AXI Central Direct Memory Access v4.1

LogiCORE IP Product Guide

Vivado Design Suite

PG034 April 4, 2018

XILINX
ALL PROGRAMMABLE™

Table of Contents

IP Facts

Chapter 1: Overview

Feature Summary . . . 8
Applications . . . 8
Licensing and Ordering . . . 8

Chapter 2: Product Specification

Standards . . . 9
Performance . . . 9
Resource Utilization . . . 10
Port Descriptions . . . 12
Register Space . . . 13

Chapter 3: Designing with the Core

Clocking . . . 32
Resets . . . 32
CDMA Lite Mode Restrictions . . . 32

Chapter 4: Design Flow Steps

Customizing and Generating the Core . . . 44
Constraining the Core . . . 49
Simulation . . . 50
Synthesis and Implementation . . . 50

Chapter 5: Example Design

Appendix A: Upgrading

Migrating to the Vivado Design Suite . . . 52
Upgrading in the Vivado Design Suite . . . 52

Appendix B: Debugging

Finding Help on Xilinx.com . . . 53
Vivado Design Suite Debug Feature . . . 54

Hardware Debug 55

Appendix C: Additional Design Information

Appendix D: Additional Resources and Legal Notices

Xilinx Resources 57
Documentation Navigator and Design Hubs 57
References 57
Revision History 59
Please Read: Important Legal Notices 60

Introduction

The Xilinx LogiCORE™ IP AXI Central Direct Memory Access (CDMA) core is a soft Xilinx Intellectual Property (IP) core for use with the Vivado® Design Suite. The AXI CDMA provides high-bandwidth Direct Memory Access (DMA) between a memory-mapped source address and a memory-mapped destination address using the AXI4 protocol. An optional Scatter Gather (SG) feature can be used to offload control and sequencing tasks from the system CPU. Initialization, status, and control registers are accessed through an AXI4-Lite slave interface, suitable for the Xilinx MicroBlaze™ processor.

Features

- AXI4 interface for data transfer
- Independent AXI4-Lite slave interface for register access
- Independent AXI4 Master interface for optional Scatter/Gather function
- Optional Data Realignment Engine
- Register Direct Mode
- Optional Scatter Gather DMA support
- Optional Store and Forward support
- Parameterized Read and Write Address Pipeline depths
- Fixed-address and incrementing-address burst support

LogiCORE IP Facts Table	
Core Specifics	
Supported Device Family(1)	UltraScale+™ UltraScale™ Zynq®-7000 All Programmable SoC, Xilinx 7 series FPGAs
Supported User Interfaces	AXI4, AXI4-Lite
Resources	See Table 2-3 and Table 2-4
Provided with Core	
Design Files	VHDL
Example Design	VHDL
Test Bench	VHDL
Constraints File	Xilinx Design Constraints (XDC) delivered with IP generation
Simulation Model	Not Provided
Supported S/W Driver(2)	Standalone and Linux
Tested Design Flows(3)	
Design Entry	Vivado Design Suite
Simulation	For supported simulators, see the Xilinx Design Tools: Release Notes Guide.
Synthesis	Xilinx Synthesis Technology (XST) Vivado Synthesis
Support	
Provided by Xilinx at the Xilinx Support web page	

Notes:

1. For a complete list of supported devices, see the Vivado IP catalog.
2. Standalone driver details can be found in the software development kit (SDK) directory.

   <install_directory>/SDK/<release>/data/embeddedsw/doc/xilinx_drivers.htm

   Linux OS and driver support information is available from the Xilinx Wiki page.
3. For the supported versions of the tools, see the Xilinx Design Tools: Release Notes Guide.

Overview

The AXI CDMA is designed to provide a centralized DMA function for use in an embedded processing system employing the AXI4 system interfaces. The core supports both a simple CDMA operation mode and an optional Scatter Gather mode.

Figure 1-1 shows the functional composition of the AXI CDMA. It contains the following major functional blocks:

- Register Module
- Scatter/Gather Block
- DMA Controller
- DataMover
- Reset Module

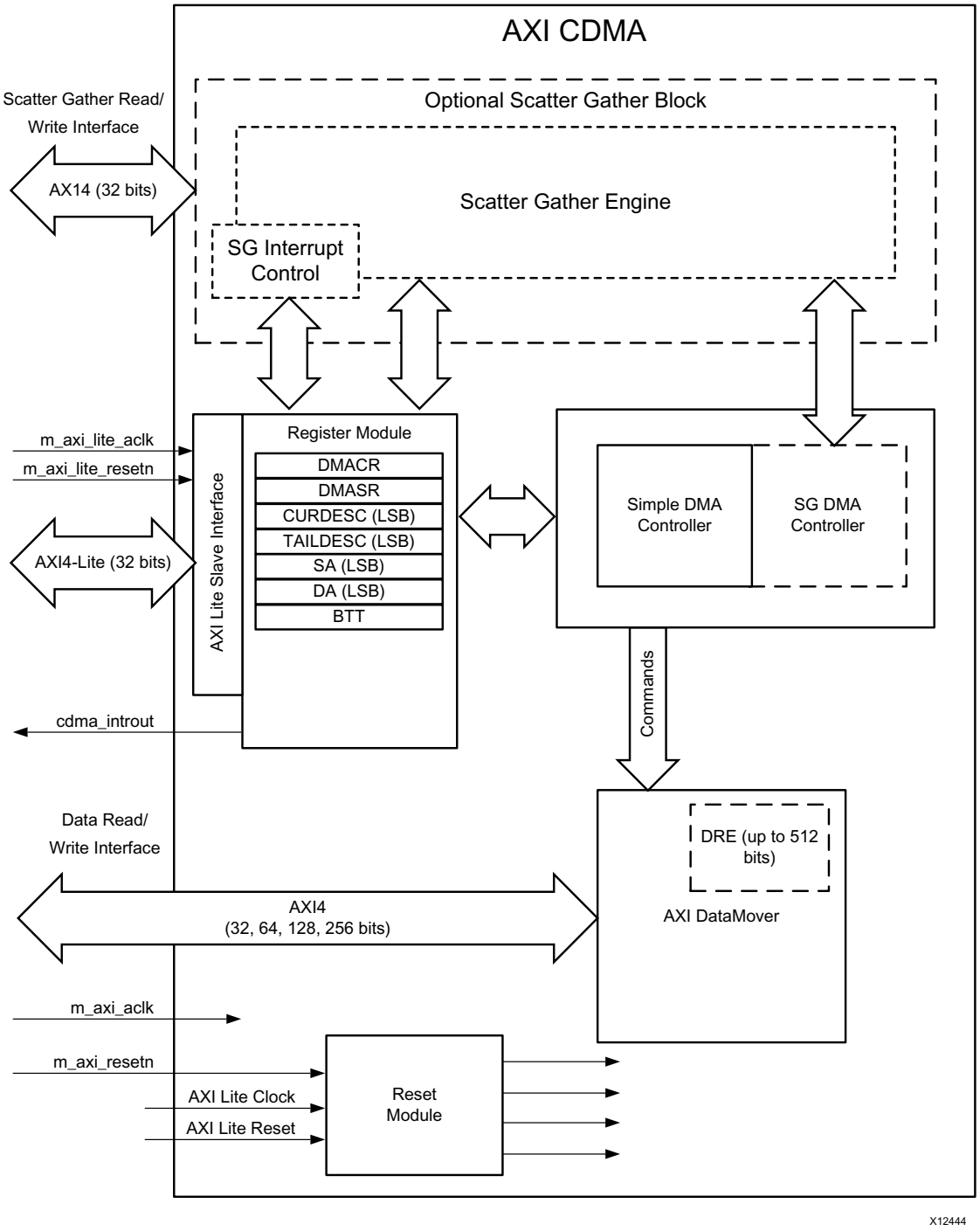

Figure 1-1: **AXI CDMA Block Diagram**

Register Module

This block contains control and status registers that allow you to configure AXI CDMA through the AXI4-Lite slave interface. See Register Space in Chapter 2.

Scatter/Gather Block

The AXI CDMA can optionally include Scatter/Gather (SG) functionality for off-loading CPU management tasks to hardware automation. The Scatter/Gather Engine fetches and updates CDMA control transfer descriptors from system memory through the dedicated AXI4 Scatter Gather Master interface. The SG engine provides internal descriptor queuing, which allows descriptor prefetch and processing in parallel with ongoing CDMA data transfer operations.

DMA Controller

This block manages overall CDMA operation. It coordinates DataMover command loading and status retrieval and updates it back to the Register Module.

DataMover

The DataMover is used for high-throughput transfer of data. The DataMover provides CDMA operations with 4 KB address boundary protection, automatic burst partitioning, and can queue multiple transfer requests. Furthermore, the DataMover provides byte-level data realignment (for up to 512-bit data widths) allowing the CDMA to read from and write to any byte offset combination.

Unaligned Transfers

The AXI DataMover core optionally supports the Data Realignment Engine (DRE). When the DRE is enabled, data is realigned to the byte (8 bits) level on the Memory Map datapath (up to 512-bit).

If the DRE is enabled, data reads can start from any Address byte offset. Similarly, when the DRE is enabled, the writes can happen at any byte offset address. For example, if Memory Map Data Width = 32, data is aligned if it is located at address offsets of 0x0, 0x4, 0x8, 0xC, etc. Data is unaligned if it is located at address offsets of 0x1, 0x2, 0x3 and so forth.

Note: Performing unaligned transfers when DRE is disabled will give unpredictable results.

Reset Module

The reset module generates the reset signals for various modules inside the CDMA.

Feature Summary

- Independent AXI4-Lite slave interface for register access
  - Fixed 32-bit data width
  - Optional asynchronous operation mode
- Independent AXI4 Master interface for the primary CDMA datapath. Parameterizable width of 32, 64, 128, 256, 512, and 1,024 bits with fixed-address burst (key hole) support.
- Independent AXI4 Master interface for optional Scatter/Gather function. Fixed 32-bit data width.
- Optional Data Realignment Engine for the primary CDMA datapath. Available for up to 512-bit datapath widths.
- Provides Simple DMA only mode and an optional hybrid mode supporting both Simple DMA and Scatter Gather automation.
- Support for up to 64-bit Address Space

Applications

The AXI CDMA provides a high-performance CDMA function for embedded systems.

Licensing and Ordering

This Xilinx LogiCORE™ IP module is provided at no additional cost with the Xilinx Vivado® Design Suite under the terms of the Xilinx End User License. Information about this and other Xilinx LogiCORE IP modules is available at the Xilinx Intellectual Property page. For information about pricing and availability of other Xilinx LogiCORE IP modules and tools, contact your local Xilinx sales representative.

Product Specification

Standards

The AXI CDMA core is AXI4 and AXI4-Lite compliant.

Performance

This section details the performance information for various core configurations.

Maximum Frequencies

The AXI DMA is characterized according to the benchmarking methodology described in the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 1]. Table 2-1 shows the results of the characterization runs.

Table 2-1: **Maximum Frequencies**

Family	Speed Grade	F_{Max} (MHz)	
		AXI4	AXI4-Lite
Virtex®-7	–1	200	180
Kintex®-7		200	180
Artix®-7		150	120
Virtex-7	–2	240	200
Kintex-7		240	200
Artix-7		180	140
Virtex-7	–3	280	220
Kintex-7		280	220
Artix-7		200	160

Throughput

Due to the parameterizable nature of the AXI CDMA, throughput is best measured as a percentage of the AXI4 bus bandwidth available to the AXI CDMA (Table 2-2). This available bus bandwidth is a function of the clock frequency of the AXI4 bus and the parameterized data width of the AXI CDMA. The other parameter affecting throughput is the value of Max Burst Length of the AXI CDMA.

In general, the bigger value assigned increases the realized throughput of the AXI CDMA. However, larger burst lengths can be detrimental to other components of a user system design, causing lower system performance.

Table 2-2: **AXI CDMA Throughput**

C_M_AXI_BURST_LEN	Test Packet Size	AXI4 Frequency	Observed Bus Bandwidth Utilization by AXI CDMA
16	9,000 bytes	150 MHz	70%
64	9,000 bytes	150 MHz	up to 99%

Resource Utilization

Resource utilization numbers for the AXI CDMA core are shown for the 7 series and Zynq®-7000 devices in Table 2-3 and for UltraScale™ devices in Table 2-4.

