# LogiCORE IP AXI Central Direct Memory Access (axi_cdma) (v1.00.a)

DS792 September 21, 2010 **Product Specification**

## Introduction

The AXI Central Direct Memory Access (AXI CDMA) core is a soft Xilinx IP core for use with the Xilinx Embedded Development Kit (EDK). 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 may 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™ microprocessor.

## Features

- Independent AXI4-Lite Slave interface for register access
  - Fixed 32-bit data width
- Independent AXI4 Master interface for primary CDMA data path
  - Parameterizable width of 32, 64, 128, and 256 bits
- Independent AXI4 Master interface for optional Scatter/Gather function
  - Fixed 32-bit data width
- Optional Data Re-Alignment Engine for primary CDMA data path
  - Available with 32 and 64-bit data path widths
- Provides Simple DMA only mode and an optional hybrid mode supporting both Simple DMA and Scatter Gather automation

| LogiCORE IP Facts Table | | | | | | |
|---|---|---|---|---|---|---|
| **Core Specifics** | | | | | | |
| Supported Device Family(1) | Virtex®-6, Spartan®-6 | | | | | |
| Supported User Interfaces | AXI4, AXI4-Lite | | | | | |
| | **Resources** | | | | **Frequency** | |
| | **LUTs** | **FFs** | **DSP Slices** | **Block RAMs** | **Max. Freq.** | |
| | See Table 21 and Table 22. | | | | | |
| **Provided with Core** | | | | | | |
| Documentation | Product Specification | | | | | |
| Design Files | VHDL | | | | | |
| Example Design | Not Provided | | | | | |
| Test Bench | Not Provided | | | | | |
| Constraints File | Not Provided | | | | | |
| Simulation Model | Not Provided | | | | | |
| **Tested Design Tools** | | | | | | |
| Design Entry Tools | EDK 12.3, XPS | | | | | |
| Simulation | Mentor Graphics ModelSim: v6.5c | | | | | |
| Synthesis Tools | XST | | | | | |
| **Support** | | | | | | |
| Provided by Xilinx, Inc. | | | | | | |

1. For a complete listing of supported devices, see the release notes for this core.

## Applications

The AXI CDMA provides high performance Central Direct Memory Access function for Embedded Systems.

## Functional Description

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 automation mode. Figure 1 shows the functional composition of the AXI CDMA. The basic core design has an AXI4 Master interface for the main CDMA data transport function and an AXI4-Lite Slave interface for register accesses. An additional AXI4 Master interface is activated when the optional Scatter/Gather function is enabled.

The core's control and status registers are included in the Register Module. Access to these registers is provided through the AXI4-Lite Slave interface. The register module provides control and status for all CDMA operations.

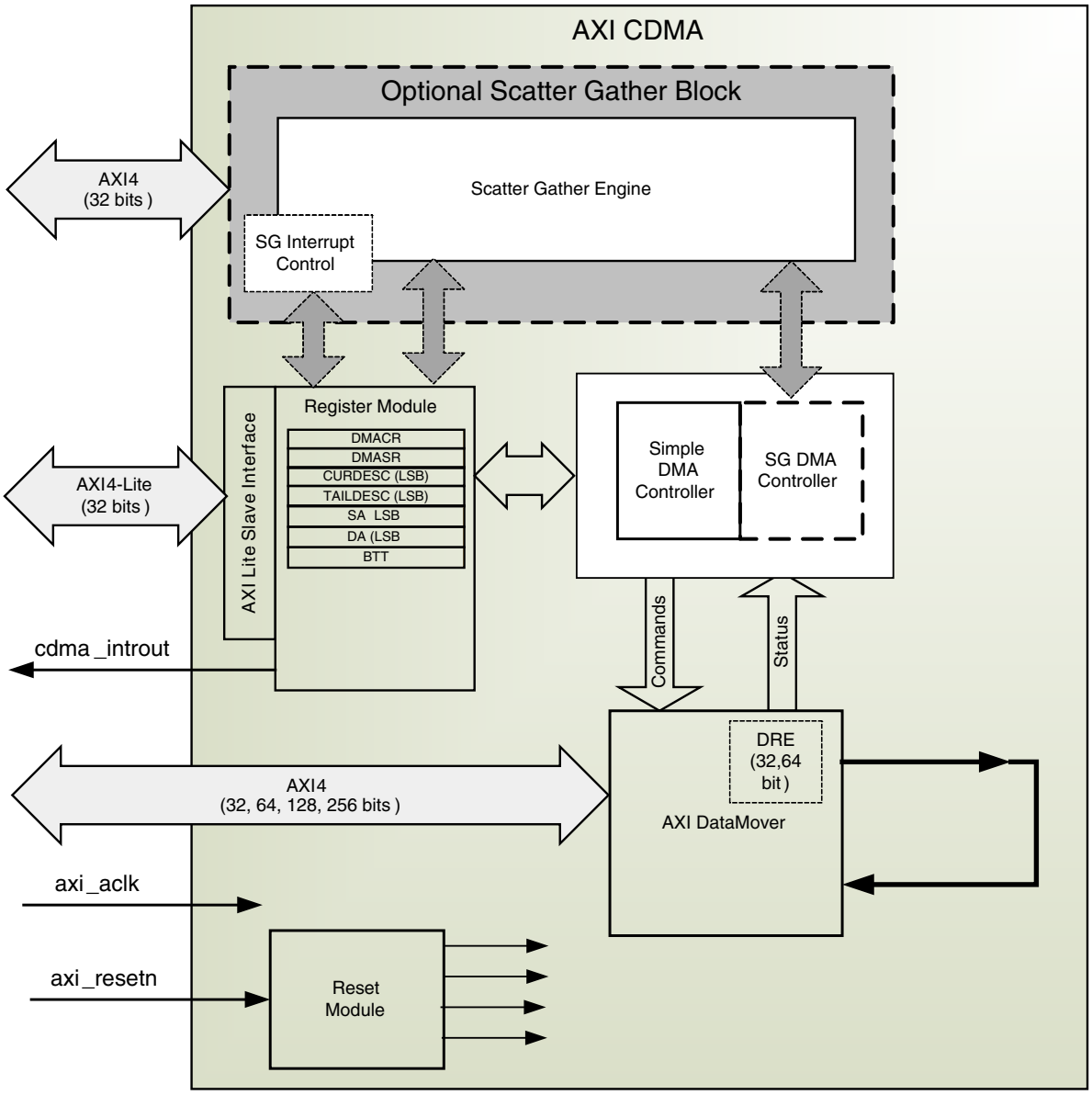


*Figure 1:* **AXI CDMA Block Diagram**

Primary high-speed CDMA data transport is provided by the AXI DataMover helper core. The AXI DataMover is used for high throughput transfer of data from AXI4 to AXI4-Stream and from AXI4-Stream to AXI4. In the case of the AXI CDMA application, the AXI DataMover's MM2S stream output is looped back to the S2MM stream input. The DataMover provides CDMA operations with 4 kbyte address boundary protection, automatic burst partitioning, as well as providing the ability to queue multiple transfer requests. Furthermore, the AXI DataMover provides byte-level data realignment (for 32-bit and 64-bit data widths) allowing the CDMA to read from and write to any byte offset combination.

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

Simple CDMA control sequencing is provided by the Simple Controller. It coordinates the DataMover command loading and status retrieval and update when a simple CDMA mode operation is commanded. The SG Controller is activated when the optional Scatter Gather function is enabled. The SG Controller coordinates the DataMover command loading and status retrieval just like the simple Controller but it also controls the Scatter Gather Engine operation.

All reset operations are coordinated by the Reset Module. It supports reset operations activated by the input `axi_resetn` (hardware reset) or by a User Application initiated by a write to the CDMA Control register reset bit (soft reset).

Clocking for the core is provided by a single input clock (`axi_aclk`). All CDMA operations are synchronous to the rising edge of this clock.

## Typical System Interconnect

The AXI CDMA core is designed to be connected in the user's embedded system via the AXI4 Interconnect. A typical MicroBlaze processor configuration is shown in Figure 2. The system's microprocessor has access to the AXI CDMA through the AXI4-Lite interface. For simple CDMA operations, the CDMA is programmed via the register interface to perform a single CDMA operation. The optional Scatter/Gather Engine fetches transfer descriptors from AXI_DDRx and automatically coordinates primary data transfers between the desired Source Address and Destination Address. The interrupt output of the AXI CDMA core is routed to the System's Interrupt Controller for use in interrupting the processor when CDMA operations have completed (in either simple or SG assisted modes).

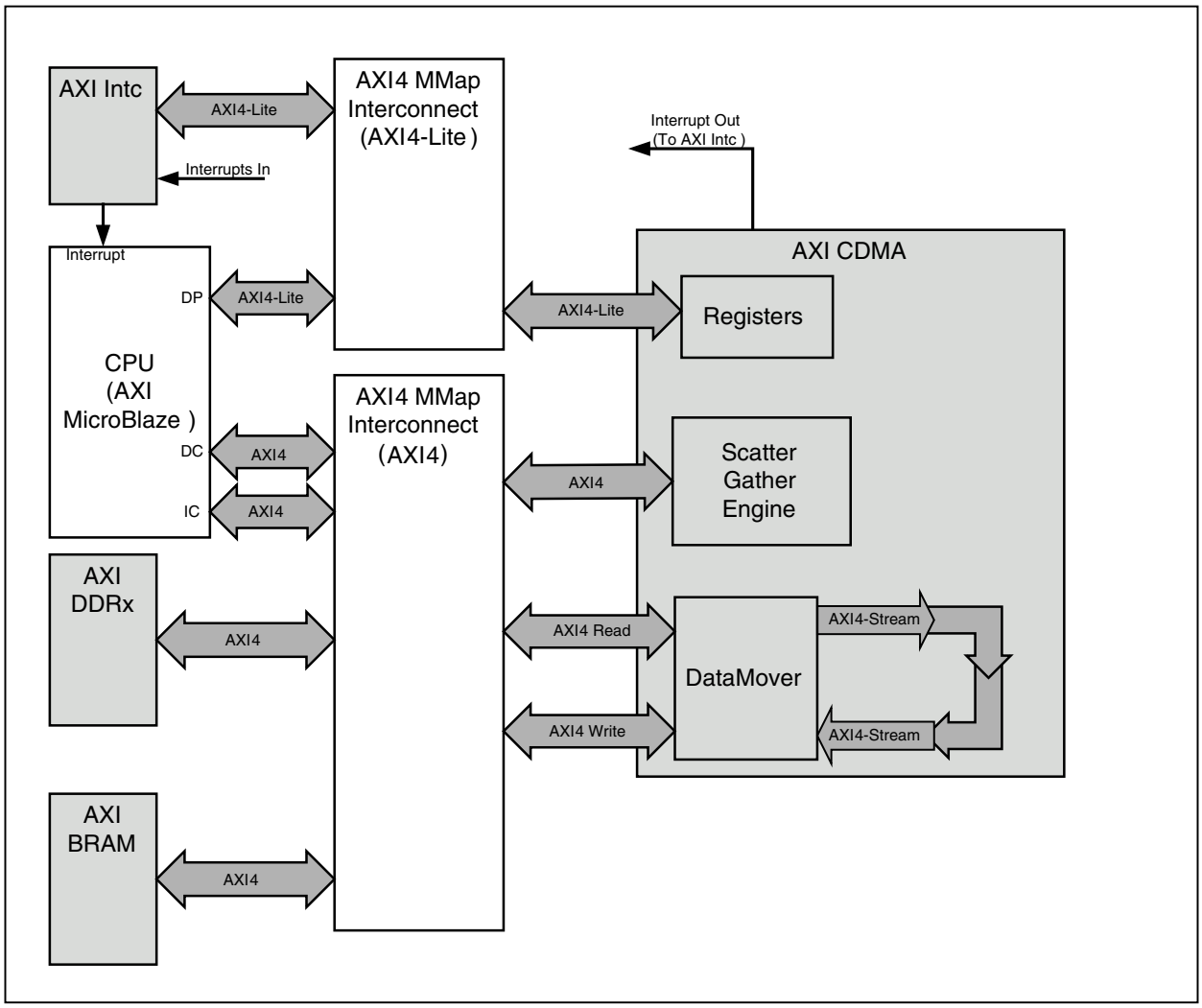


*Figure 2:* **Typical MicroBlaze Processor System Configuration Using AXI CDMA**

## I/O Signals

The AXI CDMA signals are described in Table 1.

