NORTHWEST LOGIC

AXI DMA Back-End Core

User Guide

To The Point Solutions

Version 1.05

1 Revision History

This section tracks revisions made to this document by version number

Revision	Date	Changes
1.00	09/29/2010	• Combined information from several sources into this new unified document
1.01	09/30/2010	• Minor corrections and formatting
1.02	12/17/2010	• Revised AXI Master Interface description; DMA Direct Control moved to Master Interface
1.03	02/28/2011	• Revised AXI Target Interface description
1.04	06/13/2011	• Revised C2S DMA, S2C DMA, Target, and Master Interfaces
1.05	06/14/2011	• Misc clarifications

2 AXI DMA Back End Core Overview

The AXI DMA Back-End Core is an alternate configuration of Northwest Logic's DMA Back End Core with AXI interfaces replacing the application-specific interfaces of the DMA Back End Core. For maximum application support, the AXI DMA Back-End supports multiple AXI protocol options on its AXI DMA interfaces. The AXI DMA Back-End Core implements the following AXI interfaces in place of the equivalent DMA Back End Core interfaces:

- AXI Target Interface (AXI3/4 Master) replaces:
  - Register Interface
  - Target Interface
- AXI Master Interface (AXI4-Lite) replaces:
  - Master Interface
  - DMA Direct Control Interface
- AXI DMA System to Card Interface (AXI3/4 Master or AXI4-Stream Master) replaces:
  - DMA System to Card Interface
- AXI DMA Card to System Interface (AXI3/4 Master or AXI4-Stream Slave) replaces:
  - DMA Card to System Interface

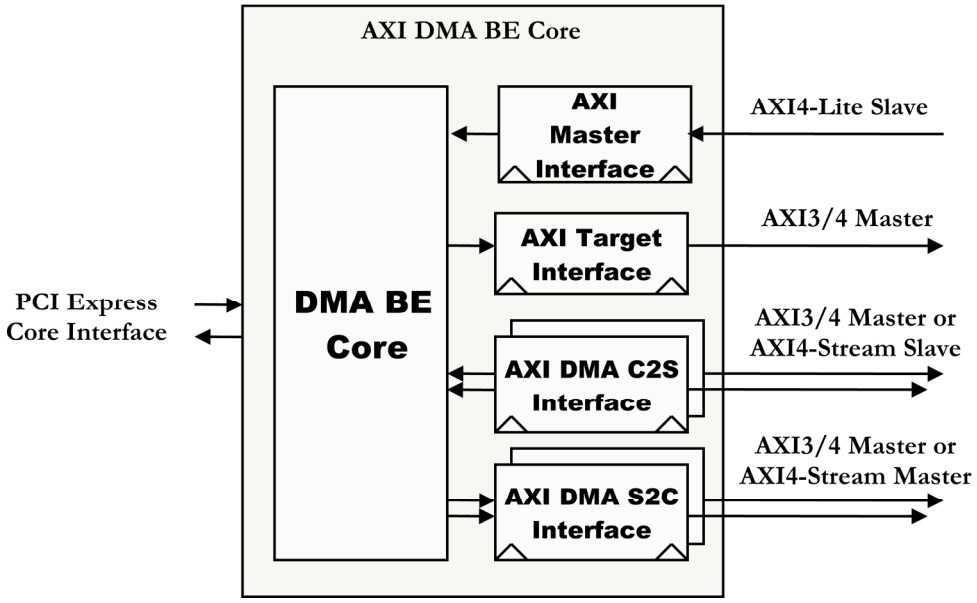

Figure 2-1 AXI DMA Back-End Core Block Diagram

2.1 AXI DMA Back-End Core Features

The AXI DMA Back-End Core is an alternate configuration of the DMA Back End Core with AXI interfaces replacing the standard application specific interfaces of the DMA Back End. Other than the interface protocol differences, the AXI DMA Back-End Core features are the same as the DMA Back-End Core. Please see the DMA Back-End Core User Guide for details.