Table 2-3: **7 Series FPGA and Zynq-7000 Device Resource Estimates**

CDMA Parameters							Resources		
Addr Width	Enable Scatter gather	Disable 4K check	Allow Unaligned Transfer	Data Width	Burst Size	Enable Async	Slice	Luts	Reg
32	TRUE	FALSE	FALSE	32	16	FALSE	650	1,376	2,155
32	TRUE	FALSE	FALSE	64	8	FALSE	921	2,170	2,286
32	TRUE	FALSE	FALSE	64	8	TRUE	947	2,160	2,478
32	TRUE	FALSE	TRUE	64	8	FALSE	1,127	2,553	2,539
32	FALSE	TRUE	FALSE	64	16	FALSE	293	579	850
64	TRUE	FALSE	FALSE	32	16	FALSE	871	1,746	3,323
64	FALSE	FALSE	FALSE	64	16	TRUE	518	1,138	1,752

Table 2-4: **UltraScale Device Resource Estimates**

CDMA Parameters							Resources		
Addr Width	Enable Scatter gather	Disable 4K check	Allow Unaligned Transfer	Data Width	Burst Size	Enable Async	Slice/CLB	Luts	Reg
32	TRUE	FALSE	FALSE	32	16	FALSE	338	1,515	2,263
32	TRUE	FALSE	FALSE	64	8	FALSE	396	1,624	2,373
32	TRUE	FALSE	FALSE	64	8	TRUE	402	1,649	2,663
32	TRUE	FALSE	TRUE	64	8	FALSE	460	1,953	2,623
32	FALSE	TRUE	FALSE	64	16	FALSE	149	563	848
64	TRUE	FALSE	FALSE	32	16	FALSE	463	1,738	3,327
64	FALSE	FALSE	FALSE	64	16	TRUE	282	1,150	1,752

Port Descriptions

The AXI CDMA signals are described in Table 2-5.

Table 2-5: **AXI CDMA I/O Signal Description**

Signal Name	Interface	Signal Type	Init Status	Description
System Signals				
m_axi_aclk	Clock	I	–	AXI CDMA Synchronization Clock
cdma_introut	Interrupt	O	0	Interrupt output for the AXI CDMA core
AXI4-Lite Slave Interface Signals				
s_axi_lite_aclk	S_AXI_LITE	I	–	Synchronization clock for the AXI4-Lite interface. This clock can be the same as m_axi_aclk (synchronous mode) or different (asynchronous mode). ***Note:*** If it is asynchronous, the frequency of this clock must be less than or equal to the frequency of the m_axi_aclk.
s_axi_lite_aresetn	S_AXI_LITE	I	–	Active-Low AXI4-Lite Reset. When asserted Low, the AXI4-Lite Register interface and the entire CDMA core logic is put into hard reset. This signal must be synchronous to s_axi_lite_aclk.
s_axi_lite_*	S_AXI_LITE	I/O	–	See Appendix A of the *Vivado AXI Reference Guide* (UG1037)[Ref 2] for AXI4 signal.
CDMA Data AXI4 Read/Write Master Interface Signals				
m_axi_*	M_AXI	I/O	–	See Appendix A of the *Vivado AXI Reference Guide* (UG1037) [Ref 2] for AXI4 signal.
Scatter Gather AXI4 Read/Write Master Interface Signals				
m_axi_sg_*	M_AXI_SG	I/O	–	See Appendix A of the *Vivado AXI Reference Guide* (UG1037) [Ref 2] for AXI4 signal.

Notes:

1. The AXI CDMA IP does not use any of the ID pins of the AXI4 interfaces.
2. The IP wrapper/instantiation template has the 'cdma_tvect_out' port names. It is safe to keep these output pins open.

Register Space

The AXI CDMA core register space is summarized in Table 2-6. The AXI CDMA registers are memory-mapped into non-cacheable memory space.

Note: The AXI4-Lite write access register is updated by the 32-bit AXI Write Data (*_wdata) signal, and is not impacted by the AXI Write Data Strobe (*_wstrb) signal. For a Write, both the AXI Write Address Valid (*_awvalid) and AXI Write Data Valid (*_wvalid) signals should be asserted together.

IMPORTANT: *The registers are 32-bits wide, and the register memory space must be aligned on 128-byte (80h) boundaries.*

Endianess

All registers are in little endian format, as shown in Figure 2-1.

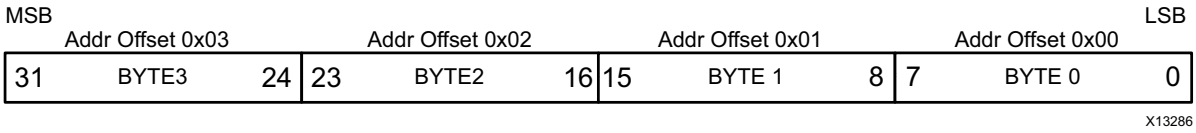

Figure 2-1: **32-bit Little Endian Example**

Register Address Map

Table 2-6: **AXI CDMA Register Summary**

Address Space Offset[(1)]	Name	Description
00h	CDMACR	CDMA Control
04h	CDMASR	CDMA Status
08h	CURDESC_PNTR	Current Descriptor Pointer
0Ch[(2)]	CURDESC_PNTR_MSB	Current Descriptor Pointer. MSB 32 bits. Applicable only when the address space is greater than 32.
10h	TAILDESC_PNTR	Tail Descriptor Pointer
14h[(2)]	TAILDESC_PNTR_MSB	Tail Descriptor Pointer. MSB 32 bits. Applicable only when the address space is greater than 32.
18h	SA	Source Address
1Ch[(2)]	SA_MSB	Source Address. MSB 32 bits. Applicable only when the address space is greater than 32.
20h	DA	Destination Address
24h[(2)]	DA_MSB	Destination Address. MSB 32 bits. Applicable only when the address space is greater than 32.

Table 2-6: **AXI CDMA Register Summary *(Cont'd)***

Address Space Offset[1]	Name	Description
28h	BTT	Bytes to Transfer

Notes:

1. Address Space Offset is relative to C_BASEADDR assignment.
2. This register is applicable only when the address width is more than 32. This register holds the upper 32 bits of the Address.

Register Details

CDMACR (CDMA Control – Offset 00h)

This register provides software application control of the AXI CDMA.

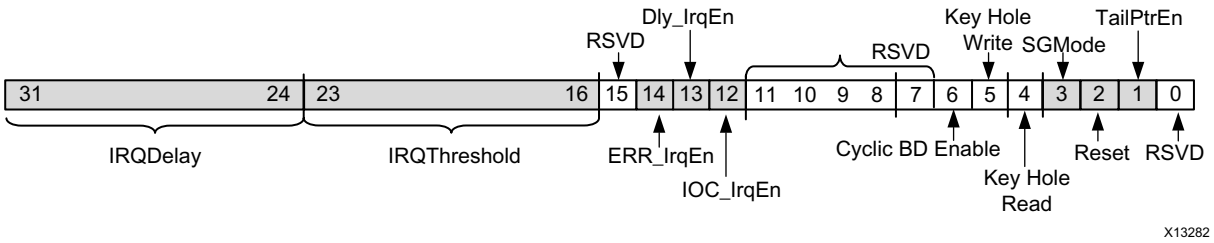

Figure 2-2: **CDMACR Register**

Table 2-7: **CDMACR Register Details**

Bits	Field Name	Default Value	Access Type	CDMA Mode Used	Description
31 to 24	IRQDelay	00h	R/W	SG	**Interrupt Delay Timeout**. This value is used for setting the interrupt delay timeout value. The interrupt timeout is a mechanism for causing the CDMA engine to generate an interrupt after the delay time period has expired. Timer begins counting at the end of a packet and resets with the receipt of a new packet or a timeout event occurs. ***Note:*** Setting this value to zero disables the delay timer interrupt. ***Note:*** This is irrelevant when Scatter Gather is excluded.
23 to 16	IRQThreshold	01h	R/W	SG	**Interrupt Threshold**. This value is used for setting the interrupt threshold. When IOC interrupt events occur, an internal counter counts down from the Interrupt Threshold setting. When the count reaches zero, an interrupt out is generated by the DMA engine. ***Note:*** The minimum setting for the threshold is 0x01. A write of 0x00 to this register has no effect. ***Note:*** This is irrelevant when Scatter Gather is excluded.
15	Reserved	0	RO	N/A	Writing to this bit has no effect and it is always read as zeros.
14	Err_IrqEn	0	R/W	Simple and SG	**Error Interrupt Enable**. When set to 1, it allows the CDMASR.Err_Irq to generate an interrupt out. 0 = Error Interrupt disabled 1 = Error Interrupt enabled

Table 2-7: **CDMACR Register Details *(Cont'd)***

Bits	Field Name	Default Value	Access Type	CDMA Mode Used	Description
13	Dly_IrqEn	0	R/W	SG	**Delay Timer Interrupt Enable**. When set to 1, it allows CDMASR.Dly_Irq to generate an interrupt out. This is only used with Scatter Gather assisted transfers. 0 = Delay Interrupt disabled 1 = Delay Interrupt enabled
12	IOC_IrqEn	0	R/W	Simple and SG	Complete Interrupt Enable. When set to 1, it allows CDMASR.IOC_Irq to generate an interrupt out for completed DMA transfers. 0 = IOC Interrupt disabled 1 = IOC Interrupt enabled
11 to 7	Reserved	0	RO	N/A	Writing to these bits has no effect and they are always read as zeros.
6	Cyclic BD Enable	0	R/W	SG	When set to 1, you can use the CDMA in Cyclic Buffer Descriptor (BD) mode without any user intervention. In this mode, the Scatter Gather module ignores the Completed bit of the BD. With this feature, you can use the same BDs in cyclic manner without worrying about any errors. This bit should be set before updating the TAILDESC register. Changing this bit while the transfer is in progress will generate undefined results.
5	Key Hole Write	0	R/W	Simple and SG	Writing 1 to this enables the keyhole write (FIXED address AXI transaction). This value should not be changed when a transfer is in progress. This value should remain constant until all the descriptors are processed (for SG = 1). CDMA shows unexpected behavior if this value is changed in the middle of a transfer. It is the responsibility of the slave device to enforce the functionality. When enabling Key Hole operation, the MAX BURST LENGTH should be set to 16.
4	Key Hole Read	0	R/W	Simple and SG	Writing 1 to this enables the keyhole read (FIXED address AXI transaction). This value should not be changed when a transfer is in progress. This value should remain constant until all the descriptors are processed (for SG = 1). CDMA shows unexpected behavior if this value is changed in the middle of a transfer. It is the responsibility of the slave device to enforce the functionality. When enabling Key Hole operation, the MAX BURST LENGTH should be set to 16.

Table 2-7: **CDMACR Register Details *(Cont'd)***

Bits	Field Name	Default Value	Access Type	CDMA Mode Used	Description
3	SGMode	0	R/W	Simple and SG	This bit controls the transfer mode of the CDMA. Setting this bit to a 1 causes the AXI CDMA to operate in a Scatter Gather mode if the Scatter Gather engine is included. 0 = Simple DMA Mode 1 = Scatter Gather Mode This bit must only be changed when the CDMA engine is idle (CDMASR.IDLE = 1). Changing the state of this bit at any other time has undefined results. This bit must be set to a 0 then back to 1 by the software application to force the CDMA SG engine to use a new value written to the CURDESC_PNTR register. This bit must be set prior to setting the CDMACR.Dly_IrqEn bit. Otherwise, the CDMACR.Dly_IrqEn bit does not get set.
2	Reset	0	R/W	Simple and SG	Soft reset control for the AXI CDMA core. Setting this bit to a 1 causes the AXI CDMA to be reset. Reset is accomplished gracefully. Committed AXI4 transfers are then completed. Other queued transfers are flushed. After completion of a soft reset, all registers and bits are in the Reset State. 0 = Reset NOT in progress – Normal operation 1 = Reset in progress
1	TailPntrEn	1	RO	SG	Indicates tail pointer mode is enabled to the SG Engine. This bit is fixed to 1 and always read as 1 when SG is included. If the CDMA is built with SG disabled (Simple Mode Only), the default value of the port is 0.
0	Reserved	0	RO	N/A	Writing to these bits has no effect, and they are always read as zeros.