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

| Signal Name | Interface | Signal Type | Init Status | Description |
|---|---|---|---|---|
| **System Signals** | | | | |
| axi_aclk | Clock | I | | AXI CDMA synchronization Clock. |
| axi_resetn | Reset | I | | AXI CDMA Reset. When asserted low, the AXI CDMA core is put into hard reset. This signal must be synchronous to axi_aclk. |
| cdma_introut | Interrupt | O | 0 | Interrupt output for the AXI CDMA core |
| **AXI4-Lite Slave Interface Signals** | | | | |
| s_axi_lite_awvalid | S_AXI_LITE | I | | AXI4-Lite Write Address Channel Write Address Valid.<br>• 1 = Write address is valid.<br>• 0 = Write address is not valid. |
| s_axi_lite_awready | S_AXI_LITE | O | 0 | AXI4-Lite Write Address Channel Write Address Ready. Indicates CDMA ready to accept the write address.<br>• 1 = Ready to accept address.<br>• 0 = Not ready to accept address. |
| s_axi_lite_awaddr(31:0) | S_AXI_LITE | I | | AXI4-Lite Write Address Bus. |
| s_axi_lite_wvalid | S_AXI_LITE | I | | AXI4-Lite Write Data Channel Write Data Valid.<br>• 1 = Write data is valid.<br>• 0 = Write data is not valid. |
| s_axi_lite_wready | S_AXI_LITE | O | 0 | AXI4-Lite Write Data Channel Write Data Ready. Indicates CDMA ready to accept the write data.<br>• 1 = Ready to accept data.<br>• 0 = Not ready to accept data. |
| s_axi_lite_wdata(31:0) | S_AXI_LITE | I | | AXI4-Lite Write Data Bus. |
| s_axi_lite_bresp(1:0) | S_AXI_LITE | O | zeros | AXI4-Lite Write Response Channel. Indicates results of the write transfer. The AXI CDMA Lite interface always responds with OKAY.<br>• 00b = OKAY - Normal access has been successful.<br>• 01b = EXOKAY - Not supported.<br>• 10b = SLVERR - Not supported.<br>• 11b = DECERR - Not supported. |
| s_axi_lite_bvalid | S_AXI_LITE | O | 0 | AXI4-Lite Write Response Channel Response Valid. Indicates response is valid.<br>• 1 = Response is valid.<br>• 0 = Response is not valid. |
| s_axi_lite_bready | S_AXI_LITE | I | | AXI4-Lite Write Response Channel Ready. Indicates target is ready to receive response.<br>• 1 = Ready to receive response.<br>• 0 = Not ready to receive response. |
| s_axi_lite_arvalid | S_AXI_LITE | I | | AXI4-Lite Read Address Channel Read Address Valid.<br>• 1 = Read address is valid.<br>• 0 = Read address is not valid. |

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

| Signal Name | Interface | Signal Type | Init Status | Description |
|---|---|---|---|---|
| s_axi_lite_arready | S_AXI_LITE | O | 0 | AXI4-Lite Read Address Channel Read Address Ready. Indicates CDMA ready to accept the read address.<br>• 1 = Ready to accept address.<br>• 0 = Not ready to accept address. |
| s_axi_lite_araddr(31:0) | S_AXI_LITE | I | | AXI4-Lite Read Address Bus. |
| s_axi_lite_rvalid | S_AXI_LITE | O | 0 | AXI4-Lite Read Data Channel Read Data Valid.<br>1 = Read data is valid<br>0 = Read data is not valid |
| s_axi_lite_rready | S_AXI_LITE | I | | AXI4-Lite Read Data Channel Read Data Ready. Indicates target ready to accept the read data.<br>• 1 = Ready to accept data.<br>• 0 = Not ready to accept data. |
| s_axi_lite_rdata(31:0) | S_AXI_LITE | O | zeros | AXI4-Lite Read Data Bus. |
| s_axi_lite_rresp(1:0) | S_AXI_LITE | O | zeros | AXI4-Lite Read Response Channel Response. Indicates results of the read transfer. The AXI CDMA Lite interface always responds with OKAY.<br>• 00b = OKAY - Normal access has been successful.<br>• 01b = EXOKAY - Not supported.<br>• 10b = SLVERR - Not supported.<br>• 11b = DECERR - Not supported. |
| **CDMA Data AXI4 Read Master Interface Signals** | | | | |
| m_axi_araddr (C_M_AXI_ADDR_ WIDTH-1: 0) | M_AXI | O | zeros | Read Address Channel Address Bus. |
| m_axi_arlen(7:0) | M_AXI | O | zeros | Read Address Channel Burst Length. In data beats - 1. |
| m_axi_arsize(2:0) | M_AXI | O | zeros | Read Address Channel Burst Size. Indicates with of burst transfer.<br>• 000b = Not Supported by AXI CDMA.<br>• 001b =Not Supported by AXI CDMA.<br>• 010b = 4 bytes (32-bit wide burst).<br>• 011b = 8 bytes (64-bit wide burst).<br>• 100b = 16 bytes (128-bit wide burst).<br>• 101b = 32 bytes (256-bit wide burst).<br>• 110b = Not Supported by AXI CDMA.<br>• 111b = Not Supported by AXI CDMA. |
| m_axi_arburst(1:0) | M_AXI | O | zeros | Read Address Channel Burst Type. Indicates type burst.<br>• 00b = FIXED - Not supported.<br>• 01b = INCR - Incrementing address.<br>• 10b = WRAP - Not supported.<br>• 11b = Reserved. |
| m_axi_arprot(2:0) | M_AXI | O | 000b | Read Address Channel Protection. Always driven with a constant output of 000b. |
| m_axi_arcache(3:0) | M_AXI | O | 0011b | Read Address Channel Cache. This is always driven with a constant output of 0011b. |

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

| Signal Name | Interface | Signal Type | Init Status | Description |
|---|---|---|---|---|
| m_axi_arvalid | M_AXI | O | 0 | Read Address Channel Read Address Valid. Indicates when the Read Address Channel qualifiers are valid.<br>• 1 = Read address is valid.<br>• 0 = Read address is not valid. |
| m_axi_arready | M_AXI | I | | Read Address Channel Read Address Ready. Indicates target is ready to accept the read address.<br>• 1 = Target read to accept address.<br>• 0 = Target not ready to accept address. |
| m_axi_rdata (C_M_AXI_DATA_ WIDTH-1: 0) | M_AXI | I | | Read Data Channel Read Data. |
| m_axi_rresp(1:0) | M_AXI | I | | Read Data Channel Response. Indicates results of the read transfer.<br>• 00b = OKAY - Normal access has been successful.<br>• 01b = EXOKAY - Not supported.<br>• 10b = SLVERR - Slave returned error on transfer.<br>• 11b = DECERR - Decode error, transfer targeted unmapped address. |
| m_axi_rlast | M_AXI | I | | Read Data Channel Last. Indicates the last data beat of a burst transfer.<br>• 1 = Last data beat.<br>• 0 = Not last data beat. |
| m_axi_rvalid | M_AXI | I | | Read Data Channel Data Valid. Indicates m_axi_rdata is valid.<br>• 1 = Valid read data.<br>• 0 = Not valid read data. |
| m_axi_rready | M_AXI | O | 0 | Read Data Channel Ready. Indicates the read channel is ready to accept read data.<br>• 1 = Ready.<br>• 0 = Not ready. |
| **CDMA Data AXI4 Write Master Interface Signals** | | | | |
| m_axi_awaddr (C_M_AXI_ADDR_ WIDTH-1: 0) | M_AXI | O | zeros | Write Address Channel Address Bus. |
| m_axi_awlen(7: 0) | M_AXI | O | zeros | Write Address Channel Burst Length. In data beats - 1. |
| m_axi_awsize(2: 0) | M_AXI | O | zeros | Write Address Channel Burst Size. Indicates with of burst transfer.<br>• 000b = Not Supported by AXI CDMA.<br>• 001b = Not Supported by AXI CDMA.<br>• 010b = 4 bytes (32 bit wide burst).<br>• 011b = 8 bytes (64 bit wide burst).<br>• 100b = 16 bytes (128 bit wide burst).<br>• 101b = 32 bytes (256 bit wide burst).<br>• 110b = Not Supported by AXI CDMA.<br>• 111b = Not Supported by AXI CDMA. |

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

| Signal Name | Interface | Signal Type | Init Status | Description |
|---|---|---|---|---|
| m_axi_awburst(1:0) | M_AXI | O | zeros | Write Address Channel Burst Type. Indicates type burst.<br>• 00b = FIXED - Not supported.<br>• 01b = INCR - Incrementing address.<br>• 10b = WRAP - Not supported.<br>• 11b = Reserved. |
| m_axi_awprot(2:0) | M_AXI | O | 000b | Write Address Channel Protection. This is always driven with a constant output of 000b. |
| m_axi_awcache(3:0) | M_AXI | O | 0011b | Write Address Channel Cache. This is always driven with a constant output of 0011b. |
| m_axi_awvalid | M_AXI | O | 0 | Write Address Channel Write Address Valid. Indicates when the Write Address Channel qualifiers are valid.<br>• 1 = Write Address is valid.<br>• 0 = Write Address is not valid. |
| m_axi_awready | M_AXI | I | | Write Address Channel Write Address Ready. Indicates target is ready to accept the write address.<br>• 1 = Target read to accept address.<br>• 0 = Target not ready to accept address. |
| m_axi_wdata (C_M_AXI_DATA_ WIDTH-1: 0) | M_AXI | O | zeros | Write Data Channel Write Data Bus. |
| m_axi_wstrb (C_M_AXI_DATA_ WIDTH/8 - 1: 0) | M_AXI | O | zeros | Write Data Channel Write Strobe Bus. Indicates which bytes are valid in the write data bus. This value is passed from the stream side strobe bus. |
| m_axi_wlast | M_AXI | O | 0 | Write Data Channel Last. Indicates the last data beat of a burst transfer.<br>• 1 = Last data beat.<br>• 0 = Not last data beat. |
| m_axi_wvalid | M_AXI | O | 0 | Write Data Channel Data Valid. Indicates m_axi_wdata is valid.<br>• 1 = Valid write data.<br>• 0 = Not valid write data. |
| m_axi_wready | M_AXI | I | | Write Data Channel Ready. Indicates the write channel target is ready to accept write data.<br>• 1 = Target is ready<br>• 0 = Target is not ready |
| m_axi_bresp(1:0) | M_AXI | I | | Write Response Channel Response. Indicates results of the write transfer.<br>• 00b = OKAY - Normal access has been successful.<br>• 01b = EXOKAY - Not supported.<br>• 10b = SLVERR - Slave returned error on transfer.<br>• 11b = DECERR - Decode error, transfer targeted unmapped address. |
| m_axi_bvalid | M_AXI | I | | Write Response Channel Response Valid. Indicates response, m_axi_bresp, is valid.<br>• 1 = Response is valid.<br>• 0 = Response is not valid. |

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

| Signal Name | Interface | Signal Type | Init Status | Description |
|---|---|---|---|---|
| m_axi_bready | M_AXI | O | 0 | Write Response Channel Ready. Indicates the write channel is ready to receive the AXI write response.<br>• 1 = Ready to receive response.<br>• 0 = Not ready to receive response. |
| **Scatter Gather AXI4 Read Master Interface Signals** | | | | |
| m_axi_sg_araddr (C_M_AXI_SG_ADDR_ WIDTH-1: 0) | M_AXI_SG | O | zeros | Scatter Gather Read Address Channel Address Bus. |
| m_axi_sg_arlen(7: 0) | M_AXI_SG | O | zeros | Scatter Gather Read Address Channel Burst Length. Length in data beats - 1. |
| m_axi_sg_arsize(2: 0) | M_AXI_SG | O | zeros | Scatter Gather Read Address Channel Burst Size. Indicates with of burst transfer.<br>• 000b = Not Supported by AXI CDMA SG Engine.<br>• 001b = Not Supported by AXI CDMA SG Engine.<br>• 010b = 4 bytes (32 bit wide burst).<br>• 011b = Not Supported by AXI CDMA SG Engine.<br>• 100b = Not Supported by AXI CDMA SG Engine.<br>• 101b = Not Supported by AXI CDMA SG Engine.<br>• 110b = Not Supported by AXI CDMA SG Engine.<br>• 111b = Not Supported by AXI CDMA SG Engine. |
| m_axi_sg_arburst(1:0) | M_AXI_SG | O | zeros | Scatter Gather Read Address Channel Burst Type. Indicates type burst.<br>• 00b = FIXED - Not supported.<br>• 01b = INCR - Incrementing address.<br>• 10b = WRAP - Not supported.<br>• 11b = Reserved. |
| m_axi_sg_arprot(2:0) | M_AXI_SG | O | 000b | Scatter Gather Read Address Channel Protection. This is always driven with a constant output of 000b. |
| m_axi_sg_arcache(3:0) | M_AXI_SG | O | 0011b | Scatter Gather Read Address Channel Cache. This is always driven with a constant output of 0011b. |
| m_axi_sg_arvalid | M_AXI_SG | O | 0 | Scatter Gather Read Address Channel Read Address Valid. Indicates if m_axi_sg_araddr is valid.<br>• 1 = Read Address is valid.<br>• 0 = Read Address is not valid. |
| m_axi_sg_arready | M_AXI_SG | I | | Scatter Gather Read Address Channel Read Address Ready. Indicates target is ready to accept the read address.<br>• 1 = Target read to accept address.<br>• 0 = Target not ready to accept address. |
| m_axi_sg_rdata (C_M_AXI_SG_DATA_ WIDTH-1: 0) | M_AXI_SG | I | | Scatter Gather Read Data Channel Read Data. |
| m_axi_sg_rresp(1:0) | M_AXI_SG | I | | Scatter Gather Read Data Channel Response. Indicates results of the read transfer.<br>• 00b = OKAY - Normal access has been successful.<br>• 01b = EXOKAY - Not supported.<br>• 10b = SLVERR - Slave returned error on transfer.<br>• 11b = DECERR - Decode error, transfer targeted unmapped address. |