Key feature differences between AXI DMA Back-End Core and DMA Back-End Core:

- The AXI DMA Back End supports the following PCIe and AXI data width combinations:
  - 128-bit AXI (128-bit PCIe Data Width)
  - 64-bit AXI (128-bit or 64-bit PCIe Data Width)
  - 32-bit AXI (128-bit, 64-bit, or 32-bit PCIe Data Width)
- DMA Interface
  - AXI DMA Card to System Interface replaces DMA Card to System Interface
    - AXI3/AXI4 Master (Addressable DMA) or AXI4-Stream Slave (FIFO DMA)
    - DMA data transfer
  - AXI DMA System to Card Interface replaces DMA System to Card Interface
    - AXI3/AXI4 Master (Addressable DMA) or AXI4-Stream Master (FIFO DMA)
    - DMA data transfer
  - AXI DMA Back-End Core implements Packet DMA
    - DMA Back End Core legacy Block DMA support is not available
- Target Interface
  - AXI Target Interface replaces Target Interface and Register Interface
    - AXI3/AXI4-Master
    - Implements write-read ordering to maintain coherency for PCI Express transactions
    - Supports full bandwidth utilization when being driven by remote PCI Express DMA Master
    - Used to respond to remote PCI Express-mastered read and write transactions
- Register Interface
  - AXI Target Interface replaces Target Interface and Register Interface
  - See Target Interface description above for additional detail
- Master Interface
  - AXI Master Interface replaces Master Interface and DMA Direct Control Interface
    - AXI4-Lite Slave (32-bit, non-burst interface intended for control)
    - Implements AXI-mapped registers to provide AXI Masters the ability to make PCI Express requests (Memory Writes/Reads, Messages, etc.)
    - Enables AXI Masters to write and read internal DMA Registers; DMA are typically initiated from a remote Host CPU over PCI Express, but can be initiated from a local AXI CPU using this interface. Note that Descriptor Table and DMA Completion status remain in Host memory.
- Complete Reference Design Provided
  - Illustrates operation of all interfaces
  - Enables hardware DMA throughput measurements
  - See Section 7 for details
- Source code available
- Customization and Integration services available

3 AMBA® Specifications

The AXI DMA Back-End Core is compatible with the following AMBA® 4 specifications:

- AMBA® AXI Protocol Version: 1.0 Specification
  - AXI3
- AMBA® AXI Protocol Version: 2.0 Specification
  - AXI4
  - AXI4-Lite
- AMBA® 4 AXI4-Stream Protocol Version: 1.0
  - AXI4-Stream

3.1 AMBA® 3 (AXI3) and AMBA® 4 (AXI4) Compatibility

The AXI DMA Back-End Core AXI3/AXI4 Master Interfaces are compatible with both the AXI3 and AXI4 Specifications

- DMA Back-End utilizes the subset of functionality present in AXI3 to enable support for AXI3 and AXI4
- Master burst sizes are limited to a maximum of 16 beats for compatibility with AXI3 and AXI4

3.2 AMBA® Specification Availability

AMBA® 4 and AMBA® 3 Specifications are available from ARM Ltd.

- http://www.arm.com/products/system-ip/amba/amba-open-specifications.php

4 AXI Target Interface

4.1 AXI Target Interface Port Descriptions

The AXI Target Interface implements AXI3/AXI4 Master Protocol. Write and read requests received via PCI Express (from remote PCI Express masters) that target AXI regions of enabled Base Address or Expansion ROM regions are forwarded to the AXI Target Interface for completion. Accesses to registers inside the AXI DMA Back-End Core are handled by the core and do not appear on the AXI Target Interface.

The AXI Target Interface implements write-read ordering to maintain coherency for PCI Express transactions

- Ordering is maintained separately for internal DMA Register and AXI destinations
- The completion of a read request to the same destination (DMA Registers or AXI) can be used to guarantee that prior writes to the same destination have completed
- Reads are blocked until all writes occurring before the read have fully completed; for AXI, a write is completed when it returns a completion response on the Write Response Channel; for internal DMA Registers, a write is completed when it is written into the DMA Registers such that a following read will return the new value
- Supports full duplex bandwidth utilization when being driven by a remote PCI Express DMA Master
- Supports multiple simultaneously outstanding write and read requests
    - Utilizes maximum 16 beat bursts for compatibility with AXI3 and AXI4

The AXI Target Interface implements FIFOs to buffer multiple writes and reads simultaneously to enable maximum bandwidth.

The AXI Target Interface implements a dual clock interface. The AXI clock domain may be different than the PCI Express clock domain. Gray Code synchronization techniques are used to enable support for a wide variety of AXI clock rates.

It is important to consume target write and read transactions relatively quickly as it is possible to stall PCI Express completions (used for S2C DMA for example) if target write and read transactions are allowed to languish in the PCI Express Core Receive Buffer.

The AXI Target Interface ports are listed in Table 4-1.