Notes:

1. RO = Read Only. Writing has no effect.
2. R/W = Read/Write

CDMASR (CDMA Status – Offset 04h)

This register provides status for the AXI CDMA.

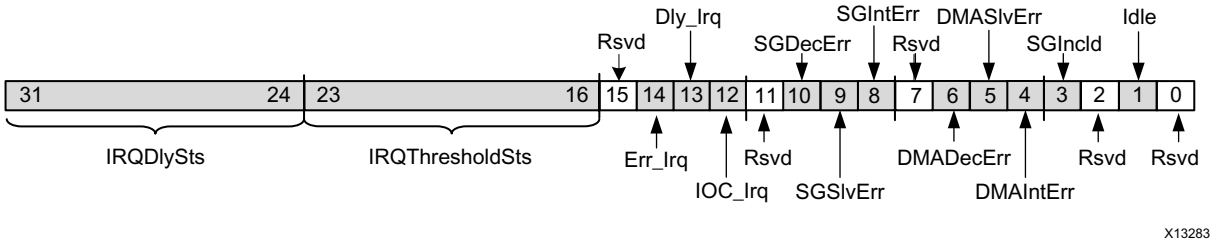

Figure 2-3: **CDMASR Register**

Table 2-8: **CDMASR Register Details**

Bits	Field Name	Default Value	Access Type	CDMA Mode Used	Description
31 to 24	IRQDelaySts	00h	RO	SG	**Interrupt Delay Time Status**. This field reflects the current interrupt delay timer value in the SG Engine.
23 to 16	IRQThresholdSts	01h	RO	SG	**Interrupt Threshold Status**. This field reflects the current interrupt threshold value in the SG Engine.
15	Reserved	0	RO	N/A	Always read as zero.
14	Err_Irq	0	R/WC	Simple and SG	**Interrupt on Error**. When set to 1, this bit indicates an interrupt event has been generated due to an error condition. If the corresponding enable bit is set (CDMACR.Err_IrqEn = 1), an interrupt out is generated from the AXI CDMA. 0 = No error Interrupt 1 = Error interrupt active Writing a 1 to this bit will clear it.
13	Dly_Irq	0	R/WC	SG	**Interrupt on Delay**. When set to 1, this bit indicates an interrupt event has been generated on a delay timer timeout. If the corresponding enable bit is set (CDMACR.Dly_IrqEn = 1), an interrupt out is generated from the AXI CDMA. 0 = No Delay Interrupt 1 = Delay Interrupt active Writing a 1 to this bit will clear it. This bit is cleared whenever CDMACR.SGMode is set to 0.

Table 2-8: **CDMASR Register Details *(Cont'd)***

Bits	Field Name	Default Value	Access Type	CDMA Mode Used	Description
12	IOC_Irq	0	R/WC	Simple and SG	**Interrupt on Complete**. When set to 1, this bit indicates an interrupt event has been generated on completion of a DMA transfer (either a Simple or SG). If the corresponding enable bit is set (CDMACR.IOC_IrqEn = 1), an interrupt out is generated from the AXI CDMA. 0 = No IOC Interrupt 1 = IOC Interrupt active When operating in SG mode, the criteria specified by the interrupt threshold must also be met. Writing a 1 to this bit will clear it.
11	Reserved	0	RO	N/A	Writing to this bit has no effect and it is always read as zeros.
10	SGDecErr	0	RO	SG	**Scatter Gather Decode Error**. This bit indicates that an AXI decode error has been received by the SG Engine during an AXI transfer (transfer descriptor read or write). This error occurs if the SG Engine issues an address request to an invalid location. This error condition causes the AXI CDMA to gracefully halt. The CDMASR.IDLE bit is set to 1 when the CDMA has completed shut down. The CURDESC_PNTR register is updated with the descriptor pointer value when this error is detected. 0 = No SG Decode Errors 1 = SG Decode Error detected. CDMA Engine halts. A reset (soft or hard) must be issued to clear the error condition.
9	SGSlvErr	0	RO	SG	**Scatter Gather Slave Error**. This bit indicates that an AXI slave error response has been received by the SG Engine during an AXI transfer (transfer descriptor read or write). This error condition causes the AXI CDMA to halt gracefully. The CDMASR.IDLE bit is set to 1 when the CDMA has completed shut down. The CURDESC_PNTR register is updated with the descriptor pointer value when this error is detected. 0 = No SG Slave Errors 1 = SG Slave Error detected. CDMA Engine halts. A reset (soft or hard) must be issued to clear the error condition.

Table 2-8: **CDMASR Register Details *(Cont'd)***

Bits	Field Name	Default Value	Access Type	CDMA Mode Used	Description
8	SGIntErr	0	RO	SG	**Scatter Gather Internal Error**. This bit indicates that an internal error has been encountered by the SG Engine. This error condition causes the AXI CDMA to halt gracefully. The CDMASR.IDLE bit is set to 1 when the CDMA has completed shutdown. The CURDESC_PNTR register is updated with the descriptor pointer value when this error is detected. 0 = No SG Internal Errors 1 = SG Internal Error detected. CDMA Engine halts. A reset (soft or hard) must be issued to clear the error condition.
7	Reserved	0	RO	N/A	Writing to this bit has no effect and it is always read as zeros.
6	DMADecErr	0	RO	Simple and SG	**DMA Decode Error.** This bit indicates that an AXI decode error has been received by the AXI DataMover. This error occurs if the DataMover issues an address request to an invalid location. This error condition causes the AXI CDMA to halt gracefully. The CDMASR.IDLE bit is set to 1 when the CDMA has completed shut down. The CURDESC_PNTR register is updated with the descriptor pointer value when this error is detected. 0 = No CDMA Decode Errors. 1 = CDMA Decode Error detected. CDMA Engine halts. A reset (soft or hard) must be issued to clear the error condition.
5	DMASlvErr	0	RO	Simple and SG	**DMA Slave Error**. This bit indicates that an AXI slave error response has been received by the AXI DataMover during an AXI transfer (read or write). This error condition causes the AXI CDMA to halt gracefully. The CDMASR.IDLE bit is set to 1 when the CDMA has completed shut down. The CURDESC_PNTR register is updated with the descriptor pointer value when this error is detected. 0 = No CDMA Slave Errors. 1 = CDMA Slave Error detected. CDMA Engine halts. A reset (soft or hard) must be issued to clear the error condition.

Table 2-8: **CDMASR Register Details *(Cont'd)***

Bits	Field Name	Default Value	Access Type	CDMA Mode Used	Description
4	DMAIntErr	0	RO	Simple and SG	**DMA Internal Error**. This bit indicates that a internal error has been encountered by the DataMover on the data transport channel. This error can occur if a 0 value BTT (bytes to transfer) is fed to the AXI DataMover or DataMover has an internal processing error. A BTT of 0 only happens if the BTT register is written with zeros (in Simple DMA mode) or a BTT specified in the Control word of a fetched descriptor is set to 0 (SG Mode). This error condition causes the AXI CDMA to halt gracefully. The CDMASR.IDLE bit is set to 1 when the CDMA has completed shut down. CURDESC_PNTR register is updated with the descriptor pointer value when this error is detected. 0 = No CDMA Internal Errors. 1 = CDMA Internal Error detected. CDMA Engine halts. A reset (soft or hard) must be issued to clear the error condition.
3	SGIncld	See Description	RO	Simple and SG	**SG Included.** This bit indicates if the AXI CDMA has been implemented with Scatter Gather support included. This is used by application software (drivers) to determine if SG Mode can be utilized. 0 = Scatter Gather not included. Only Simple DMA operations are supported. 1 = Scatter Gather is included. Both Simple DMA and Scatter Gather operations are supported.
2	Reserved	0	RO	N/A	Writing to these bits has no effect and they are always read as zeros.

Table 2-8: **CDMASR Register Details *(Cont'd)***

Bits	Field Name	Default Value	Access Type	CDMA Mode Used	Description
1	Idle	0	RO	Simple and SG	**CDMA Idle**. Indicates the state of AXI CDMA operations. When set and in Simple DMA mode, the bit indicates the programmed transfer has completed and the CDMA is waiting for a new transfer to be programmed. Writing to the bytes to transfer (BTT) register in Simple DMA mode causes the CDMA to start (not Idle). When set and in SG mode, the bit indicates the SG Engine has reached the tail pointer for the associated channel and all queued descriptors have been processed. Writing to the tail pointer register automatically restarts CDMA SG operations. 0 = Not Idle – Simple or SG DMA operations are in progress. 1 = Idle – Simple or SG operations completed or not started.
0	Reserved	0	RO	SG	Writing to these bits has no effect and they are always read as zeros.

Notes:

1. RO = Read Only. Writing has no effect.
2. R/W = Read/Write

CURDESC_PNTR (CDMA Current Descriptor Pointer – Offset 08h)

This register provides the Current Descriptor Pointer for the AXI CDMA Scatter Gather Descriptor Management.

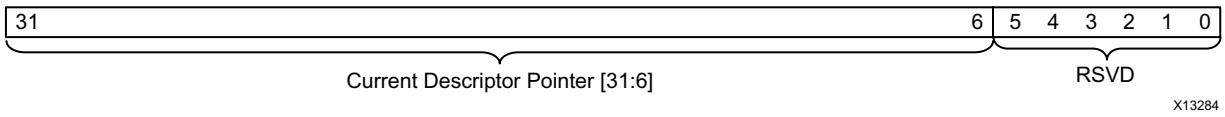

Figure 2-4:

Table 2-9: **CURDESC_PNTR Register Details**

Bits	Field Name	Default Value	Access Type	CDMA Mode Used	Description
31 to 6	Current Descriptor Pointer	0	R/W (RO)	SG	**Current Descriptor Pointer**. Indicates the pointer of the current descriptor being worked on. This register must contain a pointer to a valid descriptor prior to writing to the TAILDESC_PTR register. Failure to do so results in an undefined operation by the CDMA. When the CDMA SG Engine is running (CDMASR.IDLE = 0), the CURDESC_PNTR register is updated by the SG Engine to reflect the starting address of the current descriptor being executed. On error detection, the CURDESC_PNTR register is updated to reflect the descriptor associated with the detected error. The register should only be written by the software application when the AXI CDMA is idle (CDMASR.IDLE = 1). Descriptor addresses written to this field must be aligned to 64-byte boundaries (sixteen 32-bit words). Examples are 0x00, 0x40, 0x80. Any other alignment has undefined results. This register is cleared when CDMACR.SGMode = 0.
5 to 0	Reserved	0	RO	N/A	Writing to these bits has no effect and they are always read as zeros.

Notes:

1. RO = Read Only. Writing has no effect.
2. R/W = Read/Write

CURDESC_PNTR_MSB (CDMA Current Descriptor Pointer Offset – 0Ch)

This register provides the upper 32 bits of the Current Descriptor Pointer for the AXI CDMA Scatter Gather Descriptor Management. This is applicable when the address space is greater than 32.

Figure 2-5: **CURDESC_PNTR_MSB Register**

Table 2-10: **CURDESC_PNTR_MSB Register Details**

Bits	Field Name	Default Value	Access Type	CDMA Mode Used	Description
31 to 0	Current Descriptor Pointer	0	R/W (RO)	SG	**Current Descriptor Pointer**. Indicates the pointer of the current descriptor being worked on. This register must contain a pointer to a valid descriptor prior to writing to the TAILDESC_PTR register. Failure to do so results in an undefined operation by the CDMA. When the CDMA SG Engine is running (CDMASR.IDLE = 0), the CURDESC_PNTR register is updated by the SG Engine to reflect the starting address of the current descriptor being executed. On error detection, the CURDESC_PNTR register is updated to reflect the descriptor associated with the detected error. The register should only be written by the software application when the AXI CDMA is idle (CDMASR.IDLE = 1). Descriptor addresses written to this field must be aligned to 64-byte boundaries (sixteen 32-bit words). Examples are 0x00, 0x40, 0x80. Any other alignment has undefined results. This register is cleared when CDMACR.SGMode = 0.