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

| Signal Name | Interface | Signal Type | Init Status | Description |
|---|---|---|---|---|
| m_axi_sg_rlast | M_AXI_SG | I | | Scatter Gather Read Data Channel Last. Indicates the last data beat of a burst transfer.<br>• 1 = Last data beat.<br>• 0 = Not last data beat. |
| m_axi_sg_rdata | M_AXI_SG | I | | Scatter Gather Read Data Channel Data Valid. Indicates m_sg_aximry_rdata is valid.<br>• 1 = Valid read data.<br>• 0 = Not valid read data. |
| m_axi_sg_rready | M_AXI_SG | O | 0 | Scatter Gather Read Data Channel Ready. Indicates the read channel is ready to accept read data.<br>• 1 = Is ready.<br>• 0 = Is not ready. |
| **Scatter Gather AXI4 Write Master Interface Signals** | | | | |
| m_axi_sg_awaddr (C_M_AXI_SG_ADDR_ WIDTH-1: 0) | M_AXI_SG | O | zeros | Scatter Gather Write Address Channel Address Bus. |
| m_axi_sg_awlen(7: 0) | M_AXI_SG | O | zeros | Scatter Gather Write Address Channel Burst Length. Length in data beats - 1. |
| m_axi_sg_awsize(2: 0) | M_AXI_SG | O | zeros | Scatter Gather Write Address Channel Burst Size. Indicates with of burst transfer.<br>• 000b = Not Supported by AXI CDMA SG Engine.<br>• 001b = Not Supported by AXI CDMA SG Engine.<br>• 010b = 4 bytes (32 bit wide burst).<br>• 011b = Not Supported by AXI CDMA SG Engine.<br>• 100b = Not Supported by AXI CDMA SG Engine.<br>• 101b = Not Supported by AXI CDMA SG Engine.<br>• 110b = Not Supported by AXI CDMA SG Engine.<br>• 111b = Not Supported by AXI CDMA SG Engine. |
| m_axi_sg_awburst(1:0) | M_AXI_SG | O | zeros | Scatter Gather Write Address Channel Burst Type. Indicates type burst.<br>• 00b = FIXED - Not supported.<br>• 01b = INCR - Incrementing address.<br>• 10b = WRAP - Not supported.<br>• 11b = Reserved. |
| m_axi_sg_awprot(2:0) | M_AXI_SG | O | 000b | Scatter Gather Write Address Channel Protection. This is always driven with a constant output of 000b. |
| m_axi_sg_awcache(3:0) | M_AXI_SG | O | 0011b | Scatter Gather Write Address Channel Cache. This is always driven with a constant output of 0011b. |
| m_axi_sg_awvalid | M_AXI_SG | O | 0 | Scatter Gather Write Address Channel Write Address Valid. Indicates if m_axi_sg_awaddr is valid.<br>• 1 = Write Address is valid.<br>• 0 = Write Address is not valid. |
| m_axi_sg_awready | M_AXI_SG | I | | Scatter Gather Write Address Channel Write Address Ready. Indicates target is ready to accept the write address.<br>• 1 = Target ready to accept address.<br>• 0 = Target not ready to accept address. |

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

| Signal Name | Interface | Signal Type | Init Status | Description |
|---|---|---|---|---|
| m_axi_sg_wdata (C_M_AXI_SG_DATA_ WIDTH-1 : 0) | M_AXI_SG | O | zeros | Scatter Gather Write Data Channel Write Data Bus. |
| m_axi_sg_wstrb (C_M_AXI_SG_DATA_ WIDTH/8 - 1: 0) | M_AXI_SG | O | 1111b | Scatter Gather Write Data Channel Write Strobe Bus. All strobe bytes asserted for SG write address channel transfer requests. |
| m_axi_sg_wlast | M_AXI_SG | O | 0 | Scatter Gather Write Data Channel Last. Indicates the last data beat of a burst transfer.<br>• 1 = Last data beat.<br>• 0 = Not last data beat. |
| m_axi_sg_wvalid | M_AXI_SG | O | 0 | Scatter Gather Write Data Channel Data Valid. Indicates the Write Data Channel has a valid data beat on the bus.<br>• 1 = Valid write data.<br>• 0 = Not valid write data. |
| m_axi_sg_wready | M_AXI_SG | I | | Scatter Gather Write Data Channel Ready. Indicates the SG Write Data Channel target slave is ready to accept write data.<br>• 1 = Target slave is ready.<br>• 0 = Target slave is not ready. |
| m_axi_sg_bresp(1:0) | M_AXI_SG | I | | Scatter Gather Write Response Channel Response. Indicates results of the write transfer.<br>• 00b = OKAY - Normal access has been successful.<br>• 01b = EXOKAY - Not supported.<br>• 10b = SLVERR - Slave returned error on transfer.<br>• 11b = DECERR - Decode error, transfer targeted unmapped address. |
| m_axi_sg_bvalid | M_AXI_SG | I | | Scatter Gather Write Response Channel Response Valid. Indicates response, m_axi_sg_bresp, is valid.<br>• 1 = Response is valid.<br>• 0 = Response is not valid. |
| m_axi_sg_bready | M_AXI_SG | O | 0 | Scatter Gather Write Response Channel Ready. Indicates source is ready to receive response.<br>• 1 = Ready to receive response.<br>• 0 = Not ready to receive response. |

## Design Parameters

The AXI CDMA Design Parameters are listed and described in Table 2. In addition to the parameters listed in this table, there are also parameters that are inferred for each AXI interface in the EDK tools. Through the design, these EDK-inferred parameters control the behavior of the AXI Interconnect. For a complete list of the interconnect settings related to the AXI interface, see DS768, *AXI Interconnect IP Data Sheet*.

*Table 2:* **AXI CDMA Design Parameter Description**

| Feature/Description | Parameter Name | Allowable Values | Default Values | VHDL Type |
|---|---|---|---|---|
| **AXI CDMA General Parameters** | | | | |
| Specifies the target FPGA family | C_FAMILY | virtex6, spartan6 | virtex6 | String |
| **AXI CDMA AXI4-Lite Parameters** | | | | |
| Address width (in bits) of AXI4-Lite Interface. This is currently fixed at 32 bits. | C_S_AXI_LITE_ADDR_WIDTH | 32 | 32 | integer |
| Data width (in bits) of AXI4-Lite Interface. This is currently fixed at 32 bits. | C_S_AXI_LITE_DATA_WIDTH | 32 | 32 | integer |
| **AXI CDMA Data AXI4 Master Parameters** | | | | |
| Address width of the AXI4 master interface for the Data transfer path. | C_M_AXI_ADDR_WIDTH | 32 | 32 | integer |
| Data width of the AXI4 master interface for the Data transfer path. | C_M_AXI_DATA_WIDTH | 32, 64, 128, 256 | 32 | integer |
| Specifies the maximum burst length to be requested by the Data AXI4 master for both read and writes | C_M_AXI_MAX_BURST_LEN | 16, 32, 64, 128, 256 | 16 | integer |
| Specifies the inclusion or omission of the Data Realignment Engine (DRE) in the DataMover (can only be included for 32-bit and 64-bit data widths)<br>• 0 = Exclude DRE<br>• 1 = Include DRE | C_INCLUDE_DRE | 0,1 | 0 | integer |
| Specifies the use of a reduced resource version of the DataMover (can only be used for 32-bit and 64-bit data widths, no DRE, and burst lengths limited to 16, 32, and 64)<br>• 0 = Normal DataMover (full version)<br>• 1 = Reduced resource DataMover | C_USE_DATAMOVER_LITE | 0,1 | 0 | integer |
| **Scatter Gather Related Parameters** | | | | |
| Specifies the inclusion or omission of the Scatter Gather feature<br>• 0 = Exclude Scatter Gather<br>• 1 = Include Scatter Gather | C_INCLUDE_SG | 0,1 | 0 | integer |
| Address width (in bits) of AXI Scatter Gather AXI4 Master interface | C_M_AXI_SG_ADDR_WIDTH | 32 | 32 | integer |
| Data width (in bits) of AXI Scatter Gather AXI4 Master interface | C_M_AXI_SG_DATA_WIDTH | 32 | 32 | integer |
| Specifies the resolution of one tick of the SG interrupt delay timer in axi_aclk cycles | C_DLYTMR_RESOLUTION | 1 - 1000000 | 125 | integer |

## Allowable Parameter Combinations

*Table 3:* **Allowable Parameter Combination**

| Parameter Name | Affects Parameter | Relationship Description |
|---|---|---|
| C_USE_DATAMOVER_LITE | C_INCLUDE_DRE | Affected Parameter is ignored when C_USE_DATAMOVER_LITE = 1 |
| C_USE_DATAMOVER_LITE | C_M_AXI_MAX_BURST_LEN | Affected Parameter may only be assigned a value of 16, 32, or 64 when C_USE_DATAMOVER_LITE = 1 |
| C_USE_DATAMOVER_LITE | C_M_AXI_DATA_WIDTH | Affected Parameter may only be assigned a value of 32 or 64 when C_USE_DATAMOVER_LITE = 1 |
| C_USE_DATAMOVER_LITE | C_INCLUDE_SG | Affected Parameter is ignored when C_USE_DATAMOVER_LITE = 1 |
| C_INCLUDE_DRE | C_M_AXI_DATA_WIDTH | Affected Parameter may only be assigned a value of 32 or 64 when C_INCLUDE_DRE = 1 |
| C_INCLUDE_SG | C_DLYTMR_RESOLUTION | Affected Parameter is ignored when C_INCLUDE_SG = 0 |

## Parameter - I/O Signal Dependencies

*Table 4:* **Parameter - I/O Signal Dependencies**

| Parameter Name | Affects Signal | Depends on Parameter | Relationship Description |
|---|---|---|---|
| C_M_AXI_LITE_DATA_WIDTH | m_axi_lite_wdata<br>m_axi_lite_wstrb<br>m_axi_lite_rdata | | The setting of the parameter sets the vector width of the port. |
| C_M_AXI_LITE_ADDR_WIDTH | m_axi_lite_awaddr<br>m_axi_lite_araddr | | The setting of the parameter sets the vector width of the port. |
| C_M_AXI_SG_DATA_WIDTH | m_axi_sg_wdata<br>m_axi_sg_wstrb<br>m_axi_sg_rdata | | The setting of the parameter sets the vector width of the port. |
| C_M_AXI_SG_ADDR_WIDTH | m_axi_sg_awaddr<br>m_axi_sg_araddr | | The setting of the parameter sets the vector width of the port. |

*Table 4:* **Parameter - I/O Signal Dependencies *(Cont'd)***

| Parameter Name | Affects Signal | Depends on Parameter | Relationship Description |
|---|---|---|---|
| C_INCLUDE_SG | m_axi_sg_awaddr<br>m_axi_sg_awlen<br>m_axi_sg_awsize<br>m_axi_sg_awburst<br>m_axi_sg_awprot<br>m_axi_sg_awcache<br>m_axi_sg_awvalid<br>m_axi_sg_awready<br>m_axi_sg_wdata<br>m_axi_sg_wstrb<br>m_axi_sg_wlast<br>m_axi_sg_wvalid<br>m_axi_sg_wready<br>m_axi_sg_bresp<br>m_axi_sg_bvalid<br>m_axi_sg_bready<br>m_axi_sg_araddr<br>m_axi_sg_arlen<br>m_axi_sg_arsize<br>m_axi_sg_arburst<br>m_axi_sg_arprot<br>m_axi_sg_arcache<br>m_axi_sg_arvalid<br>m_axi_sg_arready<br>m_axi_sg_rdata<br>m_axi_sg_rresp<br>m_axi_sg_rlast<br>m_axi_sg_rvalid<br>m_axi_sg_rready | | If the parameter is assigned a value of zero, the SG Interface output ports are tied to 0, and the SG Interface input ports are left open. |
| C_M_AXI_ADDR_WIDTH | m_axi_araddr<br>m_axi_awaddr | | The setting of the parameter sets the vector width of the port. |
| C_M_AXI_DATA_WIDTH | m_axi_rdata<br>m_axi_wdata<br>m_axi_wstrb | | The setting of the parameter sets the vector width of the port. |
| C_USE_DATAMOVER_LITE | m_axi_rdata<br>m_axi_wdata<br>m_axi_wstrb | | The setting of the affecting parameter to 1 limits the vector width of the effected ports.<br>m_axi_rdata = 32 or 64<br>m_axi_wdata= 32 or 64<br>m_axi_wstrb = 4 or 8 |

## Parameter Descriptions

### C_FAMILY

- **Type:** string
- **Allowed Values:** Spartan-6 and Virtex-6 and later
- **Definition:** Indicates the target FPGA device family for the design
- **Description:** This parameter is set by the EDK tools to a value reflecting the FPGA device family selected for the EDK project.