Table 4-1 AXI Target Interface Ports

PORT	TYPE	DESCRIPTION
t_areset_n	Input	Active-low asynchronous assert, t_aclk-synchronous de-assert reset; Must be asserted when DMA Back End PCI Express reset is asserted.
t_aclk	Input	AXI interface clock; may be a different clock than the clock used on the PCI Express-side of the AXI DMA Back-End Core; synchronization techniques are used to enable support for a wide variety of clock rates
t_awregion[2:0]	Output	Write Address Channel; Optional AWBURST, AWLOCK, AWCACHE, AWPROT are not implemented; AWBURST is always incrementing-address burst; cache, protected, and exclusive accesses not supported; see below for t_awregion information
t_awaddr[31:0]	Output	
t_awlen[3:0]	Output	
t_awsize[2:0]	Output	
t_awvalid	Output	
t_awready	Input	
t_wdata[D-1:0]	Output	Write Data Channel
t_wstrb[B-1:0]	Output	
t_wlast	Output	
t_wvalid	Output	
t_wready	Input	
t_bresp[1:0]	Input	Write Response Channel; space is reserved in the master to receive response from all outstanding write requests, so t_bready is always 1 and does not need to be used
t_bvalid	Input	
t_bready	Output	

Table 4-1 AXI Target Interface Ports (Continued)

PORT	TYPE	DESCRIPTION
t_arregion[2:0]	Output	Read Address Channel; Optional ARBURST, ARLOCK, ARCACHE, ARPROT are not implemented; ARBURST is always incrementing-address burst; cache, protected, and exclusive accesses not supported; see below for t_arregion information
t_araddr[31:0]	Output	
t_arlen[3:0]	Output	
t_arsize[2:0]	Output	
t_arvalid	Output	
t_arready	Input	
t_rdata[D-1:0]	Input	Read Data Channel; space is reserved in the master to receive data from all outstanding read requests, so t_rready is always 1
t_rresp[1:0]	Input	
t_rlast	Input	
t_rvalid	Input	
t_rready	Output	

- D = Core Data Width and is the same Core Data Width as the AXI DMA Back-End Core
- B = Core Data Width in Bytes == D/8

targ_awregion and targ_arregion indicate PCI Express Base Address Region hit information:

- 0 – BAR0
- 1 – BAR1
- 2 – BAR2
- 3 – BAR3
- 4 – BAR4
- 5 – Bar5
- 6 - Expansion ROM
- 7 - Reserved

Region information should be used to translate the PCI Express Base Address Regions into their corresponding AXI address region. This is accomplished by adding a pre-defined, fixed AXI address offset to targ_awaddr/targ_araddr based upon the value of targ_awregion/targ_arregion. This effectively translates the PCI Express address into an AXI address and allows the Base Address Regions to be mapped differently in PCIe and AXI address maps. The offsets added could be hard tied based upon a known AXI address map or implemented in AXI addressable registers to allow the offsets to be programmed by the AXI-side CPU.

5 AXI DMA Interface

The AXI DMA Interface is the mechanism through which user logic interacts with the DMA Engine. The AXI DMA Interface orchestrates the flow of DMA data between user logic and PCI Express.

The AXI DMA System to Card and AXI DMA Card to System interfaces support multiple AXI protocol options (which are selected with the DMA Back End DMA Engine inputs c2s/s2c_fifo_addr_n):

- AXI3/AXI4
- AXI4-Stream

5.1 Addressable FIFO DMA

When a DMA Engine is configured to implement an AXI3/AXI4 interface, system software can set the "Addressable FIFO DMA" Descriptor bit in all Descriptors of an application DMA transfer to instruct the DMA Back End to provide the same starting AXI address provided by software for all AXI transactions for this packet. This allows the user hardware design to implement FIFOs for some AXI DMA transactions while simultaneously also supporting addressable RAM for other AXI DMA transactions. Please see the Descriptor definition in the DMA Back-End Core User Guide 4.17 (or later) for details. Please pay close attention to the AXI address alignment requirements for selecting addresses to use for AXI FIFO DMA targets.

5.2 AXI DMA System to Card Interface

Each System to Card DMA Engine has one set of the following ports.