Notes:

1. RO = Read Only. Writing has no effect.
2. R/W = Read/Write

TAILDESC_PNTR (CDMA Tail Descriptor Pointer – Offset 10h)

This register provides Tail Descriptor Pointer for the AXI CDMA Scatter Gather Descriptor Management.

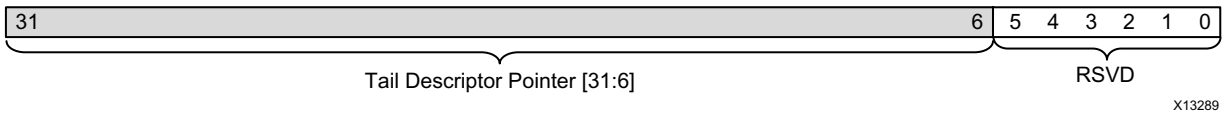

Figure 2-6: **TAILDESC_PNTR Register**

Table 2-11: **TAILDESC_PNTR Register Details**

Bits	Field Name	Default Value	Access Type	CDMA Mode Used	Description
31 to 6	Tail Descriptor Pointer	0	R/W (RO)	SG	**Tail Descriptor Pointer**. Indicates pause pointer for descriptor chain execution. The AXI CDMA SG Engine pauses descriptor fetching after completing operations on the descriptor whose current descriptor pointer matches the tail descriptor pointer. When the AXI CDMA is in SG Mode and the address space is 32 bits (CDMACR.SGMode = 1), a write by the software application to the TAILDESC_PNTR register causes the AXI CDMA SG Engine to start fetching descriptors starting from the CURDESC_PNTR register value. If the SG engine is paused at a tail pointer pause point, the SG engine restarts descriptor execution at the next sequential transfer descriptor. If the AXI CDMA is not idle (CDMASR.IDLE = 0), writing to the TAILDESC_PNTR has no effect except to reposition the SG pause point. This register is cleared when CDMACR.SGMode = 0.
5 to 0	Reserved	0	RO	N/A	Writing to these bits has no effect and they are always read as zeros.

Notes:

1. RO = Read Only. Writing has no effect.
2. R/W = Read/Write

TAILDESC_PNTR_MSB (CDMA Tail Descriptor Pointer — Offset 14h)

This register provides the MSB 32 bits of Tail Descriptor Pointer for the AXI CDMA Scatter Gather Descriptor Management. This is applicable only when the address space is greater than 32.

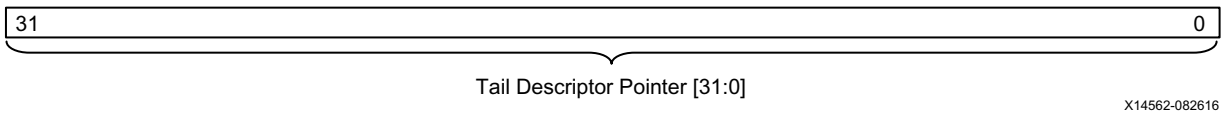

Figure 2-7: **TAILDESC_PNTR_MSB Register**

Table 2-12: **TAILDESC_PNTR_MSB Register Details**

Bits	Field Name	Default Value	Access Type	CDMA Mode Used	Description
31 to 0	Tail Descriptor Pointer	0	R/W (RO)	SG	**Tail Descriptor Pointer**. Indicates pause pointer for descriptor chain execution. The AXI CDMA SG Engine pauses descriptor fetching after completing operations on the descriptor whose current descriptor pointer matches the tail descriptor pointer. When the AXI CDMA is in SG Mode, and the address space is more than 32 bits, (CDMACR.SGMode = 1), a write by the software application to the TAILDESC_PNTR_MSB register causes the AXI CDMA SG Engine to start fetching descriptors starting from the CURDESC_PNTR register value. If the SG engine is paused at a tail pointer pause point, the SG engine restarts descriptor execution at the next sequential transfer descriptor. If the AXI CDMA is not idle (CDMASR.IDLE = 0), writing to the TAILDESC_PNTR has no effect except to reposition the SG pause point. This register is cleared when CDMACR.SGMode = 0.

Notes:

1. RO = Read Only. Writing has no effect.
2. R/W = Read/Write

SA (CDMA Source Address – Offset 18h)

This register provides the source address for Simple DMA transfers by AXI CDMA.

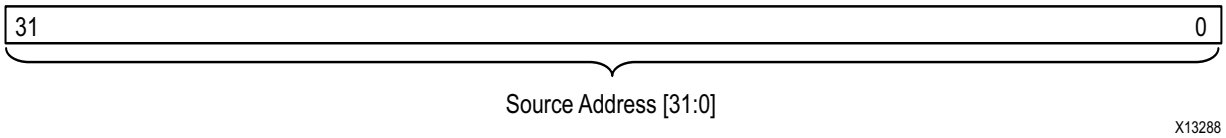

Figure 2-8: **SA Register**

Table 2-13: **SA Register Details**

Bits	Field Name	Default Value	Access Type	CDMA Mode Used	Description
31 to 0	SA	0	R/W	Simple	**Source Address Register**. This register is used by Simple DMA operations (CDMACR.SGMode = 0) as the starting read address for DMA data transfers. The address value written can be at any byte offset. The software application should only write to this register when the AXI CDMA is idle (CDMASR.IDLE = 1).

Notes:
1. R/W = Read/Write

SA_MSB (CDMA Source Address – Offset 1Ch)

This register provides the MSB 32 bits of source address for Simple DMA transfers by AXI CDMA. This is applicable only when the address space is more than 32 bits.

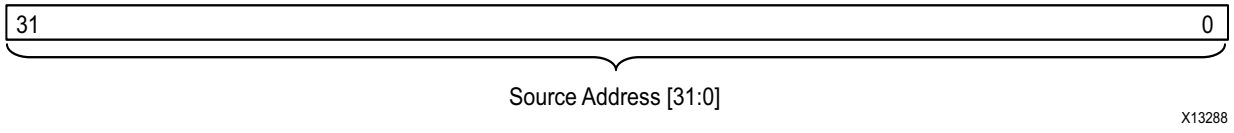

Figure 2-9: **SA_MSB Register**

Table 2-14: **SA_MSB Register Details**

Bits	Field Name	Default Value	Access Type	CDMA Mode Used	Description
31 to 0	SA	0	R/W	Simple	**Source Address Register**. This register is used by Simple DMA operations (CDMACR.SGMode = 0) as the starting read address for DMA data transfers. The address value written can be at any byte offset. The software application should only write to this register when the AXI CDMA is idle (CDMASR.IDLE = 1).

Notes:

1. R/W = Read/Write

DA (CDMA Destination Address – Offset 20h)

This register provides the Destination Address for Simple DMA transfers by AXI CDMA.

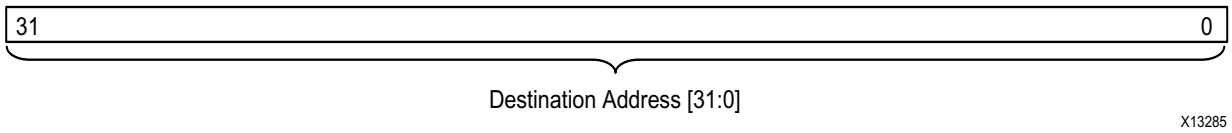

Figure 2-10: **DA Register**

Table 2-15: **DA Register Details**

Bits	Field Name	Default Value	Access Type	CDMA Mode Used	Description
31to 0	DA	0	RO	Simple	**Destination Address Register**. This register is used by Simple DMA operations as the starting write address for DMA data transfers. The address value written has restrictions relative to the Source Address and Data Realignment Engine (DRE) inclusion as follows. If DRE is not included in the AXI CDMA, then the address offset of the Destination address *must* match that of the Source Address register value. Offset is defined as that portion of a system address that is used to designate a byte position within a single data beat width. For example, a 32-bit data bus has four addressable byte positions within a single data beat (0, 1, 2, and 3). The portion of the address that designates these positions is the offset. The number of address bits used for the offset varies with the transfer bus data width. The software application should only write to this register when the AXI CDMA is idle (CDMASR.IDLE = 1).

Notes:

1. RO = Read Only. Writing has no effect.
2. R/W = Read/Write

DA_MSB (CDMA Destination Address — Offset 24h)

This register provides the MSB 32 bits of the Destination Address for Simple DMA transfers by AXI CDMA. This is applicable only when the address space is greater than 32.

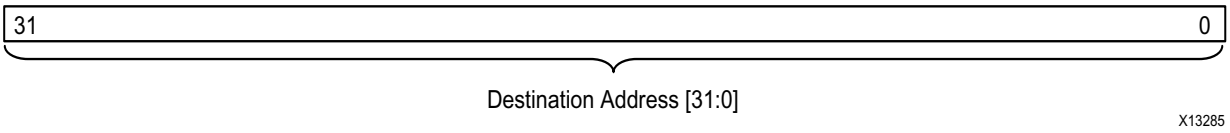

Figure 2-11: **DA_MSB Register**

Table 2-16: **DA_MSB Register Details**

Bits	Field Name	Default Value	Access Type	CDMA Mode Used	Description
31to 0	DA	0	RO	Simple	**Destination Address Register**. This register is used by Simple DMA operations as the starting write address for DMA data transfers. The software application should only write to this register when the AXI CDMA is idle (CDMASR.IDLE = 1).

Notes:

1. RO = Read Only. Writing has no effect.
2. R/W = Read/Write

BTT (CDMA Bytes to Transfer – Offset 28h)

This register provides the value for the bytes to transfer for Simple DMA transfers by the AXI CDMA.

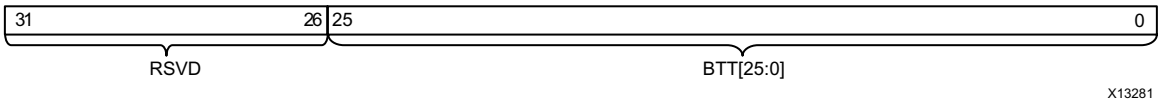

Figure 2-12: **BTT Register**

Table 2-17: **BTT Register Details**

Bits	Field Name	Default Value	Access Type	Description
31 to 26	Reserved	0	RO	Writing to these bits has no effect, and they are always read as zeros.
25 to 0	BTT	0	R/W (RO)	**Bytes to Transfer**. This register field is used for Simple DMA transfers and indicates the desired number of bytes to DMA from the Source Address to the Destination Address. A maximum of 67,108,863 bytes of data can be specified by this field for the associated transfer. Writing to the BTT register also initiates the Simple DMA transfer. ***Note:*** A value of zero (0) is not allowed and causes a DMA internal error to be set by AXI CDMA. The software application should only write to this register when the AXI CDMA is idle (CDMASR.IDLE = 1).

Notes:

1. RO = Read Only. Writing has no effect.
2. R/W = Read/Write

Designing with the Core

Clocking

AXI CDMA provides two clocking modes of operation: asynchronous and synchronous. In asynchronous mode, the AXI4-Lite interface is asynchronous to the AXI4-MMap interface. In this case, `s_axi_lite_aclk` can be less than or equal to `m_axi_aclk`. In synchronous mode, both `s_axi_lite_aclk` and `m_axi_aclk` should be connected to the same clock source.

Resets

An active-Low reset assertion on the `s_axi_lite_aresetn` input results in a reset of the entire AXI CDMA core logic. This is considered a hardware reset and there are no graceful completions of AXI4 transfers in progress. A hardware reset initializes all AXI CDMA registers to the default state, all internal queues are flushed, and all internal logic is returned to power on conditions. It is required that the `s_axi_lite_aresetn` input be synchronous to the `s_axi_lite_aclk` input.

Note: Reset should be asserted for a minimum of 16 clock cycles to the core to take effect.

CDMA Lite Mode Restrictions

The AXI DataMover that is internally used by the AXI CDMA can be optionally programmed to a reduced feature set to provide a reduced resource utilization footprint in the target FPGA (see CDMA resource utilizations in Table 2-3 and Table 2-4). Using the DataMover Lite operation mode puts restrictions on the available CDMA features. The following is a list of feature restrictions (compared to the Full mode).

- AXI CDMA data transport width is restricted to 32 and 64 bits.
- Maximum Burst Length is restricted to 16, 32, and 64 data beats.

