length can cause the DMA to enter an undefined behavior.
    - After an error is reported, the user must program RESET bit to flush the DMA and then reprogram the DMA to process a new descriptor table.
- XFER_ERR
    - The first encountered error will be reflected in the status register until the user reads the register to clear it.
    - The user should poll the status register bit to detect any XFER_ERR.
    - After the first XFER_ERR is logged in the DMA register, the user must program RESET bit to flush the DMA and then reprogram the DMA to process a new descriptor table.
    - In certain scenarios, this error may not be reflected consistently in the Buffer Descriptor.

Appendix A. Resource Utilization

A sample resource utilization of the SGDMA Controller IP Core is shown in Table A.1 and Table A.3.

Target Device: LFCPNX-100 LFG672 (7-High Performance)

Table A.1. CertusPro-NX Resource Utilization with Additional Clock Domains Disabled and AXI4-Lite interface for CSR Register Access

AXI MM		AXI Streaming	FIFO Buffer Depth		Resource				
DWIDTH	AWIDTH	TDATA	MM2S	S2MM	LUT4 Logic	LUT4 Ripple Logic	PFU Register	I/O Buffers	EBR
32	32	32	1024	1024	2185	620	2075	17	4
32	32	8	1024	1024	2171	620	2062	17	4
32	32	128	1024	1024	2799	604	2440	17	16
128	32	8	1024	1024	2724	610	2390	17	16
32	32	32	512	512	2165	620	2047	17	2
128	32	128	1024	1024	2879	610	2507	17	16
128	32	128	4096	4096	2901	628	2563	17	72
32	32	32	4096	4096	2211	638	2131	17	18
32	64	32	1024	1024	2354	684	2299	17	4
32	64	8	1024	1024	2308	684	2286	17	4
128	64	128	1024	1024	3069	674	2731	17	16

Table A.2. CertusPro-NX Resource Utilization with Additional Clock Domains Enabled and AXI4-Lite interface for CSR Register Access

AXI MM		AXI Streaming	FIFO Buffer Depth		Resource				
DWIDTH	AWIDTH	TDATA	MM2S	S2MM	LUT4 Logic	LUT4 Ripple Logic	PFU Register	I/O Buffers	EBR
32	32	32	1024	1024	2189	630	2384	21	4
32	32	8	1024	1024	2145	630	2371	21	4
32	32	128	1024	1024	2795	614	2739	21	16
128	32	8	1024	1024	2700	620	2689	21	16
32	32	32	512	512	2179	626	2366	21	2
128	32	128	1024	1024	2877	620	2806	21	16
128	32	128	4096	4096	2934	640	2844	21	64
32	32	32	4096	4096	2234	650	2422	21	16
32	64	32	1024	1024	2410	694	2608	21	4
32	64	8	1024	1024	2365	694	2595	21	4
128	64	128	1024	1024	3134	684	3030	21	16

Target Device: LAV-AT-E70-3LFG676C (Performance Grade = 3)

Table A.3. Avant Resource Utilization with Additional Clock Domains Disabled and AXI4-Lite interface for CSR Register Access

AXI MM		AXI Streaming	FIFO Buffer Depth		Resource				
DWIDTH	AWIDTH	TDATA	MM2S	S2MM	LUT4 Logic	LUT4 Ripple Logic	PFU Register	I/O Buffer	EBR
32	32	32	1024	1024	2163	552	2178	17	2
32	32	8	1024	1024	2131	552	2165	17	2
32	32	128	1024	1024	2756	540	2543	17	8
128	32	8	1024	1024	2670	542	2493	17	8
32	32	32	512	512	2173	540	2150	17	2
128	32	128	1024	1024	2848	542	2610	17	8
128	32	128	4096	4096	2851	560	2666	17	32
32	32	32	4096	4096	2214	570	2234	17	8
32	64	32	1024	1024	2331	616	2299	17	2
32	64	8	1024	1024	2341	616	2286	17	2
128	64	128	1024	1024	3080	606	2731	17	8

Table A.4. Avant Resource Utilization with Additional Clock Domains Enabled and AXI4-Lite interface for CSR Register Access