Table 5-1 AXI DMA System to Card Interface

PORT	TYPE	DESCRIPTION
s2c_areset_n	Output	Active-low asynchronous assert, s2c_aclk synchronous de-assert reset; asserted when the DMA Engine has been reset by software or by PCI Express reset
s2c_aclk	Input	AXI interface clock; may be a different clock than the clock used on the PCI Express-side of the AXI DMA Back-End Core; synchronization techniques are used to enable support for a wide variety of clock rates
s2c_fifo_addr_n	Input	Interface AXI Protocol Selection: • 1 – FIFO DMA using AXI4-Stream Protocol • 0 – Addressable DMA using AXI3/AXI4 Protocol This port selects the interface protocol and affects the operation of the remaining ports
s2c_awaddr[35:0]	Output	FIFO DMA: Write Address Channel is unused; tie s2c_awready == 1 and ignore s2c_aw* outputs Addressable DMA: Write Address Channel; Optional AWBURST, AWLOCK, AWCACHE, AWPROT are not implemented; AWBURST is always incrementing-address burst; cache, protected, and exclusive accesses not supported; s2c_awusereop is a non-standard AXI signal that when 1 indicates that this is the final write request of a DMA packet transfer
s2c_awlen[3:0]	Output	
s2c_awsize[2:0]	Output	
s2c_awvalid	Output	
s2c_awready	Input	
s2c_awusereop	Output	
s2c_wdata[D-1:0]	Output	FIFO DMA: Write Data Channel implements AXI4-Stream Master protocol using s2c_wdata(tdata), s2c_wstrb(tkeep), s2c_wlast(tlast), s2c_wvalid(tvalid), and s2c_wready(tready); NULL (TKEEP == 0) bytes are only placed at the end of a stream (packet); position bytes not implemented; optional TSTRB, TID, and TDEST not implemented; interleaving of streams is not performed; a new stream will start only after the prior stream finishes; s2c_wusercontrol is a non-standard AXI signal, valid for the entire packet transfer (typically multiple AXI transfers), that provides the UserControl[63:0] value software placed in the first Descriptor of the packet. Optional signal which may be used to pass information on a per packet basis from user software to user hardware; s2c_wusercontrol is only valid for FIFO DMA Addressable DMA: Write Data Channel implements AXI3/AXI4 Master protocol; s2c_wusereop is a non-standard AXI signal, with same timing as s2c_wlast, that when 1 indicates that this is the final data transfer of a DMA packet transfer
s2c_wstrb[B-1:0]	Output	
s2c_wlast	Output	
s2c_wusereop	Output	
s2c_wusercontrol[63:0]	Output	
s2c_wvalid	Output	
s2c_wready	Input	
s2c_bresp[1:0]	Input	FIFO DMA: Write Response Channel is unused; tie s2c_bready == 1 and ignore s2c_b* outputs Addressable DMA: Write Response Channel; space is reserved in the master to receive response from all outstanding write requests, so t_bready is always 1 and does not need to be used
s2c_bvalid	Input	
s2c_bready	Output	

5.3 AXI DMA Card to System Interface

Unlike System to Card DMA transfers where the DMA data requests and DMA data travel in the same direction, Card to System DMA has requests coming to the card from PCI Express while data goes the opposite direction from the card to PCI Express.

Each Card to System DMA Engine has one set of the following ports:

PORT	TYPE	DESCRIPTION
c2s_areset_n	Output	Active-low asynchronous assert, c2s_aclk synchronous de-assert reset; asserted when the DMA Engine has been reset by software or by PCI Express reset
c2s _aclk	Input	AXI interface clock; may be a different clock than the clock used on the PCI Express-side of the AXI DMA Back-End Core; synchronization techniques are used to enable support for a wide variety of clock rates
c2s _fifo_addr_n	Input	Interface AXI Protocol Selection: • 1 – FIFO DMA using AXI4-Stream Protocol • 0 – Addressable DMA using AXI3/AXI4 Protocol This port selects the interface protocol and affects the operation of the remaining ports
c2s _araddr[35:0]	Output	FIFO DMA: Read Address Channel is unused; tie c2s_arready == 1 and ignore c2s_ar* outputs Addressable DMA: Read Address Channel; Optional AWBURST, AWLOCK, AWCACHE, AWPROT are not implemented; AWBURST is always incrementing-address burst; cache, protected, and exclusive accesses not supported
c2s _arlen[3:0]	Output	
c2s _arsize[2:0]	Output	
c2s _arvalid	Output	
c2s _arready	Input	
c2s _rdata[D-1:0]	Input	FIFO DMA: Read Data Channel implements AXI4-Stream Slave protocol using c2s_rdata(tdata), c2s_ruserstrb(tkeep), c2s_rlast(tlast), c2s_rvalid(tvalid), and c2s_rready(tready). NULL (TKEEP[i] == 0) bytes only permitted at the end of a stream; position bytes not implemented; optional TSTRB, TID, and TDEST not implemented; interleaving of streams is not supported; a new stream may only start after the prior stream finishes; c2s_ruserstatus is a non-standard AXI signal, which must be valid when c2s_rlast (tlast) == 1 & c2s_rvalid (tvalid) == 1, that is used to update the UserStatus[63:0] value in the last Descriptor of the packet; optional signal which may be used to pass information on a per packet basis from user hardware to user software; c2s_ruserstatus is only valid for FIFO DMA; if unused, tie to 0 Addressable DMA: Read Data Channel implements AXI3/AXI4 Master protocol; non-standard AXI ports c2s_ruserstrb & c2s_ruserstatus are unused and must be tied to 0.
c2s _rresp[1:0]	Input	
c2s _rlast	Input	
c2s _ruserstrb[B-1:0]	Input	
c2s_ruserstatus[63:0]	Input	
c2s _rvalid	Input	
c2s _rready	Output	