- Maximum allowed bytes to transfer (BTT) value that is programmed per transfer request is limited to the specified maximum burst length times the specified CDMA data width divided by 8.

  MAX_BURST_LEN × (AXI_DATA_WIDTH/8)

- The DRE function is not supported with Data Mover Lite.

Note: In case the CDMA is configured in Lite Mode, the 4K address crossing guard must be done by the software application when specifying the Source Address, the Destination Address, and the BTT values programmed into the CDMA registers.

Programming Sequence

Simple DMA mode is the basic mode of operation for the CDMA when Scatter Gather is excluded. In this mode, the CDMA executes one programmed DMA command and then stops. This requires the CDMA registers to be set up by an external AXI4 Master for each DMA operation required.

These basic steps describe how to set up and initiate a CDMA transfer in simple operation mode.

1. Verify CDMASR.IDLE = 1.
2. Program the CDMACR.IOC_IrqEn bit to the desired state for interrupt generation on transfer completion. Also set the error interrupt enable (CDMACR.ERR_IrqEn), if so desired.
3. Write the desired transfer source address to the Source Address (SA) register. The transfer data at the source address must be valid and ready for transfer. If the address space selected is more than 32, write the SA_MSB register also.
4. Write the desired transfer destination address to the Destination Address (DA) register. If the address space selected is more than 32, then write the DA_MSB register also.
5. Write the number of bytes to transfer to the CDMA Bytes to Transfer (BTT) register. Up to 8,388,607 bytes can be specified for a single transfer (unless DataMover Lite is being used). Writing to the BTT register also starts the transfer.
6. Either poll the CDMASR.IDLE bit for assertion (CDMASR.IDLE = 1) or wait for the CDMA to generate an output interrupt (assumes CDMACR.IOC_IrqEn = 1).
7. If interrupt based, determine the interrupt source (transfer completed or an error has occurred).
8. Clear the CDMASR.IOC_Irq bit by writing a 1 to the DMASR.IOC_Irq bit position.
9. Ready for another transfer. Go back to step 1.

Scatter Gather Mode

Scatter Gather is a mechanism that allows for automated data transfer scheduling through a pre-programmed instruction list of transfer descriptors (Scatter Gather Transfer Descriptor Definition). This instruction list is programmed by your software application into a memory-resident data structure that must be accessible by the AXI CDMA SG interface. This list of instructions is organized into what is referred to as a transfer descriptor chain. Each descriptor has an address pointer to the next sequential descriptor to be processed. The last descriptor in the chain generally points back to the first descriptor in the chain but it is not required.

CDMA operations begins with the setup of the descriptor pointer registers and the CDMA control registers. The following lists minimum steps, in order, required for AXI CDMA operations:

1. Write a valid pointer to the channel CURDESC_PNTR register (Offset 0x08). If the address space selected is more than 32, then write the CURDESC_PNTR_MSB to specify the upper bits of current descriptor pointer.
2. Write control information to the channel CDMACR register (Offset 0x00) to set interrupt enables and key hole feature if desired.
3. Write a valid pointer to the channel TAILDESC_PNTR register (Offset 0x10). This starts the channel fetching and processing descriptors. If the address space selected is more than 32, then write the TAILDESC_PNTR_MSB to specify the upper bits of current descriptor pointer.
4. CDMA scatter gather operations continue until the descriptor at TAILDESC_PNTR is processed, and then the engine idles as indicated by CDMASR.Idle = 1.

Cyclic CDMA Mode

AXI CDMA can be run in cyclic mode by making certain changes to the BD chain setup. In cyclic mode, CDMA fetches and processes the same BDs without any interruption. The CDMA continues to fetch and process until it is stopped or reset. To enable cyclic operation, the BD chain should be set up as show in Figure 3-1.

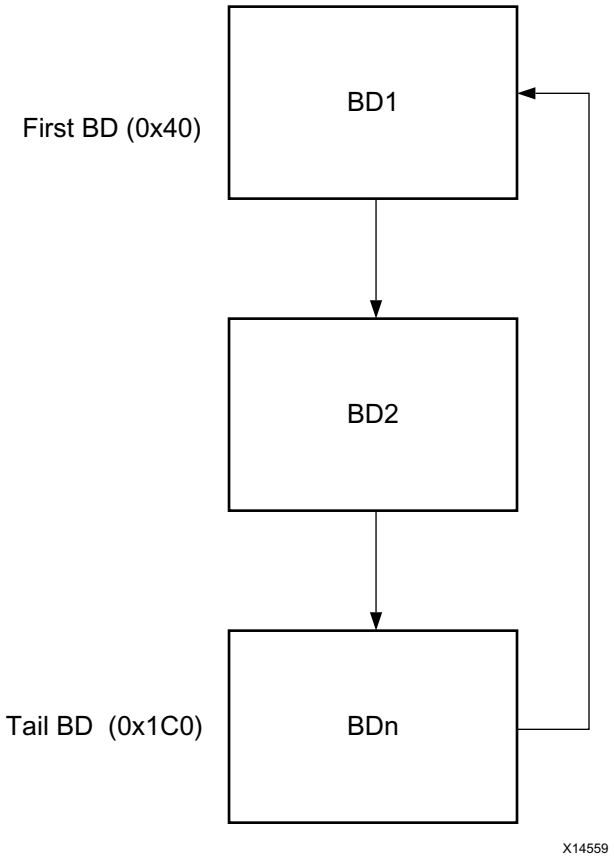

Figure 3-1: **BD Chain**

In this setup the Tail BD points back to the first BD. The Tail Descriptor Register does not serve any purpose and is used only to trigger the CDMA engine. Follow the same programming sequences as mentioned in Scatter Gather Mode. Ensure that the cyclic bit in the control register is set.

Program the Tail Descriptor register with some value which is not a part of the BD chain. Say for example 0x50.

After the Tail Descriptor register is programmed, the CDMA starts fetching and processing the BDs (which are set up in a ring fashion) until the CDMA is stopped or reset.

Transfer Descriptor Management

Prior to starting CDMA Scatter Gather operations, the software application must set up a transfer descriptor chain. After the AXI CDMA begins SG operations, it fetches, processes, and then updates the descriptors. By analyzing the descriptors, the software application can track the status of the associated CDMA data transfer and determine the completion status of the transfer. With this information, the software application can manage the transfer.

The act of writing to the TAILDESC register causes the AXI CDMA hardware, if it is paused at the tail pointer, to begin processing descriptors again.

If the AXI CDMA hardware is not paused at the TAILDESC pointer, writing to the TAILDESC register makes AXI CDMA continue to process descriptors until it reaches the new tail descriptor pointer location.

Scatter Gather Transfer Descriptor Definition

This defines the format and contents of the AXI CDMA Scatter Gather Transfer Descriptors. These are used only by the SG function if it is enabled in the CDMA and the CDMACR.SGMode bit is set to 1. A transfer descriptor consists of eight 32-bit words. The descriptor represents the control and status information needed for a single CDMA transfer plus address linkage to the next sequential descriptor. Each descriptor can define a single CDMA transfer of up to 8,388,607 Bytes of data. A descriptor chain is defined as a series of descriptors that are sequentially linked through the address linkage built into the descriptor format.

The AXI CDMA SG Engine traverses the descriptor chain following the linkage until the last descriptor of the chain has been completed. The relationship and identification of the AXI CDMA transfer descriptor words is shown in Table 3-1.

Note: Transfer Descriptors must be aligned on 16 32-bit word alignment. Example valid offsets are 0x00, 0x40, 0x80, and 0xC0.

Table 3-1: **Transfer Descriptor Word Summary**

Address Space Offset[(1)]	Name	Description
00h	NXTDESC_PNTR	Next Descriptor Pointer
04h	NXTDESC_PNTR_MSB	MSB 32 bits of Next Descriptor Pointer. This is used only when the address space if greater than 32.
08h	SA	Source Address
0Ch	SA_MSB	MSB 32 bits of Source Address. This is used only when the address space if greater than 32.
10h	DA	Destination Address
14h	DA_MSB	MSB 32 bits of Destination Address. This is used only when the address space if greater than 32.
18h	CONTROL	Transfer Control
1Ch	STATUS	Transfer Status

Notes:

1. Address Space Offset is relative to the address of the first word of the Transfer Descriptor in system memory.

Transfer Descriptor NXTDESC_PNTR (Next Descriptor Pointer – Offset 00h)

This word provides the address pointer to the first word of the next transfer descriptor in the descriptor chain.

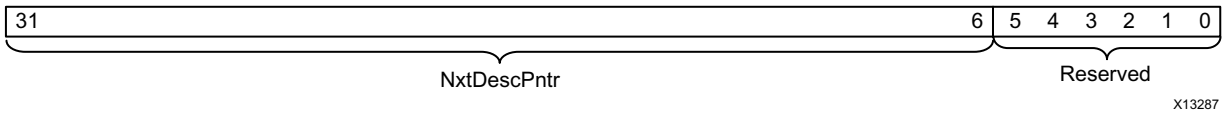

Figure 3-2: **Transfer Descriptor NXTDESC_PNTR Word**

Table 3-2: **Transfer Descriptor NXTDESC_PNTR Word Details**

Bits	Field Name	Description
31 to 6	NxtDescPntr	**Next Descriptor Pointer**. This field is an address pointer (most significant 26 bits) to the first word of the next transfer descriptor to be executed by the CDMA SG Engine. The least-significant 6 bits of this register are appended to this value when used by the SG Engine forcing transfer descriptors to be loaded in memory at 128-byte address alignment.
5 to 0	Reserved	These bits are reserved and fixed to zeros. This forces the address value programmed in this register to be aligned to 128-byte aligned addresses.

Transfer Descriptor NXTDESC_PNTR_MSB (Next Descriptor Pointer – Offset 04h)

This word provides the upper 32 bits of the address pointer to the first word of the next transfer descriptor in the descriptor chain. This is applicable only when the address space is greater than 32.

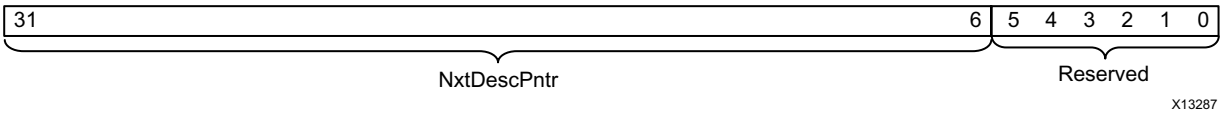

Figure 3-3: **Transfer Descriptor NXTDESC_PNTR_MSB Word**

Table 3-3: **Transfer Descriptor NXTDESC_PNTR_MSB Word Details**

Bits	Field Name	Description
31 to 0	NxtDescPntr	**Next Descriptor Pointer**. This field contains the MSB 32 bits of the Next Descriptor Pointer. This is applicable only when the address space is more than 32.

Transfer Descriptor SA Word (Source Address – Offset 08h)

This word provides the starting address for the data read operations for the associated DMA transfer.

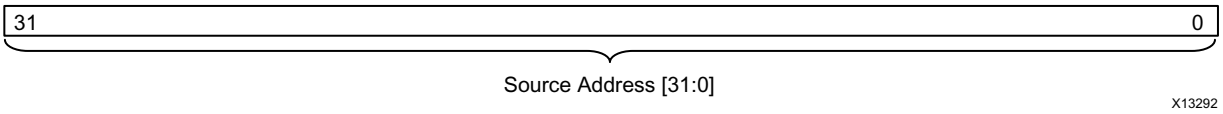

Figure 3-4: **Transfer Descriptor SA Word**

Table 3-4: **Transfer Descriptor SA Word Details**

Bits	Field Name	Description
31 to 0	SA	**Source Address**. This value specifies the starting address for data read operations for the associated DMA transfer. The address value can be at any byte offset.

Transfer Descriptor SA_MSB Word (Source Address — Offset 0Ch)

This word provides the MSB 32 bits of the starting address for the data read operations for the associated DMA transfer. This is used only when the address space is more than 32.