### C_S_AXI_LITE_ADDR_WIDTH

- **Type:** Integer
- **Allowed Values:** 32 (default = 32)
- **Definition:** Address bus width of the AXI4-Lite interface
- **Description:** This integer parameter is used by the AXI4-Lite interface to size the read and write address channel related components. The EDK tool suite will assign this parameter a fixed value of 32.

### C_S_AXI_LITE_DATA_WIDTH

- **Type:** Integer
- **Allowed Values:** 32 (default = 32)
- **Definition:** Data bus width of the AXI4-Lite interface
- **Description:** This integer parameter is used by the AXI4-Lite interface to size the read and write data channel related components within the Lite interface. The EDK tool suite will assign this parameter a fixed value of 32.

### C_INCLUDE_DRE

- **Type:** Integer
- **Allowed Values:** 0,1 (default = 1)
- **Definition:** 0 = Exclude Data Realignment Engine; 1 = Include Data Realignment Engine
- **Description:** Include or exclude the Data Realignment Engine. For use cases where all transfers will be C_M_AXI_DATA_WIDTH aligned, this parameter can be set to 0 to exclude DRE, saving FPGA resources. Setting this parameter to 1 allows data realignment to the byte (8 bits) address resolution on the data transport AXI4 interface. DRE is only supported for C_M_AXI_DATA_WIDTH = 32 and C_M_AXI_DATA_WIDTH = 64.

  When DRE is included, CDMA data reads can start from any address byte offset (the transfer source address). DRE will realign the read data to match the starting offset of the programmed destination address.

  ***Note:*** If DRE is disabled (C_INCLUDE_DRE = 0) or DRE does not support the specified data width (C_M_AXI_DATA_WIDTH = 128 or 256), the offset of the transfer source address must match the offset of the transfer destination address. 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 are the offset. Note that the number of address bits used for the offset varies with the transfer bus data width.

### C_M_AXI_ADDR_WIDTH

- **Type:** Integer
- **Allowed Values:** 32 (default = 32)
- **Definition:** Address bus width of the data transport AXI4 master interface
- **Description:** This integer parameter is used to size the address bus for the data transport AXI4 master interface. The EDK tool suite assigns this parameter a fixed value of 32.

### C_M_AXI_ADDR_WIDTH

- **Type:** Integer
- **Allowed Values:** 32 (default = 32)
- **Definition:** Address Channel width of the AXI CDMA AXI4 data transport interface
- **Description:** This integer parameter is used to size the AXI CDMA AXI4 data transport address channel related qualifiers. The EDK tool suite assigns this parameter a fixed value of 32.

### C_M_AXI_DATA_WIDTH

- **Type:** Integer
- **Allowed Values:** 32, 64, 128, 256 (default = 32)
- **Definition:** Data Channel width of the AXI CDMA AXI4 data transport interface
- **Description:** This integer parameter is used to size the AXI CDMA AXI4 data transport Data Channel related qualifiers.

### C_M_AXI_MAX_BURST_LEN

- **Type:** Integer
- **Allowed Values:** 16, 32, 64, 128, 256 (default = 16)
- **Definition:** Maximum burst length used (in data beats) by AXI CDMA for data transfers
- **Description:** This parameter limits the burst length requested by CDMA on the AXI4 data transport interface.

### C_USE_DATAMOVER_LITE

- **Type:** Integer
- **Allowed Values:** 0,1 (default = 0)
- **Definition:** 0 = Use Full DataMover; 1 = Use DataMover Lite
- **Description:** This parameter allows the DataMover used in the main CDMA data transport path to be implemented in a “lite” mode. DataMover Lite is useful for resource limited designs that requires a smaller resource utilization traded off for high performance data transfer.

### C_INCLUDE_SG

- **Type:** Integer
- **Allowed Values:** 0,1 (default = 0)
- **Definition:** 0 = Exclude Scatter Gather; 1 = Include Scatter Gather
- **Description:** Include or exclude the Scatter Gather support feature. When set to 0, only Simple Mode DMA operations are supported by AXI CDMA. When set to 1, both Simple and Scatter Gather assisted transfers are supported.

### C_M_AXI_SG_ADDR_WIDTH

- **Type:** Integer
- **Allowed Values:** 32 (default = 32)
- **Definition:** Address bus width of attached AXI on the AXI Scatter Gather interface
- **Description:** This integer parameter is used to size the Read Address and Write Address Channels of the AXI4 Scatter Gather interface. The EDK tool suite will assign this parameter a fixed value of 32.

### C_M_AXI_SG_DATA_WIDTH

- **Type:** Integer
- **Allowed Values:** 32 (default = 32)
- **Definition:** Data bus width of attached AXI on the AXI Scatter/Gather interface
- **Description:** This integer parameter is used to size the Read Data and Write Data Channels of the AXI4 Scatter Gather interface. The EDK tool suite will assign this parameter a fixed value of 32.

### C_DLYTMR_RESOLUTION

- **Type:** Integer
- **Allowed Values:** 1 to 100,000 (default = 256)
- **Definition:** Interrupt Delay Timer Resolution in 'axi_aclk' cycles
- **Description:** This integer parameter is used to set the resolution of the Interrupt Delay Timer. The value assigned specifies the number of the input 'axi_aclk' clock cycles between each tick of the delay timer. The Delay Timer is only used during Scatter Gather operations. For additional information, see the section SG Delay Interrupt, page 38.

## Register Space

The AXI CDMA core register space is summarized in Table 5. The AXI CDMA Registers are memory-mapped into non-cacheable memory space. The registers are 32 bits wide and the register memory space must be aligned on 128-byte (80h) boundaries.

## Register Address Mapping

*Table 5:* **AXI CDMA Register Summary**

| Address Space Offset[1] | Name | Description |
|---|---|---|
| 00h | CDMACR | CDMA Control Register |
| 04h | CDMASR | CDMA Status Register |
| 08h | CURDESC_PNTR | Current Descriptor Pointer Register |
| 0Ch | Reserved | N/A |
| 10h | TAILDESC_PNTR | Tail Descriptor Pointer Register |
| 14h | Reserved | N/A |
| 18h | SA | Source Address Register |
| 1Ch | Reserved | N/A |
| 20h | DA | Destination Address Register |
| 24h | Reserved | N/A |
| 28h | BTT | Bytes to Transfer Register |

1. Address Space Offset is relative to C_BASEADDR assignment. C_BASEADDR is defined in AXI CDMA MPD file and set by XPS.

### Endianess

All registers are in Little Endian format, as shown in Figure 3.

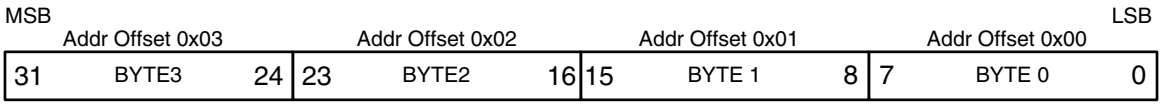


*Figure 3:* **32-bit Little Endian Example**

## Register Detail

### CDMACR (CDMA Control Register - Offset 00h)

This register provides software application control of the AXI CDMA.

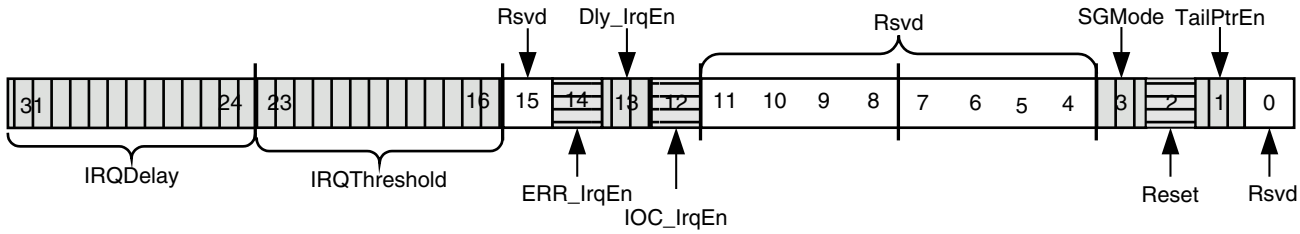


*Figure 4:* **CDMACR Register**

*Table 6:* **CDMACR Register Details**

| Bits | Field Name | Default Value | Access Type | CDMA Mode Used | Description |
|---|---|---|---|---|---|
| 31 to 24 | IRQDelay | 00h | R/W | SG | Interrupt Delay Time Out. This value is used for setting the interrupt delay time out value. The interrupt time out is a mechanism for causing the CDMA engine to generate an interrupt after the delay time period has expired. This is used for cases when the interrupt threshold is not met after a period of time, and the CPU desires an interrupt to be generated. Timer begins counting when the CDMA is IDLE (CDMACR.Idle = '1'). This generally occurs when the CDMA has completed all scheduled work defined by the transfer descriptor chain (reached the tail pointer) and has not satisfied the Interrupt Threshold count.<br>***Note:*** Setting this value to zero will disable the delay timer interrupt. |
| 23 to 16 | IRQThreshold | 01h | R/W | SG | Interrupt Threshold value. This field is used for setting the Scatter Gather interrupt coalescing 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 CDMA engine.<br>***Note:*** The minimum setting for the threshold is 0x01. A write of 0x00 to this register will have no effect. If the CDMA is built with SG disabled (Simple Mode Only), the default value of the port is zeros. |
| 15 | Reserved | 0 | RO | N/A | Writing to this bit has no effect and it will always read as zeros. |
| 14 | Err_IrqEn | 0 | R/W | Simple & SG | Interrupt on Error Interrupt Enable. When set to 1, will allow the CDMASR.Err_Irq to generate an interrupt out.<br>• 0 = Error Interrupt disabled<br>• 1 = Error Interrupt enabled |

*Table 6:* **CDMACR Register Details *(Cont'd)***

| Bits | Field Name | Default Value | Access Type | CDMA Mode Used | Description |
|---|---|---|---|---|---|
| 13 | Dly_IrqEn | 0 | R/W | SG | Interrupt on Delay Timer Interrupt Enable. When set to 1, will allow CDMASR.Dly_Irq to generate an interrupt out. This is only used with Scatter Gather assisted transfers.<br>• 0 = Delay Interrupt disabled<br>• 1 = Delay Interrupt enabled<br>***Note:*** This bit is cleared whenever CDMACR.SGMode is set to '0'. |
| 12 | IOC_IrqEn | 0 | R/W | Simple & SG | Interrupt on Complete Interrupt Enable. When set to 1, will allow CDMASR.IOC_Irq to generate an interrupt out for completed DMA transfers.<br>• 0 = IOC Interrupt disabled<br>• 1 = IOC Interrupt enabled |
| 11 to 4 | Reserved | 0 | RO | N/A | Writing to these bits has no effect, and they will always read as zeros. |
| 3 | SGMode | 0 | RW | Simple & 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 (C_INCLUDE_SG = 1).<br>• 0 = Simple DMA Mode<br>• 1 = Scatter Gather Mode (Only valid if C_INCLUDE_SG = 1)<br>***Note:*** 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 will have undefined results.<br>***Note:*** 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.<br>***Note:*** This bit must be set prior to setting the CDMACR.Dly_IrqEn bit. Otherwise, the CDMACR.Dly_IrqEn bit will not get set. |
| 2 | Reset | 0 | RW | Simple & SG | Soft reset control for the AXI CDMA core. Setting this bit to a 1 causes the AXI CDMA to be reset. Reset will be accomplished gracefully. Committed AXI4 transfers will be completed. Other queued transfers will be flushed. After completion of a soft reset, all registers and bits will be in the Reset State.<br>• 0 = Reset NOT in progress - Normal operation.<br>• 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 will always read as zeros. |
| RO = Read Only. Writing has no effect.<br>R/W = Read / Write. | | | | | |

### CDMASR (CDMA Status Register- Offset 04h)

This register provides status for the AXI CDMA.