AXI MM		AXI Streaming	FIFO Buffer Depth		Resource				
DWIDTH	AWIDTH	TDATA	MM2S	S2MM	LUT4 Logic	LUT4 Ripple Logic	PFU Register	I/O Register	EBR
32	32	32	1024	1024	2219	536	2491	21	2
32	32	8	1024	1024	2181	536	2475	21	2
32	32	128	1024	1024	2830	532	2853	21	8
128	32	8	1024	1024	2744	526	2802	21	8
32	32	32	512	512	2189	528	2473	21	2
128	32	128	1024	1024	2916	526	2917	21	8
128	32	128	4096	4096	2923	546	2947	21	32
32	32	32	4096	4096	2234	556	2525	21	8
32	64	32	1024	1024	2385	600	2614	21	2
32	64	8	1024	1024	2356	600	2602	21	2
128	64	128	1024	1024	3140	590	3041	21	8

Target Device: LFD2NX-40-9BG196C (Performance Grade = 9)

Table A.5. Certus-NX Resource Utilization with Additional Clock Domains Disabled and AXI4-Lite interface for CSR Register Access

AXI MM		AXI Streaming	FIFO Buffer Depth		Resource				
DWIDTH	AWIDTH	TDATA	MM2S	S2MM	LUT4 Logic	LUT4 Ripple Logic	PFU Register	I/O Buffer	EBR
32	32	32	1024	1024	2206	620	2145	17	4
32	32	8	1024	1024	2193	620	2132	17	4
32	32	128	1024	1024	2803	592	2510	17	16
128	32	8	1024	1024	2723	598	2460	17	16
32	32	32	512	512	2193	620	2117	17	2
128	32	128	1024	1024	2908	598	2577	17	16
128	32	128	4096	4096	2931	614	2633	17	72
32	32	32	4096	4096	2240	638	2201	17	18
32	64	32	1024	1024	2369	684	2369	17	4
32	64	8	1024	1024	2326	684	2356	17	4
128	64	128	1024	1024	3157	662	2801	17	16

Table A.6. Certus-NX Resource Utilization with Additional Clock Domains Enabled and AXI4-Lite interface for CSR Register Access

AXI MM		AXI Streaming	FIFO Buffer Depth		Resource				
DWIDTH	AWIDTH	TDATA	MM2S	S2MM	LUT4 Logic	LUT4 Ripple Logic	PFU Register	I/O Register	EBR
32	32	32	1024	1024	2205	630	2454	21	4
32	32	8	1024	1024	2173	630	2441	21	4
32	32	128	1024	1024	2822	614	2809	21	16
128	32	8	1024	1024	2735	620	2759	21	16
32	32	32	512	512	2185	626	2436	21	2
128	32	128	1024	1024	2905	620	2876	21	16
128	32	128	4096	4096	2938	640	2914	21	64
32	32	32	4096	4096	2243	650	2492	21	16
32	64	32	1024	1024	2426	694	2678	21	4
32	64	8	1024	1024	2378	694	2665	21	4
128	64	128	1024	1024	3162	684	3100	21	16

Note: Resource utilization differ with different configurations of the SGDMA IP. The above resource utilization is provided for reference only. You can view the resource utilization under *Report > Map > Map Resource Usage*. To view the resource usage, you must run the Synthesize and Map Design.

Appendix B. SGDMA Controller Performance

SGDMA Controller IP cycle latency is shown in Table B.1.

Configuration:

- Data Width: 32 bits
- Data Transfer rate: 1024 bytes without backpressure

Table B.1. SGDMA Controller IP Performance

	Task	Number of Cycle (Clk Domain)
Register Programming	From Request bit set to start BD Read	~7-10
Buffer Descriptor Read	BD Read to BD Read completion	4
S2MM Data Transfer	BD read completion to 1st AXIS_TREADY	3
	1st AXIS_TDATA receive to 1st AXI4-MM Write	5
	1st AXIS_TDATA receive to last AXI4-MM Write	313
	From Request bit set to last AXI4-MM Write	324
MM2S Data Transfer	BD read completion to 1st AXI4-MM Read	3
	1st AXI4-MM Read to 1st AXIS_TVALID	7
	1st AXI4-MM Read to last AXIS_TVALID	263
	From Request bit set to last AXIS_TVALID	274

References

For more information refer to:

- SGDMA Controller IP Release Notes (FPGA-RN-02058)
- SGDMA Driver API Reference (FPGA-TN-02340)
- Lattice Radiant Timing Constraint Methodology (FPGA-AN-02059)
- Avant-E web page
- Avant-G web page
- Avant-X web page
- Certus-N2 web page
- Certus-NX web page
- CertusPro-NX web page
- CrossLink-NX web page
- MachXO5-NX web page
- Scatter-Gather DMA Controller IP Core web page
- Lattice Radiant 2023.1.1 Software Release Notes
- Lattice Radiant Software 2023.1 User Gude
- Lattice Radiant FPGA design software
- Lattice Insights for Lattice Semiconductor training courses and learning plans

Technical Support Assistance

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

For frequently asked questions, refer to the Lattice Answer Database at www.latticesemi.com/Support/AnswerDatabase.

Revision History

Note: In some instances, the IP may be updated without changes to the user guide. The user guide may reflect an earlier IP version but remains fully compatible with the later IP version. Refer to the IP Release Notes for the latest updates.

Revision 1.7, IP v2.6.0, December 2025

Section	Change Summary
All	• Updated IP version on the cover page. • Made editorial fixes across the document.
Introduction	• Updated the IP core and Radiant version, added Driver Support row, and added table notes regarding the IP version in Table 1.1. Summary of the SGDMA Controller IP. • Updated Licensing and Ordering Information section content.
Functional Description	• Updated Figure 2.2. SGDMA Controller Core Block Diagram with Separate Clock Domains for AXI-S Transmitter and Receiver Enabled. • Updated Reset Overview section content, including adding Known Behavior sub-section.
IP Parameter Description	Updated Figure 3.1. SGDMA Controller IP User Interface.
Signal Description	Updated Table 4.1. Signal List to updated Description column for AXI-4 Stream Interface.
Register Description	• Updated Table 5.3. MM2S_CTRL – Memory Map to Streaming Control Register and Table 5.6. S2MM_CTRL – Memory Map to Streaming Control Register to remove known issue text for RESET. • Updated Table 5.4. MM2S_STS – Memory Map to Streaming Status Register to update XFER_CMPL description. • Updated Table 5.7. S2MM_STS – Memory Map to Streaming Status Register to update XFER_CMPL, AXI_DEC_ERR, and AXI_SLV_ERR description. • Updated Table 5.10. MM2S_ADDR – Memory Map to Streaming Address, Table 5.15. S2MM_ADDR – Streaming to Memory Map Address, and Table 5.17. S2MM_CONTROL – Streaming to Memory Map Control to update description for BUFFER_ADDR. • Updated field and description for BUFFER_SIZE in Table 5.12. MM2S_CONTROL – Memory Map to Streaming Control. • Updated field for RSVD and field and description for TRANSFERRED_SIZE in Table 5.13. MM2S_STATUS – Memory Map to Streaming Status and Table 5.18. S2MM_STATUS – Streaming to Memory Map Status.
Example Design	Updated section content, including Overview of Example Design and Features and Simulating the Example Design section content, including moving the Running Functional Simulation section content here, and removing Compiling the Example Design and Hardware Testing sections.
Designing with the IP	• Added note for screenshots at the beginning of the section. • Updated Radiant reference in Design Implementation. • Removed Compiling with Example Design sub-section.
Programming Model	Added Error Scenarios section.
Revision History	Added note regarding IP version.

Revision 1.6, IP v2.5.0, June 2025

Section	Change Summary
All	Updated IP version on the cover page.
Introduction	• Added MachXO5-NX device, added supported devices for LAV-AT-G/X, and updated name from Supported FPGA Family to Supported Devices; updated IP Core and Radiant version, and removed Targeted Devices row in Table 1.1. Summary of the SGDMA Controller IP. • Added MachXO5-NX support in Table 1.2. IP Support Readiness on Lattice FPGAs and Lattice Radiant Software Suite and Table 1.3. Ordering Part Number. • Added LFMXO5-65 device in Table 1.4. Lattice SGDMA IP Core Supported Speed Grade's Maximum Frequency for Each Individual Clock Domain during IP Standalone Compilation at the 0 Degrees Corner Scenario.
IP Parameter Description	Updated Figure 3.1. SGDMA Controller IP User Interface to align with the latest IP version.