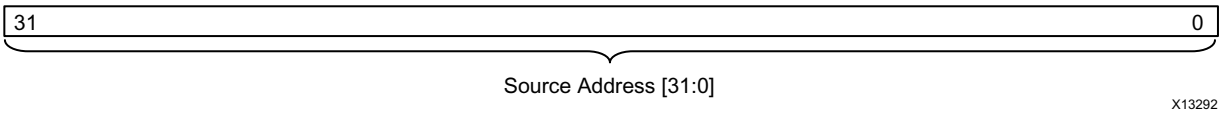

Figure 3-5: **Transfer Descriptor SA_MSB Word**

Table 3-5: **Transfer Descriptor SA_MSB Word Details**

Bits	Field Name	Description
31 to 0	SA	**Source Address**. This value specifies the MSB 32 bits of the starting address for data read operations for the associated DMA transfer. This field is applicable only when the address space is more than 32.

Transfer Descriptor DA Word (Destination Address – Offset 10h)

This word provides the starting address for the data write operations for the associated DMA transfer.

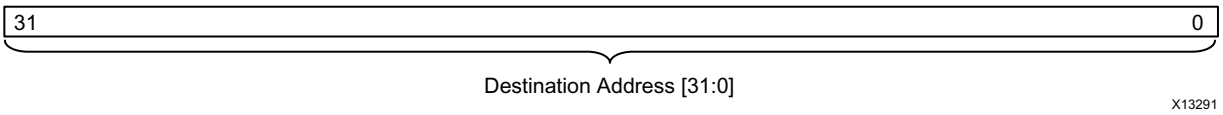

Figure 3-6: **Transfer Descriptor DA Word**

Table 3-6: **Transfer Descriptor DA Word Details**

Bits	Field Name	Description
31 to 0	DA	**Destination Address**. This value specifies the starting write address for DMA data transfers. The address value has restrictions relative to the Source Address and AXI CDMA DRE inclusion. If DRE is not included in the AXI CDMA, the address offset of the Destination Address *must* match that of the Source Address value.

Transfer Descriptor DA_MSB Word (Destination Address — Offset 14h)

This word provides the MSB 32 bits of the starting address for the data write operations for the associated DMA transfer. This is used only when the address space is greater than 32.

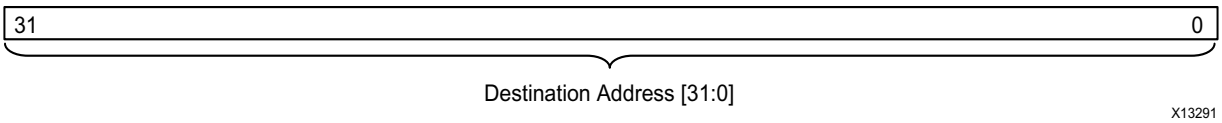

Figure 3-7: **Transfer Descriptor DA_MSB Word**

Table 3-7: **Transfer Descriptor DA_MSB Word Details**

Bits	Field Name	Description
31 to 0	DA	**Destination Address**. This value specifies the MSB 32 bits of the starting write address for DMA data transfers.

Transfer Descriptor CONTROL Word (Control – Offset 18h)

This value provides control for the AXI CDMA transfer.

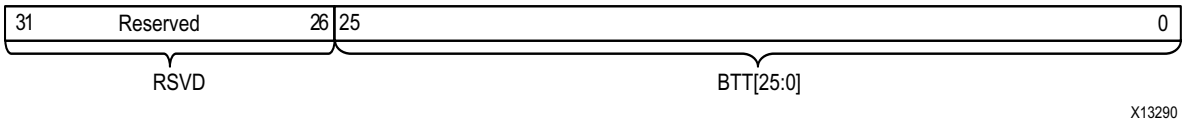

Figure 3-8: **Transfer Descriptor CONTROL Word**

Table 3-8: **Transfer Descriptor CONTROL Word Details**

Bits	Field Name	Description
31 to 26	Reserved	These bits are reserved and should be set to zero.
25 to 0	BTT	**Bytes to Transfer**. This field in the Control word specifies the desired number of bytes to DMA from the Source Address to the Destination Address. A maximum of 67,108,863 bytes of data can be specified by this field for the associated transfer. A value of zero (0) is not allowed and causes a DMA internal error to be set by AXI CDMA.

Transfer Descriptor Status Word (Status – Offset 1Ch)

This value provides the status to the software application regarding the execution of the Transfer Descriptor by the AXI CDMA. This status word should be zeroed when the transfer descriptor is programmed by the software application. When the AXI CDMA SG Engine has completed execution of the transfer descriptor, the status word is updated and written back to the original status word location in memory by the SG Engine.

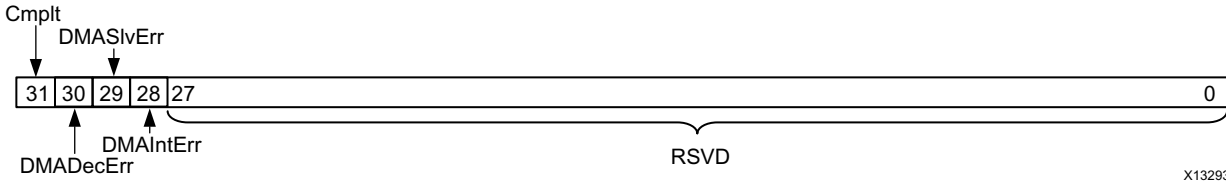

Figure 3-9: **Transfer Descriptor Status Word**

Table 3-9: **Transfer Descriptor Status Word Details**

Bits	Field Name	Description
31	CmpIt	**Transfer Completed**. This indicates to the software application that the CDMA Engine has completed the transfer as described by the associated descriptor. The software application can manipulate any descriptor with the Completed bit set to 1 when in Tail Pointer Mode (currently the only supported mode). 0 = Descriptor not completed 1 = Descriptor completed If the CDMA SG Engine fetches a descriptor, this bit is set to 1, the descriptor is considered a stale descriptor. An SGIntErr is flagged in the AXI CDMA Status register and the AXI CDMA engine halts with no update to the descriptor.
30	DMADecErr	**DMA Decode Error**. This bit indicates that an AXI decode error was received by the AXI CDMA DataMover. This error occurs if the DataMover issues an address that does not have a mapping assignment to a slave device. This error condition causes the AXI CDMA to halt gracefully. 0 = No CDMA Decode Errors 1 = CDMA Decode Error received. CDMA Engine halts at this descriptor
29	DMASlvErr	**DMA Slave Error**. This bit indicates that an AXI slave error response was received by the AXI CDMA DataMover during the AXI transfer (read or write) associated with this descriptor. This error condition causes the AXI CDMA to halt gracefully. 0 = No CDMA Slave Errors 1 = CDMA Slave Error received. CDMA Engine halts at this descriptor.
28	DMAIntErr	**DMA Internal Error**. This bit indicates that an internal error was encountered by the AXI CDMA DataMover on the data transport channel during the execution of this descriptor. This error can occur if a 0 value BTT (bytes to transfer) is fed to the AXI DataMover or DataMover has an internal processing error. A BTT of 0 only happens if the BTT field in the transfer descriptor CONTROL word is programmed with a value of zero. This error condition causes the AXI CDMA to gracefully halt. The CDMASR.IDLE bit is set to 1 when the CDMA has completed shutdown. 0 = No CDMA Internal Errors 1 = CDMA Internal Error detected. CDMA Engine halts at this descriptor
27 to 0	Reserved	These bits are reserved and should be set to zero.

Interrupts

An interrupt output is provided by the AXI CDMA. This output drives High when an internal interrupt event is logged in the CDMA Status Register (CDMASR) and the associated interrupt enable bit is set in the CDMA Control Register (CDMACR).

IMPORTANT: *This interrupt output is synchronized to the `s_axi_lite_aclk` clock input.*

Internal interrupt events are different depending on whether the AXI CDMA is operating in Simple DMA mode or SG Mode. Simple DMA mode generates an IOC interrupt whenever a programmed transfer is completed. In addition, three error conditions reported by the DataMover (internal error, slave error, and decode error) can also generate an interrupt assertion. For Scatter Gather mode, the delay interrupt and the three SG engine error interrupt events are added to the interrupt event mix.

SG Interrupt Threshold and Interrupt Coalescing

The AXI CDMA interrupt coalescing feature is enabled by setting the CDMACR.IRQThreshold field to a value greater than the default value of 1. When a SG session is started in the CDMA, the CDMACR.IRQThreshold field is loaded into the SG Engine threshold counter. With each Interrupt On Complete event generated by the SG Engine (occurs whenever the AXI CDMA completes a transfer descriptor), the threshold count is decremented. When the count reaches zero, an IOC_Irq is generated by the SG Engine. If the CDMACR.IOC_IrqEn = 1, an interrupt is generated on the AXI CDMA `cdma_introut` signal.

If the delay interrupt feature is enabled (CDMACR.IRQDelay not equal to 0), a delay interrupt event causes a reload of the SG Engine interrupt threshold counter. In addition, a write by the software application to the threshold value (CDMACR.Threshold), the internal threshold counter is reloaded.

SG Delay Interrupt

The delay interrupt feature is used in conjunction with the interrupt coalescing filter. Using the delay interrupt feature allows the software application to guarantee it receives an interrupt from the AXI CDMA, when the interrupt threshold is not met but the programmed descriptor chain has been completely processed by the AXI CDMA. The delay interrupt timer feature is enabled by setting the CDMACR.IRQDelay value to a non-zero value. The delay time is loaded into an internal counter in the SG Engine when a SG session is started. The internal timer begins counting up whenever the AXI CDMA is idle (CDMASR.IDLE = 1). The delay timer is reset and halted whenever the AXI CDMA is not idle (CDMASR.IDLE = 0).

Error Interrupts

Any detected error results in the AXI CDMA gracefully halting. Per AXI4 protocol, all AXI transfers that have been committed with accepted address channel transfers must complete the associated data transfer. Therefore, the AXI CDMA completes all committed AXI4 transfers before setting the CDMASR.IDLE bit. When the CDMASR.IDLE bit is set to 1, the AXI CDMA engine is truly halted.

The pointer to the descriptor associated with the errored transfer is updated to the CURDESC_PNTR register. On the rare occurrence that more than one error is detected, only the CURDESC_PNTR register for one of the errors are logged. To resume operations, a reset must be issued to the AXI CDMA either by a hardware reset or by writing a 1 to the CDMACR.Reset bit.

The following is a list of possible errors:

- **DMAIntErr** – CDMA Internal Error indicates that an internal error in the AXI DataMover was detected. This can occur under two conditions. First, for the DataMover MM2S and S2MM channels, it can occur when a BTT = 0 is written to the primary AXI DataMover command input. This happens if a transfer descriptor is fetched and executed with the CONTROL word having the BTT field = 0.
- **DMASlvErr** – CDMA Slave Error occurs when the slave to or from which data is transferred responds with a SLVERR.
- **DMADecErr** – CDMA Decode Error occurs when the address request is targeted to an address that does not exist.
- **SGIntErr** – Scatter Gather Internal Error occurs when a BTT = 0. This error only occurs if a fetched descriptor already has the Complete bit set. This condition indicates to the AXI CDMA that the descriptor has not been processed by the CPU, and therefore is considered stale.
- **SGSlvErr** – Scatter Gather Slave Error occurs when the slave to or from which descriptors are fetched and updated responds with a SLVERR.
- **SGDecErr** – Scatter Gather Decode Error occurs when the address request is targeted to an address that does not exist.

Note: Scatter Gather error bits are unable to be updated to the descriptor in remote memory. They are only captured in the associated channel CDMASR where the error occurred.

Design Flow Steps

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

- *Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator* (UG994) [Ref 3]
- *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 1]
- *Vivado Design Suite User Guide: Getting Started* (UG910) [Ref 4]
- *Vivado Design Suite User Guide: Logic Simulation* (UG900) [Ref 5]

Customizing and Generating the Core

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

If you are customizing and generating the core in the Vivado IP integrator, see the *Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator* (UG994) [Ref 3] for detailed information. IP integrator might auto-compute certain configuration values when validating or generating the design. To check whether the values do change, see the description of the parameter in this chapter. To view the parameter value, run the `validate_bd_design` command in the Tcl Console.

Vivado IP Catalog Options

The AXI CDMA can be found in **\AXI_Infrastructure** and also in **Embedded_Processing\AXI_Infrastructure\DMA** in the IP catalog.