Card to System DMA Engines are designed to be flexible. Software controls the data flow when using Addressable DMA (read requests are made via AXI3/4 is response to software DMA data requests). Hardware controls the data flow when using FIFO DMA (packets are written into an AXI4-Stream slave and forwarded to system software). For FIFO C2S DMA, hardware is the source of the packets, but the DMA Engine must be enabled and a buffer setup to receive packets from hardware before hardware will be able to transfer packets to system software.

6 AXI Master Interface

The AXI Master Interface is an AXI4-Lite Slave interface that enables the user to:

- Generate PCI Express requests with up to 1 DWORD (32-bit) payload
- Write and read DMA Back-End internal registers to start DMA operation and obtain interrupt status

The AXI Master Interface implements a register set to enable the above functions.

- A PCI Express request is carried out by writing the PCI Express-specific information (PCI Express Address, Format and Type, etc.) to the register set and then writing to another register to execute the request.
- DMA Registers are made accessible via AXI reads and writes

6.1 Master Interface Port Descriptions

The Master Interface ports are described in Table 6-1.

Table 6-1 Master Interface

PORT	TYPE	DESCRIPTION
m_aclk	Input	AXI interface clock; may be a different clock than the clock used on the PCI Express-side of the AXI DMA Back-End Core; synchronization techniques are used to enable support for a wide variety of clock rates
m_areset_n	Input	Active-low asynchronous assert, m_aclk-synchronous de-assert reset

Table 6-1 Master Interface (Continued)

PORT	TYPE	DESCRIPTION
Write Address Channel		
m_awvalid	Input	A Write Address Channel transfer occurs when m_awvalid == 1 and m_awready == 1
m_awready	Output	
m_awaddr[15:0]	Input	Byte address of register to write
Write Data Channel		
m_wvalid	Input	A Write Data Channel transfer occurs when m_wvalid == 1 and m_wready == 1
m_wready	Output	
m_wdata[31:0]	Input	Data to write
m_wstrb[3:0]	Input	Byte enables for write
Write Response Channel		
m_bvalid	Output	A Write Response Channel transfer occurs when m_bvalid == 1 and m_bready == 1
m_bready	Input	
m_bresp[1:0]	Output	Status of write request: 0 – Successful; 1, 2,3 Error
Read Address Channel		
m_arvalid	Input	A Read Address Channel transfer occurs when m_arvalid == 1 and m_arready == 1
m_arready	Output	
m_araddr[15:0]	Input	Byte address of register to read
Read Response Channel		
m_rvalid	Output	A Read Response Channel transfer occurs when m_rvalid == 1 and m_rready == 1
m_rready	Input	
m_rdata[31:0]	Output	Data read
m_rresp[1:0]	Output	Status of read request: 0 – Successful; 1, 2,3 Error

6.2 Master Interface Register Map

The Master Interface implements a register set for an AXI Master to perform PCI Express and DMA transactions and provides access to read and write the same DMA Back End registers used by PCIe software to start DMA and obtain status.

Table 6-2 Master Interface Register Map

GROUP BASE ADDRESS	DESCRIPTION
0x7FFF-0000	DMA Back-End Registers; writes to this region target the internal DMA Back-End Registers for starting DMA operations, obtaining interrupt status, etc. Please see Section 6.2.1 below and DMA Back End User Guide for a Description of DMA Back-End Registers.
0xFFFF-0x8000	PCI Express Master Interface Registers See Section 6.2.2 below.

6.2.1 Master Interface access of DMA Back-End Registers

An AXI Master may write and read DMA Back-End internal registers by issuing write and read requests on the Master Interface. All registers implemented inside the DMA Back-End are accessible including DMA Registers and Common Registers.