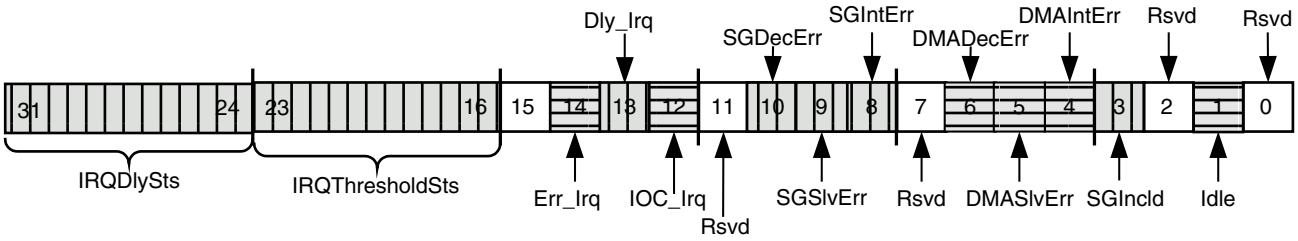


*Figure 5:* **CDMASR Register**

*Table 7:* **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 & 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 will be generated from the AXI CDMA.<br>0 = No error Interrupt.<br>1 = Error interrupt active. |
| 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 time out. If the corresponding enable bit is set (CDMACR.Dly_IrqEn = 1), an interrupt out will be generated from the AXI CDMA.<br>• 0 = No Delay Interrupt.<br>• 1 = Delay Interrupt active.<br>***Note:*** This bit is cleared whenever CDMACR.SGMode is set to '0'. |
| 12 | IOC_Irq | 0 | R/WC | Simple & 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 will be generated from the AXI CDMA.<br>• 0 = No IOC Interrupt.<br>• 1 = IOC Interrupt active.<br>***Note:*** When operating in SG mode, the criteria specified by the interrupt threshold must also be met. |
| 11 | Reserved | 0 | RO | N/A | Writing to this bit has no effect, and it will always read as zeros. |

*Table 7:* **CDMASR Register Details *(Cont'd)***

| Bits | Field Name | Default Value | Access Type | CDMA Mode Used | Description |
|---|---|---|---|---|---|
| 10 | SGDecErr | 0 | RO | SG | **Scatter Gather Decode Error**. This bit indicates that a AXI decode error has been received by the SG Engine during a AXI transfer (transfer descriptor read or write). This error occurs if the SG Engine issues an address that does not have a mapping assignment to a slave device. This error condition will cause the AXI CDMA to gracefully halt. The CDMASR.Idle bit will be set to 1 when the CDMA has completed shut down.<br>• 0 = No SG Decode Errors.<br>• 1 = SG Decode Error detected. CDMA Engine will halt.<br>***Note:*** In SG Mode, the CURDESC_PNTR register will be updated with the descriptor pointer value when this error is detected. If multiple errors are detected, the errors will be logged in the CDMASR, but only one address will be updated to the CURDESC_PNTR. 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 a AXI slave error response has been received by the SG Engine during a AXI transfer (transfer descriptor read or write). This error condition will cause the AXI CDMA to gracefully halt. The CDMASR.Idle bit will be set to 1 when the CDMA has completed shut down.<br>• 0 = No SG Slave Errors.<br>• 1 = SG Slave Error detected. CDMA Engine will halt.<br>***Note:*** In SG Mode, the CURDESC_PNTR register will be updated with the descriptor pointer value when this error is detected. If multiple errors are detected, the errors will be logged in the CDMASR, but only one address will be updated to the CURDESC_PNTR. A reset (soft or hard) must be issued to clear the error condition. |
| 8 | SGIntErr | 0 | RO | SG | **Scatter Gather Internal Error**.This bit indicates that a internal error has been encountered by the SG Engine. This error condition will cause the AXI CDMA to gracefully halt. The CDMASR.Idle bit will be set to 1 when the CDMA has completed shut down.<br>• 0 = No SG Internal Errors.<br>• 1 = SG Internal Error detected. CDMA Engine will halt.<br>***Note:*** In SG Mode, the CURDESC_PNTR register will be updated with the descriptor pointer value when this error is detected. If multiple errors are detected, the errors will be logged in the CDMASR, but only one address will be updated to the CURDESC_PNTR. 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 will always read as zeros. |

*Table 7:* **CDMASR Register Details *(Cont'd)***

| Bits | Field Name | Default Value | Access Type | CDMA Mode Used | Description |
|---|---|---|---|---|---|
| 6 | DMADecErr | 0 | RO | Simple & 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 that does not have a mapping assignment to a slave device. This error condition will cause the AXI CDMA to gracefully halt. The CDMASR.Idle bit will be set to 1 when the CDMA has completed shut down.<br>• 0 = No CDMA Decode Errors.<br>• 1 = CDMA Decode Error detected. CDMA Engine will halt.<br>***Note:*** In SG Mode, the CURDESC_PNTR register will be updated with the descriptor pointer value when this error is detected. If multiple errors are detected, the errors will be logged in the CDMASR, but only one address will be updated to the CURDESC_PNTR. A reset (soft or hard) must be issued to clear the error condition. |
| 5 | DMASlvErr | 0 | RO | Simple & SG | **DMA Slave Error**. This bit indicates that a AXI slave error response has been received by the AXI DataMover during a AXI transfer (read or write). This error condition will cause the AXI CDMA to gracefully halt. The CDMASR.Idle bit will be set to 1 when the CDMA has completed shut down.<br>• 0 = No CDMA Slave Errors.<br>• 1 = CDMA Slave Error detected. CDMA Engine will halt.<br>***Note:*** In SG Mode, the CURDESC_PNTR register will be updated with the descriptor pointer value when this error is detected. If multiple errors are detected, the errors will be logged in the CDMASR, but only one address will be updated to the CURDESC_PNTR. A reset (soft or hard) must be issued to clear the error condition. |
| 4 | DMAIntErr | 0 | RO | Simple & 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 will cause the AXI CDMA to gracefully halt. The CDMASR.Idle bit will be set to 1 when the CDMA has completed shut down.<br>• 0 = No CDMA Internal Errors.<br>• 1 = CDMA Internal Error detected. CDMA Engine will halt.<br>***Note:*** In SG Mode, the CURDESC_PNTR register will be updated with the descriptor pointer value when this error is detected. If multiple errors are detected, the errors will be logged in the CDMASR, but only one address will be updated to the CURDESC_PNTR. A reset (soft or hard) must be issued to clear the error condition. |

*Table 7:* **CDMASR Register Details *(Cont'd)***

| Bits | Field Name | Default Value | Access Type | CDMA Mode Used | Description |
|---|---|---|---|---|---|
| 3 | SGIncld | See Description | RO | Simple & SG | **SG Included.** This bit indicates if the AXI CDMA has been implemented with Scatter Gather support included (C_SG_ENABLE = 1). This is used by application software (drivers) to determine if SG Mode can be utilized.<br>• 0 = Scatter Gather not included. Only Simple DMA operations are supported.<br>• 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 will always read as zeros. |
| 1 | Idle | 1 | RO | Simple & SG | **CDMA Idle**. Indicates the state of AXI CDMA operations.<br>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 BTT register in Simple DMA mode causes the CDMA to start (not Idle).<br>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 will automatically restart CDMA SG operations.<br>• 0 = Not Idle - Simple or SG DMA operations are in progress.<br>• 1 = Idle - Simple or SG operations completed or not started. |
| 0 | Reserved | 0 | RO | SG | Writing to these bits has no effect, and they will always read as zeros. |
| RO = Read Only. Writing has no effect<br>R/WC = Read / Write to Clear. A CPU write of 1 will clear the associated bit to 0. | | | | | |

### CURDESC_PNTR (CDMA Current Descriptor Pointer Register- Offset 08h)

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


*Figure 6:* **CURDESC_PNTR Register**

*Table 8:* **CURDESC_PNTR Register Details**

| Bits | Field Name | Default Value | Access Type | CDMA Mode Used | Description |
|---|---|---|---|---|---|
| 31 to 6 | Current Descriptor Pointer | zeros | R/W (RO) | SG | **Current Descriptor Pointer**. This register field is written by the software application (in SG Mode) to set the starting address of the first transfer descriptor to execute for a SG operation. The address written corresponds to a 32-bit system address with the least significant 6 bits truncated. This register field must contain a valid descriptor address prior to the software application writing the CDMA TAILDESC_PNTR register value. Failure to do so will result in undefined operation by the CDMA.<br>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.<br>On error detection, the CURDESC_PNTR register will be updated to reflect the descriptor associated with the detected error.<br>***Note:*** The register can only be written to by the software application when the AXi CDMA is Idle (CDMASR.Idle=1). At all other times, this register is Read Only (RO). Descriptor addresses written to this field must be aligned to 64-byte boundaries (16 32-bit words). Examples are 0x00, 0x40, 0x80, etc. Any other alignment will have undefined results. |
| 5 to 0 | Reserved | 0 | RO | N/A | Writing to these bits has no effect, and they will always read as zeros. |

RO = Read Only. Writing has no effect.
R/WC = Read / Write to Clear. A write of 1 will clear the associated bit to 0.

### TAILDESC_PNTR (CDMA Tail Descriptor Pointer Register- Offset 10h)

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

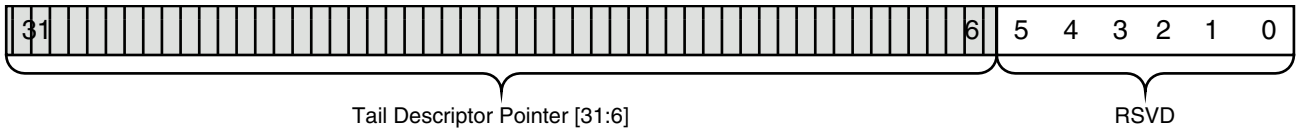


*Figure 7:* **TAILDESC_PNTR Register**

*Table 9:* **TAILDESC_PNTR Register Details**

| Bits | Field Name | Default Value | Access Type | CDMA Mode Used | Description |
|---|---|---|---|---|---|
| 31 to 6 | Tail Descriptor Pointer | zeros | R/W (RO) | SG | **Tail Descriptor Pointer**. This register field is written by the software application (in SG Mode) to set the current pause pointer for descriptor chain execution. The AXI CDMA SG Engine will pause descriptor fetching after completing operations on the descriptor whose current descriptor pointer matches the tail descriptor pointer.<br>When the AXI CDMA is in SG Mode (CDMACR.SGMode = 1), a write by the software application to the TAILDESC_PNTR register will cause the AXI CDMA SG Engine to start fetching descriptors starting from the CURDESC_PNTR register value. If the SG engine is paused at a tailpointer pause point, the SG engine will restart descriptor execution at the next sequential transfer descriptor. If the AXI CDMA is not idle (CDMASR.Idle = 0), writing to the TAILDESC_PNTR will have no effect except to reposition the SG pause point.<br>***Note:*** The software application must not move the tail pointer to a location that has not been updated with valid transfer descriptors. The software application must process and reallocate all completed descriptors (Descriptor Status.Cmplt = 1), clear the completed bits and then move the tail pointer. The software application must move the pointer to the last descriptor address it has updated. |
| 5 to 0 | Reserved | 0 | RO | N/A | Writing to these bits has no effect, and they will always read as zeros. |
| RO = Read Only. Writing has no effect<br>R/WC = Read / Write to Clear - A CPU write of 1 will clear the associated bit to 0. | | | | | |

### SA (CDMA Source Address Register - Offset 18h)

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


*Figure 8:* **SA Register**

*Table 10:* **SA Register Details**

| Bits | Field Name | Default Value | Access Type | CDMA Mode Used | Description |
|---|---|---|---|---|---|
| 31 to 0 | SA | Zeros | 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.<br>***Note:*** The software application should only write to this register when the AXi CDMA is Idle (CDMASR.Idle=1). |
| RO = Read Only. Writing has no effect.<br>R/W = Read / Write. | | | | | |

## DA (CDMA Destination Address Register- Offset 20h)

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


*Figure 9:* **DA Register**

*Table 11:* **DA Register Details**

| Bits | Field Name | Default Value | Access Type | CDMA Mode Used | Description |
|---|---|---|---|---|---|
| 31 to 0 | DA | Zeros | 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 DRE inclusion as follows.<br>If DRE is not included in the AXI CDMA (C_INCLUDE_DRE = 0) or the DMA data width is 128 or 256 bits (C_M_AXI_DATA_WIDTH = 128 or 256), 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 are the offset. Note that the number of address bits used for the offset varies with the transfer bus data width,<br>***Note:*** The software application should only write to this register when the AXi CDMA is Idle (CDMASR.Idle=1). |
| RO = Read Only. Writing has no effect.<br>R/WC = Read / Write to Clear. A CPU write of 1 will clear the associated bit to 0. | | | | | |

### BTT (CDMA Bytes to Transfer Register- Offset 28h)

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

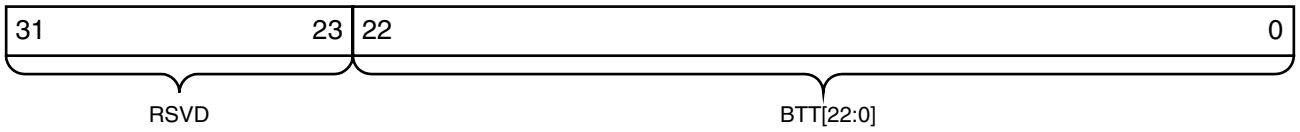


*Figure 10:* **BTT Register**

*Table 12:* **BTT Register Details**

| Bits | Field Name | Default Value | Access Type | Description |
|---|---|---|---|---|
| 31 to 23 | Reserved | Zeros | RO | Writing to these bits has no effect, and they will always read as zeros. |
| 22 to 0 | BTT | zeros | 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 8,388,607 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 will cause 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). |
| RO = Read Only. Writing has no effect.<br>R/W = Read / Write. | | | | |

## DataMover 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 the CDMA resource utilizations in Table 21 and Table 22). Using the DataMover Lite operation mode puts restrictions on the available CDMA features. Below is a list of feature restrictions (compared to the Full mode).