To access the AXI CDMA, perform the following:

1. Open a project by selecting **File** then **Open Project** or create a new project by selecting **File** then **New Project** in the Vivado Integrated Design Environment (IDE).
2. Open the IP catalog and navigate to any of the taxonomies.
3. Double-click **AXI Central Direct Memory Access** to bring up the AXI CDMA Vivado IDE.

For details, see the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 1] and the *Vivado Design Suite User Guide: Getting Started* (UG910) [Ref 4].

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

If you are customizing and generating the core in the Vivado IP integrator, see the *Vivado Design Suite User Guide, Designing IP Subsystems Using IP Integrator* (UG994) [Ref 3] for detailed information. Vivado IDE might auto-compute certain configuration values when validating or generating the design, as noted in this section. You can view the parameter value after successful completion of the validate_bd_design command

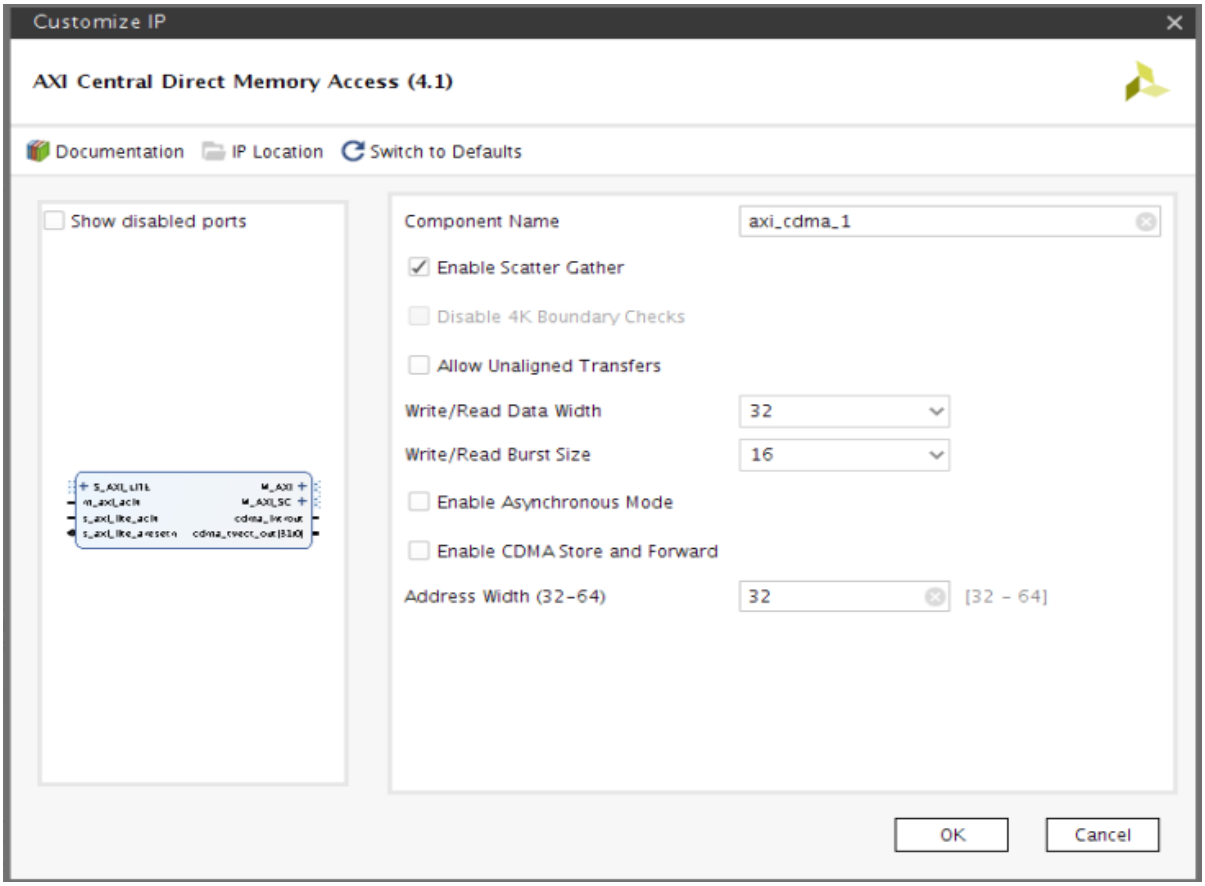

Figure 4-1: **AXI CDMA Vivado IDE**

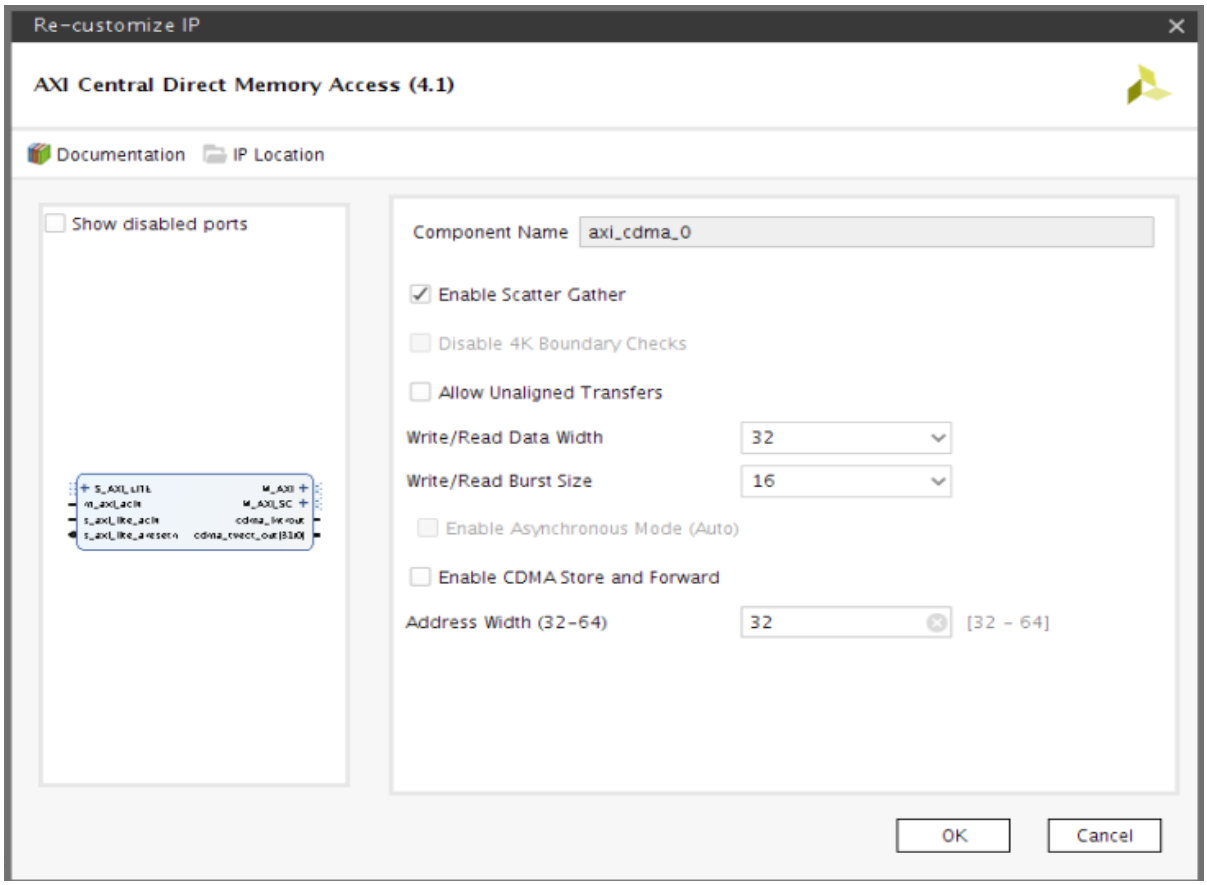

Figure 4-2: **IP Integrator**

Vivado IDE System Parameter

The AXI CDMA Vivado IDE contains one Vivado IDE (Figure 4-1) that provides information about the core, allows configuration of the core, and provides the ability to generate the core.

- **Component Name** – The base name of the output files generated for the core. Names must begin with a letter and can be composed of any of the following characters: a to z, 0 to 9, and "_".

The following options affect both channels of the AXI CDMA core.

- **Enable Scatter Gather** – Checking this option enables Scatter Gather Mode operation in AXI CDMA. Disabling the Scatter Gather Engine mode causes all output ports for the Scatter/Gather interface to be tied to zero, and all of the input ports to be left open.

- **Disable 4K Boundary Checks** – Checking this option puts the AXI CDMA in low resource utilization mode. This reduces considerably the performance and features of the AXI CDMA. This option is available only when Scatter Gather is disabled.
- **Allow Unaligned Transfers** – Enabling this option allows the AXI CDMA to access unaligned memory for Read and Write. This option is not available when 4K checks are disabled. Unaligned transfers are allowed for data widths up to 512 bits.
- **Read/Write Data Width** – Use this option to set the data width of the AXI4 data bus. Valid values are 32, 64, 128, 256, 512, and 1,024.
- **Write/Read Burst Size** – Use this option to set Maximum Burst size of the AXI4 read/write operation. When using the Keyhole feature of AXI CDMA, this needs to be set to 16.
- **Enable Asynchronous Mode** – Select this box if the clocks used in AXI CDMA are asynchronous.
- **Address Width (32-64)** – Select the required address space width. This can be any number between 32 and 64.
- **Enable CDMA Store and Forward** – Select this to enable an internal buffer in the AXI CDMA. Having an internal buffer in CDMA helps improve the performance of the AXI4 data bus.This option can also help avoid some lock up scenarios, when multiple AXI CDMAs transfer data amongst the common end points.

IMPORTANT: *The Enable Asynchronous Mode parameter is automatically set when the IP is used in the Vivado IP integrator.*

User Parameters

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

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

Vivado IDE parameter	User Parameter	Default Value
Enable Asynchronous mode	C_AXI_LITE_IS_ASYNC	0
Write/Read Data Width	C_M_AXI_DATA_WIDTH	32
Write/Read burst size	C_M_AXI_MAX_BURST_LEN	16
Disable 4K Checks	C_USE_DATAMOVER_LITE	0
Allow unaligned transfer	C_INCLUDE_DRE	0
Enable Scatter Gather	C_INCLUDE_SG	1
Address Width (32-64)	C_ADDR_WIDTH	32
Enable CDMA Store and Forward	C_INCLUDE-SF	0

Output Generation

For details, see the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 1].

Constraining the Core

Necessary XDC constraints are delivered along with the core generation in the Vivado Design Suite.

Required Constraints

This section is not applicable for this IP core.

Device, Package, and Speed Grade Selections

This section is not applicable for this IP core.

Clock Frequencies

This section is not applicable for this IP core.

Clock Management

This section is not applicable for this IP core.

Clock Placement

This section is not applicable for this IP core.

Banking

This section is not applicable for this IP core.

Transceiver Placement

This section is not applicable for this IP core.

I/O Standard and Placement

This section is not applicable for this IP core.

Simulation

For comprehensive information about Vivado simulation components, as well as information about using supported third-party tools, see the *Vivado Design Suite User Guide: Logic Simulation* (UG900) [Ref 5].

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

Synthesis and Implementation

For details about synthesis and implementation, see the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 1]. For details about implementation for the example design, see Chapter 5, Example Design.

Example Design

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

The top module instantiates all components of the core and example design that are needed to implement the design in hardware, as shown in Figure 5-1. This includes clock generator (MMCME2), register configuration, data generator, and data checker modules.