Section	Change Summary
Register Description	Updated IP version for known issue in Table 5.3. MM2S_CTRL – Memory Map to Streaming Control Register and Table 5.6. S2MM_CTRL – Memory Map to Streaming Control Register.
Designing with the IP	Updated Figure 7.1 Module/IP Block Wizard, Figure 7.2. IP Configuration, Figure 7.3. Check Generated Result, Figure 7.8. Simulation Wizard, and Figure 7.9. Add and Reorder Source.
References	Added MachXO5-NX web page reference.

Revision 1.5, IP v2.4.0, March 2025

Section	Change Summary
All	Updated IP version in the cover page and changed all SGDMAC to *SGDMA Controller* references (including table caption, figures, and figure captions) in the document.
Acronyms in This Document	Corrected SGDMA description.
Introduction	Updated IP Core version in Table 1.1. Summary of the SGDMA Controller IP.
Functional Description	Updated Figure 2.4. SGDMA Controller IP MM2S Operation Phase.
Register Description	Updated IP version in Table 5.3. MM2S_CTRL – Memory Map to Streaming Control Register and Table 5.6. S2MM_CTRL – Memory Map to Streaming Control Register.
Example Design	Updated IP package version across this section.
Designing with the IP	Updated the constraint file in Timing Constraints, figure captions for Figure 7.4 Excerpt from Lattice Design Constraints File (.ldc) for the SGDMA Controller IP and Figure 7.5 Excerpt from SDC Constraints File (.sdc) for the SGDMA Controller IP, and Compiling with Example Design section content.
References	Added reference to the SGDMA IP core web page.

Revision 1.4, IP v2.3.0, December 2024

Section	Change Summary
All	Added IP version to the cover page and revision history.
Introduction	• Updated Overview of the IP content to add DMA controller information. Updated Table 1.1. Summary of the SGDMAC IP to add Certus-NX, Certus-N2, CrossLink-NX and APB info, updated Targeted Devices content, changed IP core version to IP Changes and updated content as well, and updated IP core and Radiant software version. • Added IP Support Summary section. • Updated Features section content. • Updated Licensing and Ordering Information section to add CrossLink-NX, Certus-NX, and Certus-N2 information in Table 1.3. Ordering Part Number. • Renamed IP Validation Summary section to Hardware Support and updated section content. • Added Speed Grade Supported section. • Removed Minimum Device Requirement section.
Functional Description	• Updated IP Architecture Overview section content including Figure 2.1. SGDMAC Core Block Diagram and added Figure 2.2. SGDMAC Core Block Diagram with Separate Clock Domains for AXI-S Transmitter and Receiver Enabled. • Updated Clocking section content including Table 2.1. Two-Clock Domain Attributes and Figure 2.3. SGDMAC IP Clock Domain Block Diagram. • Updated Reset section content including Table 2.2. Reset Attributes.
IP Parameter Description	Updated Figure 3.1. SGDMAC IP User Interface and Table 3.1. General Attributes
Signal Description	Updated Table 4.1. Signal list to update sections and rows, including for AXI-S Transmitter clock domains, Control and Status Register Access Interfaces, and APB Interface
Register Description	Updated Table 5.3. MM2S_CTRL – Memory Map to Streaming Control Register and Table 5.6. S2MM_CTRL – Memory Map to Streaming Control Register to add Known Issue info.