- The AXI CDMA data transport width is restricted to 32 and 64 bits.
  - C_M_AXI_DATA_WIDTH = 32 or 64 only
- The Max Burst Length is restricted to 16, 32, and 64 data beats.
  - C_M_AXI_MAX_BURST_LEN = 16, 32, or 64 only
- 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.
  - C_M_AXI_MAX_BURST_LEN * (C_M_AXI_DATA_WIDTH/8)
- The DRE function is not supported with Data Mover Lite.
  - C_INCLUDE_DRE must be set to 0
- The AXI4 4k address crossing guard is not provided in the CDMA. 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.

## Simple DMA Operation Mode

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

### Simple DMA Programming Example

The following section describes the basic steps to setup 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.
4. Write the desired transfer destination address to the Destination Address (DA) Register.
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 will also start 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 Operation Mode

Scatter Gather is a mechanism that allows for automated DMA transfer scheduling via a pre-programmed instruction list of transfer descriptors (See "Scatter Gather Transfer Descriptor Definition" on page 29.). This instruction list is programmed by the user 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. The AXI CDMA Tail Descriptor Pointer register needs to be programmed with the address of the first word of the last descriptor of the chain. When the AXI CDMA executes the last descriptor and finds that the Tail Descriptor pointer matches the address of the completed descriptor, the SG Engine will stop descriptor fetching and wait (SG is paused).

### Transfer Descriptor Management

Prior to starting CDMA Scatter Gather operations, the software application must set up a transfer descriptor chain. Once the AXI CDMA begins SG operations, it will fetch, process, and then update 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 descriptors and DMA data buffers.

The software applications should process each DMA data buffer associated with completed descriptor and reallocate the descriptor for AXI CDMA use. To prevent software and hardware from modifying the same descriptor, a Tail Pointer Mode was created. The tail pointer is initialized by software to point to the end of the descriptor chain. This becomes the pause point for hardware. When hardware begins running, it will fetch and process each descriptor in the chain until it reaches the tail pointer. The AXI CDMA will then pause descriptor processing. The software will typically then process and re-allocate any descriptor with the Complete bit set to 1. While the software is processing descriptors, AXI CDMA hardware is prevented from modifying the descriptors being processed by software by the tail pointer. When the software finishes re-allocating a set of descriptors, it then moves the tail pointer location to the end of the re-allocated descriptors by writing to the AXI CDMA TAILDESC register. The act of writing to the TAILDESC register will cause 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 will have no effect on the hardware. In this situation, the AXI CDMA will simply continue to process descriptors until reaching the new tail descriptor pointer location.

## Scatter Gather Transfer Descriptor Definition

This section 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 by assigning the parameter C_INCLUDE_SG a value of 1 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 via the address linkage built into the descriptor format. The AXI CDMA SG Engine will traverse 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 13.

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

*Table 13:* **Transfer Descriptor Word Summary**

| Address Space Offset(1) | Name | Description |
|---|---|---|
| 00h | NXTDESC_PNTR | Next Descriptor Pointer |
| 04h | RESERVED | N/A |
| 08h | SA | Source Address |
| 0Ch | RESERVED | N/A |
| 10h | DA | Destination Address |
| 14h | RESERVED | N/A |
| 18h | CONTROL | Transfer Control |
| 1Ch | STATUS | Transfer Status |

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.

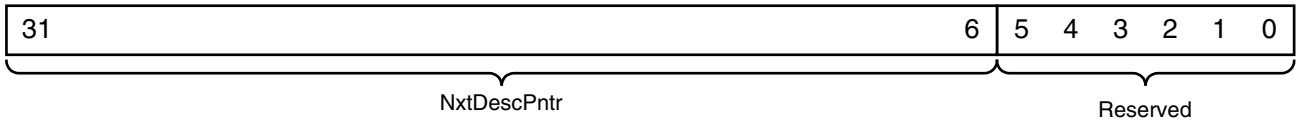


*Figure 11:* **Transfer Descriptor NXTDESC_PNTR Word**

*Table 14:* **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.<br>***Note:*** 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 SA Word (Source Address - Offset 08h)

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

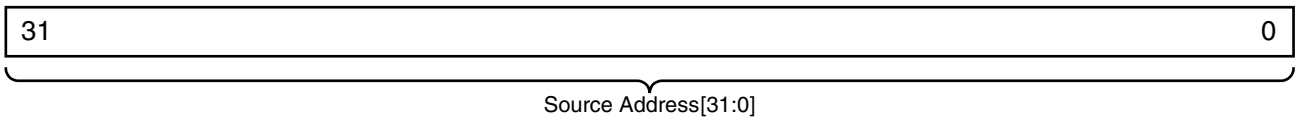


*Figure 12:* **Transfer Descriptor SA Word**

*Table 15:* **Transfer Descriptor SA Word Details**

| Bits | Field Name | Description |
|---|---|---|
| 31 to 0 | DA | **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 DA Word (Destination Address - Offset 10h)

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

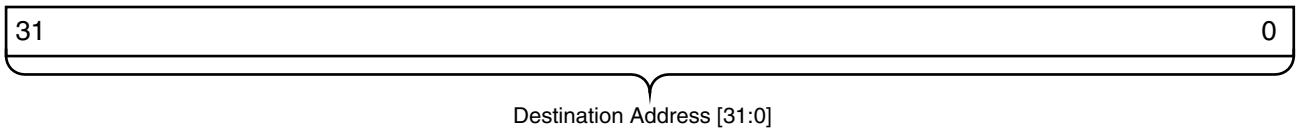


*Figure 13:* **Transfer Descriptor DA Word**

*Table 16:* **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.<br>***Note:*** If DRE is not included in the AXI CDMA (C_INCLUDE_DRE = 0) or the specified CDMA data width is not supported by DRE (C_M_AXI_DATA_WIDTH = 128 or 256), then the address offset of the Destination Address **must** match that of the Source Address value. |

### Transfer Descriptor CONTROL Word (Control - Offset 18h)

This value provides control for the AXI CDMA transfer.

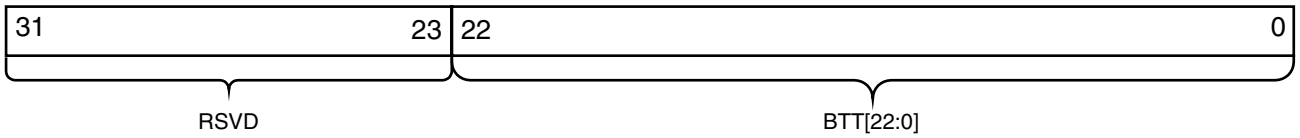


*Figure 14:* **Transfer Descriptor CONTROL Word**

*Table 17:* **Transfer Descriptor CONTROL Word Details**

| Bits | Field Name | Description |
|---|---|---|
| 31 to 23 | Reserved | These bits are reserved and should be set to zero. |
| 22 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 8,388,607 bytes of data can be specified by this field for the associated transfer.<br>***Note:*** A value of zero (0) is not allowed and will cause a DMA internal error to be set by AXI CDMA. |

### Transfer Descriptor STATUS Word (Status - Offset 1Ch)

This value provides 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 will be updated and written back to the original Status word location in memory by the SG Engine.

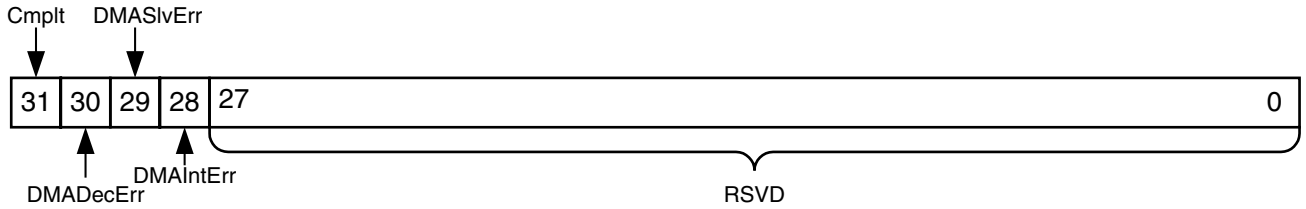


*Figure 15:* **Transfer Descriptor STATUS Word**

*Table 18:* **Transfer Descriptor STATUS Word Details**

| Bits | Field Name | Description |
|---|---|---|
| 31 | Cmplt | **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 may manipulate any descriptor with the Completed bit set to 1 when in Tail Pointer Mode (currently the only supported mode).<br>• 0 = Descriptor not completed.<br>• 1 = Descriptor completed.<br>***Note:*** If the CDMA SG Engine fetches a descriptor is with this bit set to 1, the descriptor is considered a stale descriptor. An SGIntErr will be flagged in the AXI CDMA Status Register and the AXI CDMA engine will halt 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 will cause the AXI CDMA to gracefully halt.<br>• 0 = No CDMA Decode Errors.<br>• 1 = CDMA Decode Error received. CDMA Engine will halt at this descriptor. |
| 29 | DMASlvErr | **DMA Slave Error**. This bit indicates that a 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 will cause the AXI CDMA to gracefully halt.<br>• 0 = No CDMA Slave Errors.<br>• 1 = CDMA Slave Error received. CDMA Engine will halt 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 will cause the AXI CDMA to gracefully halt. The CDMASR.Idle bit will be set to 1 when the CDMA has completed shut down.<br>• 0 = No CDMA Internal Errors.<br>• 1 = CDMA Internal Error detected. CDMA Engine will halt at this descriptor. |
| 27 to 0 | Reserved | These bits are reserved and should be set to zero. |

## Example Transfer Descriptor Usage

The following example illustrates a typical procedure for using the AXI CDMA Scatter Gather feature. The example transfer descriptor chain contains ten transfer descriptors as shown in Figure 16.

### SG Initialization Sequence

- The software application verifies that the CDMA is idle (CDMASR.Idle=1).
- The transfer descriptors are initialized in the system memory by the software application. The descriptors are arranged into a ring with each NXTDESC_PNTR word pointing to the starting memory address of the next sequential transfer descriptor. Each transfer descriptor describes a single DMA operation via the Source Address (SA) word, the Destination Address (DA) word, and the BTT field in the Control word.
- The software application writes the CDMA Control Register (CDMACR) with the appropriate controls set. This must include setting the CDMACR.SGMode = '1'. IRQDelay and IRQThreshold may be overridden if desired. Interrupt enabling can be set, if desired.
- The software application initializes the CDMA CURDESC_PNTR register with the starting memory address of the first transfer descriptor in the chain (address of TD1 NXTDESC_PNTR word).
- The software application initializes the CDMA TAILDESC_PNTR register with the starting memory address of the last transfer descriptor in the chain (TD10). Writing to this register will also start the CDMA operations