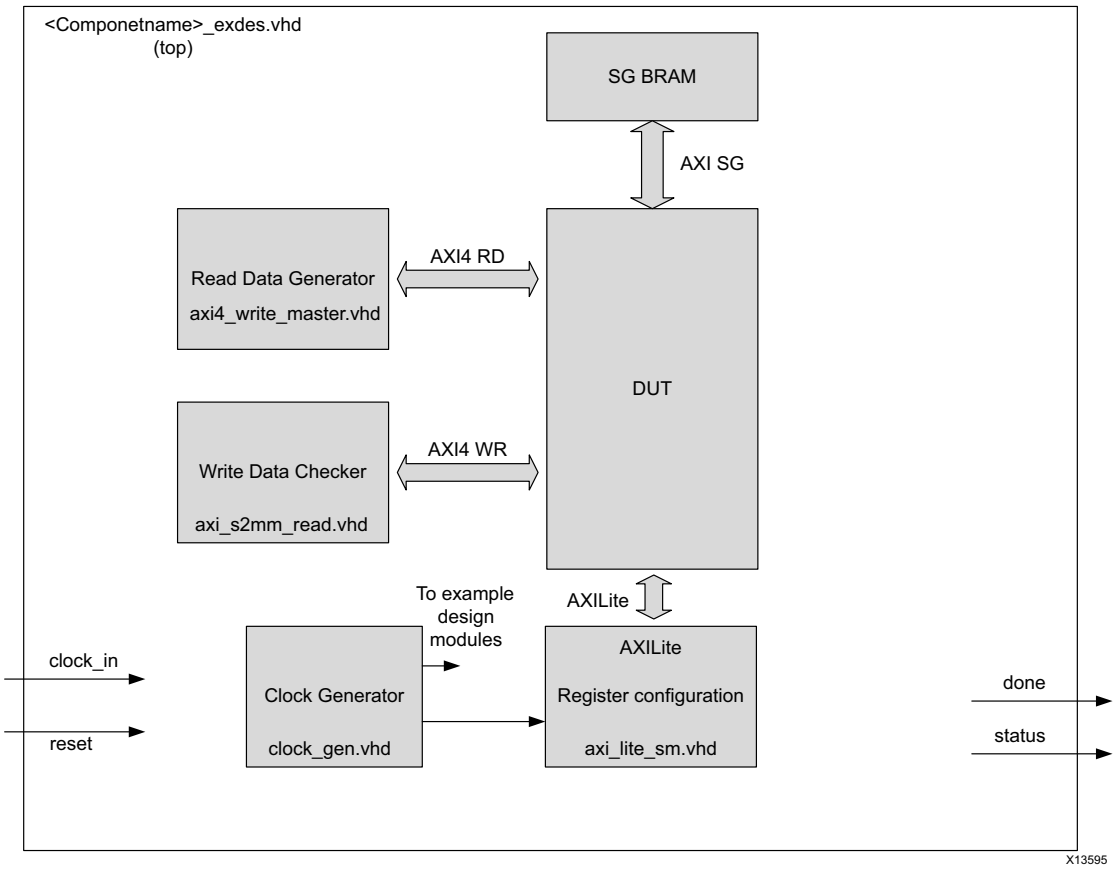

Figure 5-1: **Block Diagram of Example Design**

Upgrading

This appendix contains information about migrating a design from the ISE® Design Suite to the Vivado® Design Suite, and for upgrading to a more recent version of the IP core. For customers upgrading in the Vivado Design Suite, important details (where applicable) about any port changes and other impact to user logic are included.

Migrating to the Vivado Design Suite

For information about migrating to the Vivado Design Suite, see the *ISE to Vivado Design Suite Migration Guide* (UG911) [Ref 6].

Upgrading in the Vivado Design Suite

This section provides information about any changes to the user logic or port designations that take place when you upgrade to a more current version of this IP core in the Vivado Design Suite. There are no parameter or port changes for this core.

Debugging

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

Finding Help on Xilinx.com

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

Documentation

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

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

Answer Records

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

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

- Product name
- Tool message(s)
- Summary of the issue encountered

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

Master Answer Record for the AXI CDMA core

AR: 54685

Technical Support

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

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

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

Vivado Design Suite Debug Feature

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

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

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

Several internal signal are marked as debug signals. These can be easily added to the logic analyzer.

See the *Vivado Design Suite User Guide: Programming and Debugging* (UG908) [Ref 7].

Hardware Debug

Some of the common problems encountered and possible solutions follow.

- You have programmed your BD ring but nothing seems to work.

  Register programming sequence has to be followed to start the CDMA. See Programming Sequence in Chapter 3.

- Internal Error/Error bits set in the Status register.

  Internal error could be set when BTT specified in the descriptor is 0. SG internal error would be set if the fetched BD is a completed BD. Other error bits like Decode Error or Slave Error could also be set based on the response from Interconnect or Slave.

- You are reading data from a location, but the data does not seem to be in order.

  Verify if the start address location is aligned or un-aligned. If it is un-aligned, ensure that the DRE is enabled while configuring CDMA.

Additional Design Information

The parameter, C_INCLUDE_SF, is hidden because it is not required for most use cases. However, this parameter and associated features can be enabled by entering the following command in the Tcl Console in the Vivado® Integrated Design Environment (IDE)

The hidden parameter can be enabled in the Vivado design tools using the following command.

```
set_property -dict [list CONFIG.param_name {val}] [get_ips axi_cdma_xyz]
```

Where *axi_cdma_xyz* is the component name in the Vivado design tools.

To enable the parameter in the IP integrator, use the following command.

```
set_property -dict [list CONFIG.C_INCLUDE_SF {val}] [get_bd_cells axi_cdma_xyz]
```

Where *axi_cdma_xyz* is the component name in the IP integrator and `val` can be a 0 (to disable this feature) or 1 (to enable it).

Additional Resources and Legal Notices

Xilinx Resources

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

Documentation Navigator and Design Hubs

Xilinx® Documentation Navigator provides access to Xilinx documents, videos, and support resources, which you can filter and search to find information. To open the Xilinx Documentation Navigator (DocNav):

- From the Vivado® IDE, select **Help > Documentation and Tutorials**.
- On Windows, select **Start > All Programs > Xilinx Design Tools > DocNav**.
- At the Linux command prompt, enter `docnav`.

Xilinx Design Hubs provide links to documentation organized by design tasks and other topics, which you can use to learn key concepts and address frequently asked questions. To access the Design Hubs:

- In the Xilinx Documentation Navigator, click the **Design Hubs View** tab.
- On the Xilinx website, see the Design Hubs page.

Note: For more information on Documentation Navigator, see the Documentation Navigator page on the Xilinx website.

References

The following document provides supplemental material useful with this product guide:

1. *Vivado Design Suite User Guide: Designing with IP* (UG896)
2. *Vivado AXI Reference Guide* (UG1037)

3. *Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator* (UG994)
4. *Vivado Design Suite User Guide: Getting Started* (UG910)
5. *Vivado Design Suite User Guide - Logic Simulation* (UG900)
6. *ISE to Vivado Design Suite Migration Guide* (UG911)
7. *Vivado Design Suite User Guide: Programming and Debugging* (UG908)
8. AXI Interconnect LogiCORE IP Product Guide (PG059)
9. ARM AMBA 4 AXI4-Stream Protocol v1.0 Specification (ARM IHI 0051A)
10. Synthesis and Simulation Design Guide (UG626)

Revision History

The following table shows the revision history for this document.

Date	Version	Description of Revisions
04/04/2018	4.1	• Updated the Enable CDMA Store and Forward option. • Updated Table 2-5.
04/05/2017	4.1	IP updated to support DRE for data widths up to 512 bits.
10/05/2016	4.1	• Added a note about the AXI4-Lite write access register to the beginning of the Register Space section.
11/18/2015	4.1	Added support for UltraScale+ families.
09/30/2015	4.1	• Added a note to Table 2-6, AXI CDMA Register Summary. • Added Appendix C, Additional Design Information.
04/01/2015	4.1	Added support for 64-bit addressing.
03/20/2013	1.1	Updated for Vivado 2013.1, Zynq-7000, and 7 Series FPGAs.
12/18/2013	4.1	Added UltraScale™ architecture support.
10/02/2013	4.1	• Revision number advanced to 4.1 to align with core version number. • Added example design with implementation and test bench sections • Modified Bit 6 in CDMACR register to be Cyclic BD Enable. • Updated screen display in Chapter 4. • Added Cyclic CDMA Mode section to Chapter 3. • Added IP integrator content to Chapter 4.
03/20/2013	2.7	• Updated for Vivado 2013.1 design tool. • Removed information pertaining to Series 6 FPGAs. • Updated IP Facts table. • Updated Figure 1-1, AXI CDMA Block Diagram and supporting text. • Updated Table 2-1 and 3-1 • Updated Figure 4-1 and supporting text. • Updated Master Answer Records information and Contacting Technical Support and Debug Tools sections in Appendix B.
12/18/2012	2.6	• Updated for Vivado 2012.4 and ISE v14.4 design tools. • Updated Table 2-1, Maximum Frequencies • Updated Debugging appendix. • Updated resource utilization tables: Tables 2-3, 2-4, and 2-5. • Removed Table 3-1, Reset Assertion/Deassertion Stabilization Times • Removed Parameter Descriptions section • Updated screen capture for Chapter 4 and removed one screen. • Removed material from the Output Generation section in Chapter 4 and updated the output hierarchy.

Date	Version	Description of Revisions
10/16/2012	2.5	• Updated for Vivado 2012.3 and ISE v14.3 design tools. • Document clean up.
07/25/2012	2.0	• Updated for Vivado 2012.2, Zynq features, and ISE v14.2 • Added Vivado content in Customizing and Generating the Core
07/11/2012	1.1	Template update.
04/24/2012	1.0	Initial Xilinx Release This new document is based on the *LogiCORE IP AXI CDMA Product Specification* (DS792).

Please Read: Important Legal Notices

The information disclosed to you hereunder (the "Materials") is provided solely for the selection and use of Xilinx products. To the maximum extent permitted by applicable law: (1) Materials are made available "AS IS" and with all faults, Xilinx hereby DISCLAIMS ALL WARRANTIES AND CONDITIONS, EXPRESS, IMPLIED, OR STATUTORY, INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, OR FITNESS FOR ANY PARTICULAR PURPOSE; and (2) Xilinx shall not be liable (whether in contract or tort, including negligence, or under any other theory of liability) for any loss or damage of any kind or nature related to, arising under, or in connection with, the Materials (including your use of the Materials), including for any direct, indirect, special, incidental, or consequential loss or damage (including loss of data, profits, goodwill, or any type of loss or damage suffered as a result of any action brought by a third party) even if such damage or loss was reasonably foreseeable or Xilinx had been advised of the possibility of the same. Xilinx assumes no obligation to correct any errors contained in the Materials or to notify you of updates to the Materials or to product specifications. You may not reproduce, modify, distribute, or publicly display the Materials without prior written consent. Certain products are subject to the terms and conditions of Xilinx's limited warranty, please refer to Xilinx's Terms of Sale which can be viewed at https://www.xilinx.com/legal.htm#tos; IP cores may be subject to warranty and support terms contained in a license issued to you by Xilinx. Xilinx products are not designed or intended to be fail-safe or for use in any application requiring fail-safe performance; you assume sole risk and liability for use of Xilinx products in such critical applications, please refer to Xilinx's Terms of Sale which can be viewed at https://www.xilinx.com/legal.htm#tos.

AUTOMOTIVE APPLICATIONS DISCLAIMER

AUTOMOTIVE PRODUCTS (IDENTIFIED AS "XA" IN THE PART NUMBER) ARE NOT WARRANTED FOR USE IN THE DEPLOYMENT OF AIRBAGS OR FOR USE IN APPLICATIONS THAT AFFECT CONTROL OF A VEHICLE ("SAFETY APPLICATION") UNLESS THERE IS A SAFETY CONCEPT OR REDUNDANCY FEATURE CONSISTENT WITH THE ISO 26262 AUTOMOTIVE SAFETY STANDARD ("SAFETY DESIGN"). CUSTOMER SHALL, PRIOR TO USING OR DISTRIBUTING ANY SYSTEMS THAT INCORPORATE PRODUCTS, THOROUGHLY TEST SUCH SYSTEMS FOR SAFETY PURPOSES. USE OF PRODUCTS IN A SAFETY APPLICATION WITHOUT A SAFETY DESIGN IS FULLY AT THE RISK OF CUSTOMER, SUBJECT ONLY TO APPLICABLE LAWS AND REGULATIONS GOVERNING LIMITATIONS ON PRODUCT LIABILITY.

© Copyright 2012–2018 Xilinx, Inc. Xilinx, the Xilinx logo, Artix, ISE, Kintex, Spartan, Virtex, Vivado, Zynq, and other designated brands included herein are trademarks of Xilinx in the United States and other countries. All other trademarks are the property of their respective owners.