Section	Change Summary
Example Design	• Updated section content including figure captions and adding Table 6.1. Files Intended for Each Individual Board the Example Design Supports and updating Table 6.2. SGDMAC IP Configuration Supported by the Example Design, Table 6.3. Example Design Component, Table 6.4. Example Design Simulation Testbench Component. Updated and added SGDMAC diagrams (Figure 6.1. SGDMAC Example Design Block Diagram with Separate AXI-S Transmitter and Receiver Clock Domains Disabled and AXI-4Lite CSR Access Interface Selected to Figure 6.4. SGDMAC Example Design Block Diagram with Separate AXI-S Transmitter and Receiver Clock Domains Enabled and APB CSR Access Interface Selected) and Figure 6.8. Timing Check Error Warning Dialog Box. • Added *the* to section name in Compiling the Example Design. • Removed Certus-Pro NX Evaluation Board and SGDMAC Example Design Simulation Testbench Block diagrams.
Designing with the IP	Updated section content including Figure 7.1 Module/IP Block Wizard, Figure 7.2. IP Configuration, Figure 7.4 Create Clock Constraint File (.sdc) for the SGDMAC IP, Figure 7.7. Example Design Physical Constraint File (.pdc), Figure 7.8. Simulation Wizard, Figure 7.9. Add and Reorder Source, and Figure 7.10. Select Simulation Top Module.
Appendix A. Resource Utilization	Updated Resource columns and table caption for Table A.1. CertusPro-NX Resource Utilization with Additional Clock Domains Disabled and AXI4-Lite interface for CSR Register Access and Table A.3. Avant Resource Utilization with Additional Clock Domains Disabled and AXI4-Lite interface for CSR Register Access, added Table A.2. CertusPro-NX Resource Utilization with Additional Clock Domains Enabled and AXI4-Lite interface for CSR Register Access to Table A.6. Certus-NX Resource Utilization with Additional Clock Domains Enabled and AXI4-Lite interface for CSR Register Access and removed Default parameter: Bold text in this section.
References	Added SGDMA Release Notes document and Certus-N2, Certus-NX, and CrossLink-NX web page references.

Revision 1.3, June 2024

Section	Change Summary
All	Added captions for figures and tables without captions in the previous version across the document.
Introduction	Updated Table 1.3. IP Validation Level to add Avant devices.
Register Description	Updated table captions and reworked the read cycle description (as applicable) in Table 5.4. MM2S_STS – Memory Map to Streaming Status Register to Table 5.8. S2MM_CURDESC – Memory Map to Streaming Current Descriptor Pointer.
Designing with the IP	• Updated section names to Successful Test Run and Failed Test Run. • Updated Running Functional Simulation section description and added Figure 7.13. Passing Simulation Log.
Programming Model	Added section information in Programming Model.
Appendix A. Resource Utilization	Updated table caption to Table A.1. CertusPro-NX Resource Utilization and added reference to Table A.2. Avant Resource Utilization in the introductory text.
Appendix B. SGDMA Performance	Updated Table B.1. SGDMA IP Performance to remove colored rows.
References	Added document reference to SGDMA Driver API Reference.

Revision 1.2, December 2023

Section	Change Summary
All	Reworked document structure for clarity by re-arranging sections and subsections.
Disclaimer	Updated section contents.

Section	Change Summary
Inclusive language	Added this section contents.
Acronyms in This document	Reworked section contents.
Introduction	• Reworked section contents. • Reworked Section 4 Ordering Part Number and converted to Subsection 1.4 Licensing and Ordering Information. • Added IP Validation Summary and Minimum Device Requirements subsections. • Reworked Subsection 1.3 Conventions and renamed to Subsection 1.7 Naming Conventions. • Added Lattice Avant devices in Table 1.2 Ordering Part Number.
Functional Description	• Reworked section contents. • Added Clocking, Reset and Function Operations subsections.
IP Parameter Description	Reworked *Subsection 2.3 Attributes Summary* and moved it under this section.
Signal Description	Reworked *Subsection 2.2 Signal Description* and converted it to Signal Description section.
Register Description	Reworked *Subsection 2.4 Register Description* and converted it to Register Description section.
Example Design	Added this section.
Designing with the IP	Reworked *Section 3 Core Generation, Simulation, and Validation* and converted to this main section.
Programming Model	Added this section.
Appendix A. Resource Utilization	• Reworked section contents. • Added Table A.2. Avant Resource Utilization.
Appendix B. SGDMA	Added this section.
References	Reworked section contents.
Technical Support Assistance	Added FAQ link in this section.

Revision 1.1, June 2021

Section	Change Summary

Introduction	• Removed statements from introductory paragraph.
	• Updated Table 1.1. Summary of SGDMAC IP Core.
	    • Revised Supported FPGA Families
	    • Revised Targeted Devices
	    • Revised Lattice Implementation

IP Generation and Evaluation	In the Hardware Evaluation section, replaced specific product names with *Lattice FPGA devices built on the Lattice Nexus platform*.
Ordering Part Number	Added part numbers.
Appendix A. Resource Utilization	Added this section.
References	Added reference to the CertusPro-NX web page.

Revision 1.0, December 2020

Section	Change Summary
All	Initial release