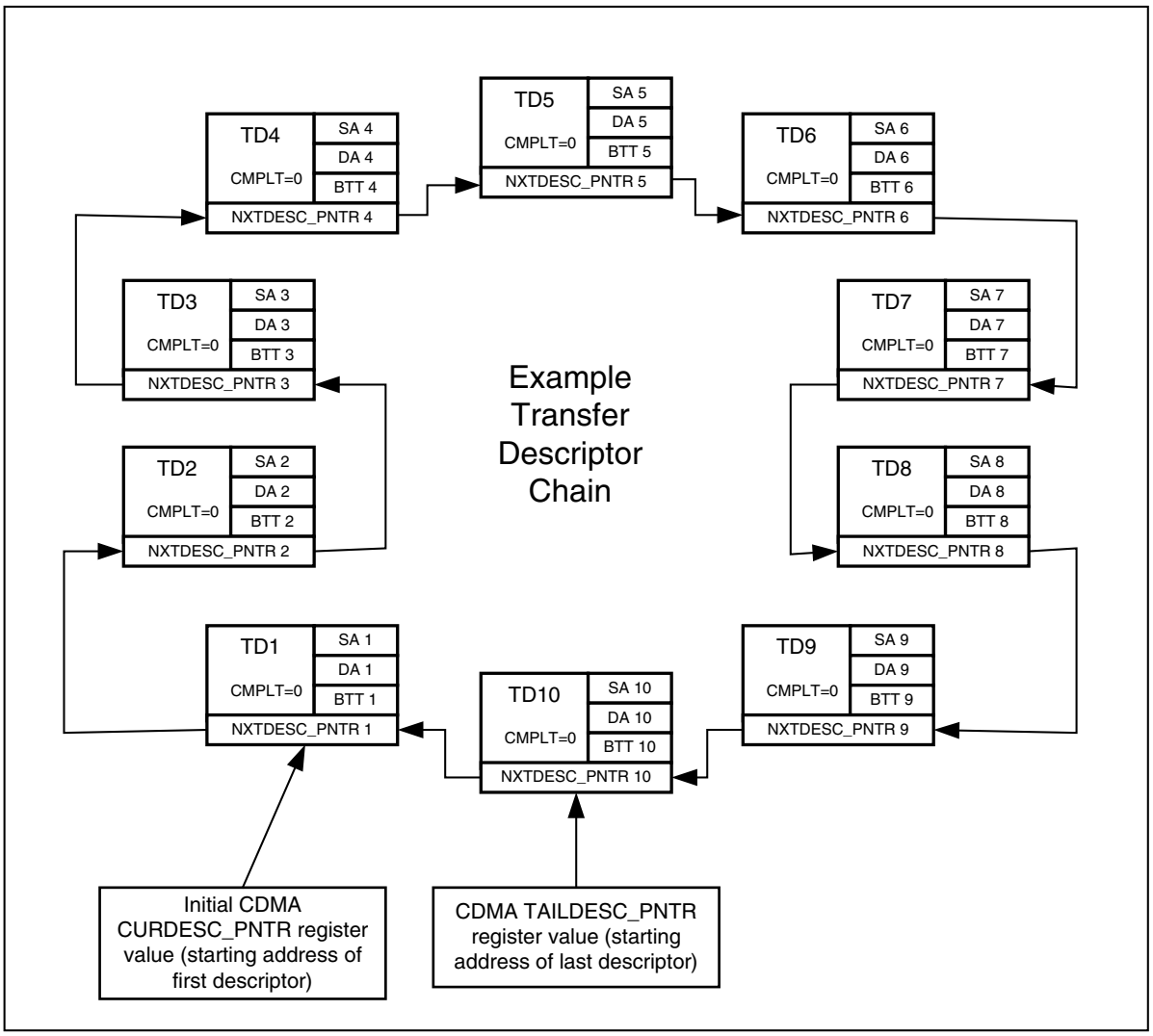


*Figure 16:* **Transfer Descriptor Chain Initialization**

### Transfer Descriptor Execution Startup

- The AXI CDMA will start SG operations when the TAILDESC_PNTR register is written by the software application. The value written should be the starting address of the last TD in the chain to be executed. The SG operation will begin by the CDMA fetching the first transfer descriptor at the address loaded into the CDMA CURDESC_PNTR register. See Figure 17.
- The AXI CDMA will sequentially fetch descriptors until the internal scatter gather fetch queue is full, as indicated by the shaded boxes or the TD at the TAILDESC_PNTR register address is completed.
- In sequential order, the CDMA pulls descriptors from the fetch queue and programs the DataMover with each transfer described by the descriptor. The CDMA CURDESC_PNTR register is updated by the SG function to reflect the actual descriptor being processed by the engine. This register can be read by the software application but it should be noted that the value will be changing while CDMASR.Idle = 0.

  ***Note:*** On error, the CDMA engine will halt and the CURDESC_PNTR register will be updated to the address of the transfer descriptor associated with the error.

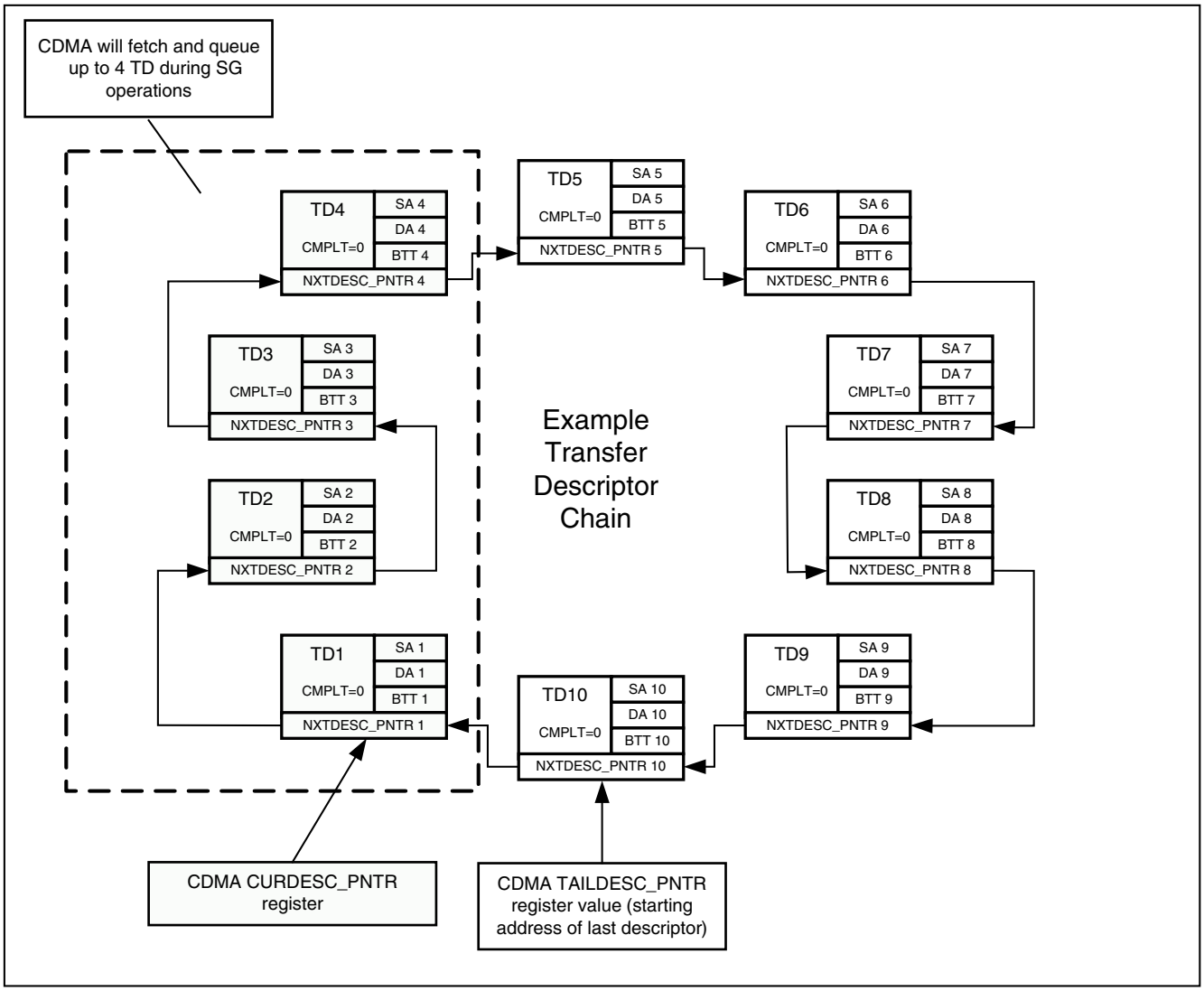


*Figure 17:* **Transfer Descriptor Execution Startup**

### Transfer Descriptor Update

- When the DMA transfer associated with a descriptor is finished by the CDMA DataMover function, the status of the transfer is scored by the CDMA SG function and then written into the STATUS word of the associated descriptor in system memory. The updated STATUS word will also have the Cmplt bit set to 1 indicating the CDMA has completed processing of the descriptor. This is shown in Figure 18.
- When the update of a descriptor is completed, the descriptor is popped from the internal CDMA queue. This opens up room for a new descriptor to be fetched and added to the internal queue. As long as there is room for a descriptor to be stored in the fetch queue, descriptors will continue to be fetched by the CDMA Scatter Gather function. Descriptor fetches and updates occur in sequential order defined by the descriptor linkage pointers (NXTDESC_PNTR).
- A descriptor that is marked as completed by the CDMA is available for the software application to reuse for future transfer automation. The CDMA will not proceed past the descriptor at the address indicated by the CDMA TAILDESC_PNTR register.

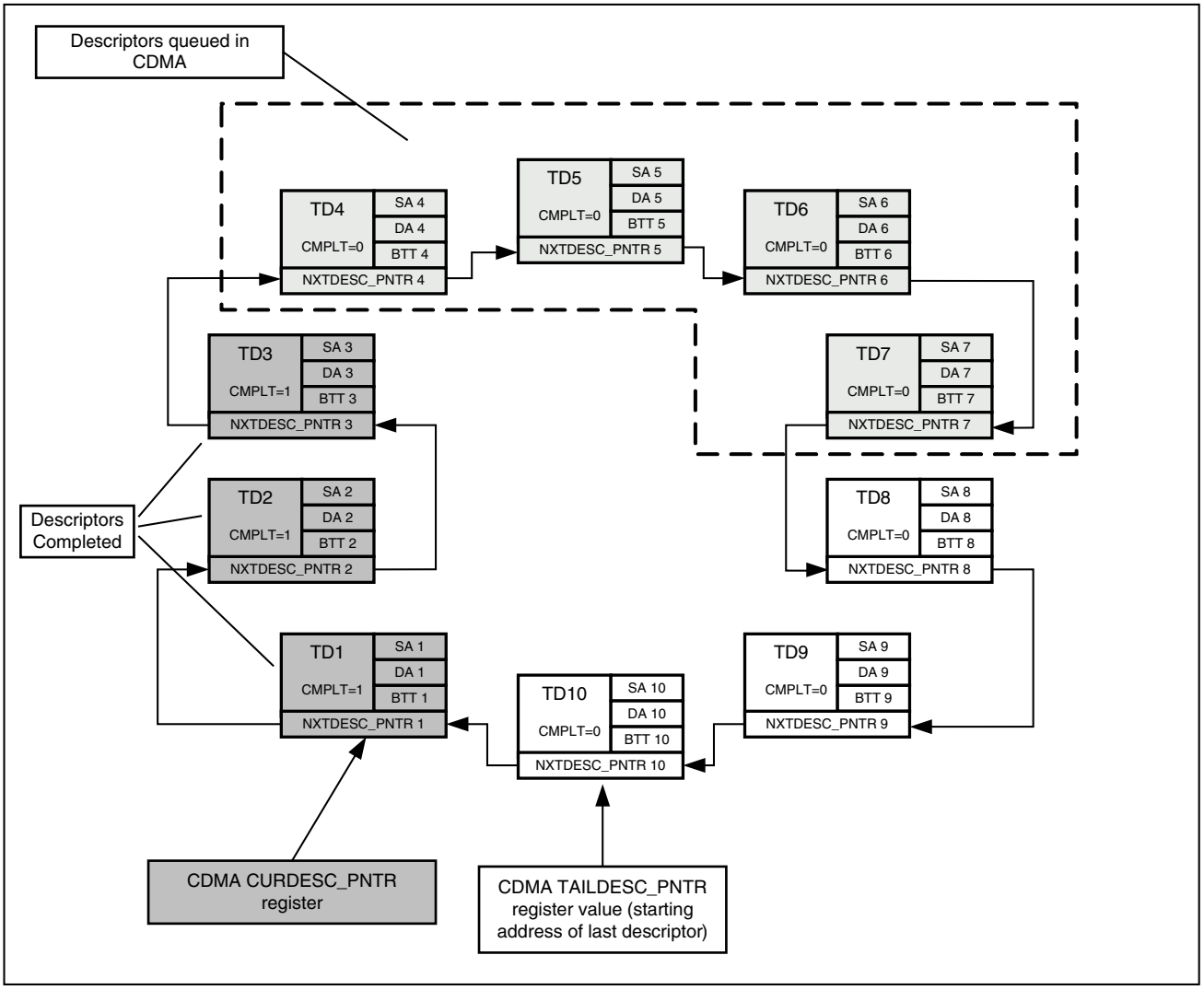


*Figure 18:* **Continued Execution**

### Transfer Descriptor Reuse

- Once the software application has recognized the completed descriptor and performed any necessary processing associated with it, the application may re-allocate the descriptors by adjusting the SA, DA, BTT, NXTDESC_PNTR, and zeroing out the STATUS word. This is depicted in Figure 19.
- After re-allocating the descriptors, the software application has the option of moving the TAILDESC_PNTR to point to the address of the last re-allocated descriptor or keeping it at the original end of chain value. This is up to the application requirements.
- If AXI CDMA is paused (CDMASR.Idle=1) because it has encountered the transfer descriptor with the starting address matching the CDMA TAILDESC_PNTR register value, it will automatically re-start descriptor fetching when the software application writes the new CDMA TAILDESC_PNTR register value. The next descriptor fetched will be the one pointed to by the NXTDESC_PNTR value of the descriptor when the pause occurred.

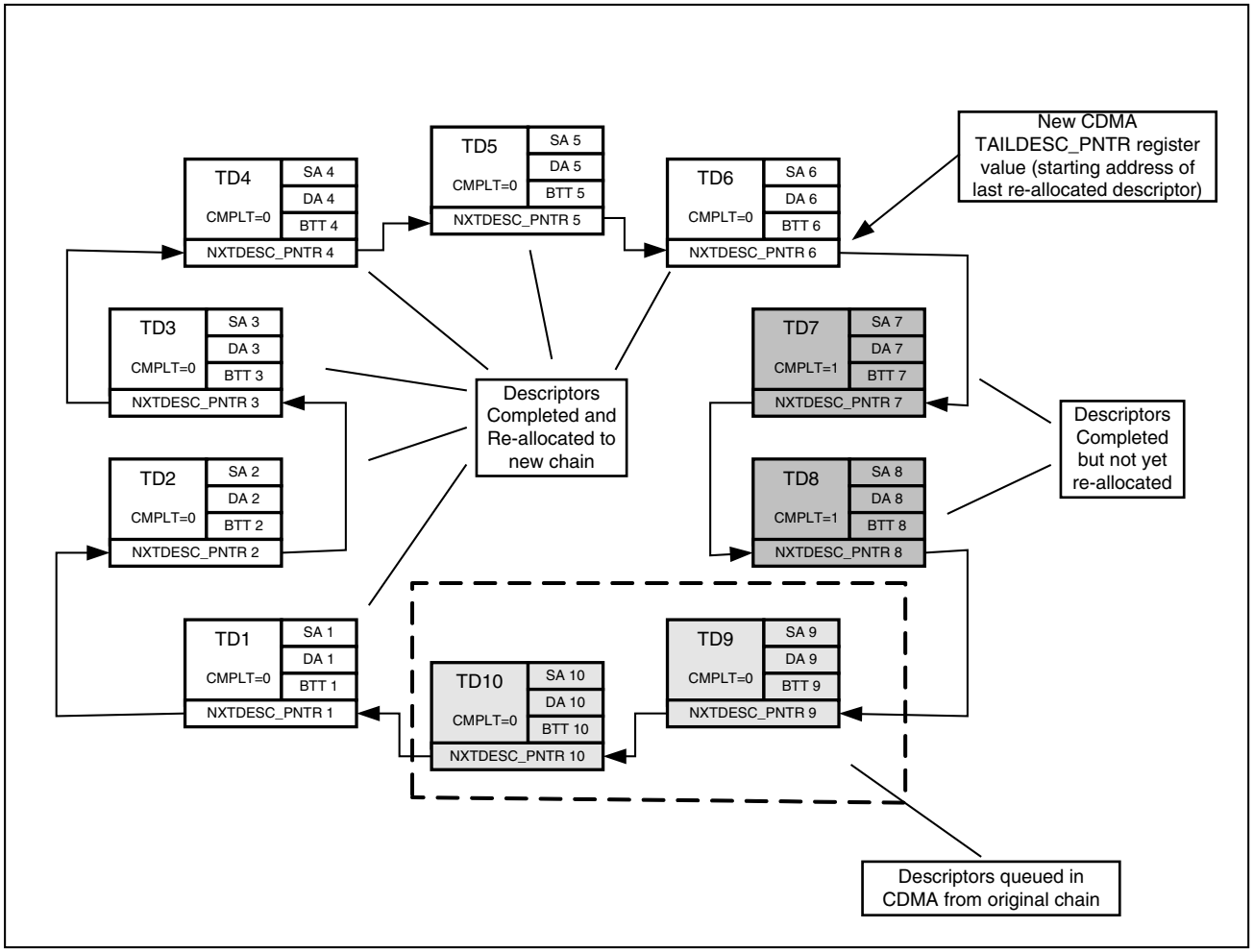


*Figure 19:* **Transfer Descriptor Re-Allocation**

### CDMA Scatter Gather Restart and Pause Behavior

The CDMA Scatter Gather engine has two different ways in which to continue SG operation after it has paused at a tail pointer descriptor. The first method is to re-allocate transfer descriptors in the chain that is currently active. The software application merely writes the address of the last re-allocated transfer descriptor to the CDMA TAILDESC_PNTR register. This will cause the CDMA SG engine to continue sequencing through the re-allocated descriptor chain until the next tail pointer location is hit. However, it is sometimes advantageous for a software application to have a different descriptor chain set up somewhere else in memory that needs to be executed after the CDMA has hit the initial tail descriptor. To do this, the CDMA SG must be restarted after the CDMA SG has paused and the CDMASR.Idle bit is set. At that point, the application must set the CDMACR.SGMode bit to '0'. Then the process described in the section SG Initialization Sequence on page 32 should be followed again. Note that the starting descriptor address of the new chain must be written to the CDMA CURDESC_PNTR register during the initialization process.

## Interrupt Generation

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). 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 use of Scatter Gather automation greatly improves DMA operational throughput by removing the system CPU from the real time control of the AXI CDMA. However, this improved transfer throughput can overwhelm the CPU with transfer completion interrupts. This is solved by providing a programmable filter method called 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 will be generated on the AXI CDMA `cdma_introut` signal. When a threshold interrupt is generated, the programmed threshold count is reloaded into the SG Engine threshold counter in preparation for the next threshold event. The real time internal threshold count value in the SG engine is provided to the software application in the CDMASR.IRQThresholdSts field of the CDMA status register. A CDMACR.IRQThreshold value of 0x01 (default value) causes the SG Engine to generate an interrupt for every transfer descriptor executed (disabling the coalescing feature).

If the delay interrupt feature is enabled (CDMACR.IRQDelay not equal to 0), a delay interrupt event will cause 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 will be 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 will receive 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 will begin 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'). A delay interrupt event occurs when the AXI CDMA is idle long enough for the internal delay counter in the SG Engine to count up to and equal the programmed delay count value. If a delay interrupt event occurs, the SG Engine delay timer is reset to zero (CDMASR.IRQDelaySts=0x00), and the delay timer will not count until the CPU services the delay interrupt by clearing the CDMASR.Dly_Irq bit to 0.

### Error Interrupts

Any detected error will result 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 will complete 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 will be 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 will be logged. In order to resume operations, a reset must be issued to the AXI CDMA either via a hardware reset (axi_resetn = '0') 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 would happen 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.

## Clocking

AXI CDMA utilizes a single clock for logic synchronization. This clock is input on the axi_aclk input port. All interfaces for the core are required to be synchronized to this clock. The AXI CDMA has been simulation tested with an `axi_aclk` frequency range of 50 MHz to 200 MHz. Actual Fmax achieved in a hardware implementation may vary See Performance, page 39.

## Resets

An active low reset assertion on the CDMA `axi_resetn` input will reset the entire AXI CDMA core. 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 `axi_resetn` input is synchronous to the `axi_aclk` master clock input and is asserted for the minimum number of clocks stated in Table 19. The table also indicates the stabilization time for AXI CDMA outputs reacting to a reset condition.

*Table 19:* **Reset Assertion/Deassertion Stabilization Times**

| Description | Value | Applicable Signal |
|---|---|---|
| Minimum assertion time | 8 clocks (axi_aclk) | axi_resetn input |
| Reset assertion to output signals in reset state (maximum) | 3 clocks (axi_aclk) | All output signals |
| Reset deassertion to normal operation state (maximum) | 3 clocks (axi_aclk) | All output signals |

## Performance

The targeted design clock frequencies of AXI CDMA for the Spartan-6 and Virtex-6 FPGA families are shown below. The maximum achievable clock frequency and resource utilization estimates may be different from those shown due to tool options, FPGA speed and logic utilization, and other factors.

- Spartan-6 FPGA: 120 MHz
- Virtex-6 FPGA: 180 MHz

## 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. 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 assigned to the C_M_AXI_BURST_LEN parameter of the AXI CDMA. In general, the bigger value assigned increases the realized throughput of the AXI CDMA. However, larger burst lengths may be detrimental to other components of a user's system design (causing lower system performance) so this is a trade-off that the user must evaluate

*Table 20:* **AXI CDMA Throughput**

| C_M_AXI_BURST_LEN | Test Packet Size | AXI4 Freq. (axi_aclk input in Virtex-6 FPGA) | Observed bus bandwidth utilization by AXI CDMA |
|---|---|---|---|
| 16 | 9000 bytes | 150Mhz | 70% |
| 64 | 9000 bytes | 150Mhz | up to 99% |

## Resource Utilization

Resource utilization numbers for the AXI CDMA core are shown for the Spartan-6 FPGA family in Table 21 and for the Virtex-6 FPGA family in Table 22. These values have been generated using the Xilinx EDK and ISE® tools for version 12.3. They are derived from ISE software Map reports from actual hardware validation systems. Parameter combinations shown are not all that are possible but are chosen to represent a range of minimum utilization (low performance) to high utilization (high performance).

*Table 21:* **Spartan-6 FPGA Resource Estimates**

| C_M_AXI_DATA_WIDTH | C_MAX_BURST_LEN | C_INCLUDE_DRE | C_INCLUDE_SG | C_USE_DATAMOVER_LITE | Slices | Slice Reg | Slice LUTs | Block RAM |
|---|---|---|---|---|---|---|---|---|
| 32 | 16 | 0 | 0 | 1 | 381 | 841 | 434 | 0 |
| 32 | 16 | 0 | 0 | 0 | 603 | 1240 | 923 | 0 |
| 32 | 16 | 1 | 0 | 0 | 712 | 1369 | 1164 | 0 |
| 64 | 256 | 1 | 0 | 0 | 926 | 1787 | 1472 | 0 |
| 256 | 256 | 0 | 0 | 0 | 1047 | 3157 | 1777 | 0 |
| 32 | 16 | 0 | 1 | 0 | 1361 | 2896 | 2185 | 1 |
| 64 | 128 | 1 | 1 | 0 | 1651 | 3436 | 2692 | 1 |
| 256 | 256 | 0 | 1 | 0 | 1767 | 4805 | 2927 | 1 |

*Table 22:* **Virtex-6 FPGA Resource Estimates**

| C_M_AXI_DATA_WIDTH | C_MAX_BURST_LEN | C_INCLUDE_DRE | C_INCLUDE_SG | C_USE_DATAMOVER_LITE | Slices | Slice Reg | Slice LUTs | Block RAM |
|---|---|---|---|---|---|---|---|---|
| 32 | 16 | 0 | 0 | 1 | 384 | 871 | 439 | 0 |
| 32 | 16 | 0 | 0 | 0 | 635 | 1279 | 982 | 0 |
| 32 | 16 | 1 | 0 | 0 | 706 | 1398 | 1088 | 0 |
| 64 | 256 | 1 | 0 | 0 | 925 | 1823 | 1461 | 0 |
| 256 | 256 | 0 | 0 | 0 | 1075 | 3196 | 1876 | 0 |
| 32 | 16 | 0 | 1 | 0 | 1415 | 2958 | 2190 | 1 |
| 64 | 128 | 1 | 1 | 0 | 1755 | 3509 | 2745 | 1 |
| 256 | 256 | 0 | 1 | 0 | 1847 | 4882 | 3111 | 1 |

## Support

Xilinx provides technical support for this LogiCORE™ IP product when used as described in the product documentation. Xilinx cannot guarantee timing, functionality, or support of product if implemented in devices that are not defined in the documentation, if customized beyond that allowed in the product documentation, or if changes are made to any section of the design labeled *DO NOT MODIFY*.

## Ordering Information

This Xilinx LogiCORE IP module is provided at no additional cost with the Xilinx ISE Design Suite Embedded Edition software under the terms of the Xilinx End User License. The core is generated using the Xilinx ISE Embedded Edition software (EDK).

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

## Reference Documents

The following document contains reference information important to understanding the design:

- AXI4 AMBA AXI Protocol Version: 2.0 Specification
- DS768, *AXI Interconnect IP Data Sheet*

## List of Acronyms

| Acronym | Spelled Out |
|---|---|
| AXI | Advanced eXtensible Interface |
| BTT | Bytes To Transfer |
| CDMA | Central Direct Memory Access |
| CPU | Central Processing Unit |
| DA | Destination Address |
| DMA | Direct Memory Access |
| DRE | Data Realignment Engine |
| DSP | Digital Signal Processing |
| EDK | Embedded Development Kit |
| FF | Flip-Flop |
| FPGA | Field Programmable Gate Array |
| I/O | Input/Output |
| IP | Intellectual Property |
| IOC | Interrupt On Complete |
| LUT | Lookup Table |
| MHz | Mega Hertz |
| MM2S | Memory Map to Stream |
| R/W | Read/Write |
| RAM | Random Access Memory |
| RO | Read Only |
| S2MM | Stream to Memory Map |
| SA | Source Address |
| SG | Scatter/Gather |
| SW | Software |
| TD | Transfer Descriptor |
| VHDL | VHSIC Hardware Description Language (VHSIC an acronym for Very High-Speed Integrated Circuits) |
| XPS | Xilinx Platform Studio (part of the EDK software) |
| XST | Xilinx Synthesis Technology |

## Revision History

The following table shows the revision history for this document:

| Date | Version | Description of Revisions |
|---|---|---|
| 09/21/10 | 1.0 | Initial Xilinx Release |

## Notice of Disclaimer