An AXI Master may start a DMA operation by writing to the appropriate DMA Back-End DMA registers. DMA Descriptors must still be located in PCIe system memory, so before a DMA operation can be started, the AXI CPU must communicate with the Host CPU to create the appropriate Descriptor Tables in PCIe system memory for the DMA operation.

The DMA Back-End Common Registers contain interrupt routing registers which may be used to route DMA interrupts either to PCIe (the default) or to AXI. Please see the DMA Back-End User Guide Common Registers for details. Generally when an AXI CPU is initiating the DMA operations, the associated interrupt vectors should be routed to the AXI CPU.

6.2.2 Master Interface access of PCI Express Devices

An AXI Master may master the PCI Express bus via the Master Interface to obtain data from remote PCI Express devices. Since AXI and PCIe address domains are normally separate and since there are additional transaction types available on PCIe that are not available on AXI, a register set is implemented in the AXI Master to allow the AXI Master to specify and execute the desired PCI Express transaction.

PCI Express splits transactions into Posted and Non-Posted Transactions

- Posted Transactions
  - Are assumed to complete normally
  - An additional write can be pended as soon as the prior write request has been forwarded to be transmitted on PCI Express
  - Memory Write
- Non-Posted Transactions
  - Do not complete until the associated data and/or completion status is returned from PCI Express; these accesses have high latency because they must traverse the PCI Express hierarchy to their destination, get executed, and then traverse the PCI Express hierarchy back to the source to return read data and/or completion status
  - Memory Read, IO Read, and Configuration Read return data and completion status
  - IO Write and Configuration Write return only completion status

To issue a command request to the PCIe bus, first program all the needed fields (PCI Express Address, PCI Express Write Data, Message Code, Command Type, and Byte Enables) and then write Command Start == 1. The write to Command Start will not be acknowledged on the AXI bus until the command has completed. At this point, the result and any returned data can be read from the registers.

AXI Master PCI Express write and read requests are required to follow all PCI Express rules. Violating PCI Express transaction rules will result in unspecified behavior. AXI Masters should typically only issue Memory Read and Memory Write requests. Message, Configuration, and IO transactions have specific uses in PCI Express that do not have equivalents in AXI. Configuration and IO transactions are not allowed to be generated from PCI Express Endpoints (most PCI Express applications).

Table 6-3 PCI Express Master Interface Registers

ADDRESS	DESCRIPTION
0x8003-0x8000	Master Command • [31] – Command Start - Set to 1 to execute a transaction on PCI Express using the contents of this and the other PCI Express Master Interface Registers • [30:24] - Reserved • [23:16] – Message Code; message code used for Message transactions • [15:8] – Command Type; type of transaction ○ 8'h00 – 32-bit Address Memory Read ○ 8'h20 - 64-bit Address Memory Read ○ 8'h40 - 32-bit Address Memory Write ○ 8'h60 - 64-bit Address Memory Write ○ 8'h04 – Type 0 Configuration Read (illegal for PCIe Endpoints) ○ 8'h44 - Type 0 Configuration Write (illegal for PCIe Endpoints) ○ 8'h05 - Type 1 Configuration Read (illegal for PCIe Endpoints) ○ 8'h45 - Type 1 Configuration Write (illegal for PCIe Endpoints) ○ {8'b0111_0, Routing[2:0]} – Message with Data Payload ○ {8'b0011_0, Routing[2:0]} – Message without Data Payload ○ 8'h02 – IO Read (illegal for PCIe Endpoints) ○ 8'h42 – IO Write (illegal for PCIe Endpoints) • [7:4] - Reserved • [3:0] – Byte Enables for write transactions
0x8007-0x8004	PCI Express Address[31:0]
0x800B-0x8008	PCI Express Address[63:32] – only used for 64-bit Memory Write, 64-bit Memory Read, and Message (message bytes 15:8]) transactions
0x800F-0x800C	PCI Express Write Data[31:0] – Data payload used for writes and Message with Data
0x8013-0x8010	Master Result • [31] – Command Done - Set to 1 when the prior command is completed; a new command cannot be pended until the prior command completes • [30:2] – Reserved • [1:0] – Command Status ○ 00 == Success ○ 01 == Reserved ○ 10 == Slave Error ○ 11 == Decode Error
0x8017-0x8014	PCI Express Read Data[31:0] – Data payload received for read requests; only valid if Command Status == 00 == Success
0xFFFF-0x8018	Reserved

7 AXI DMA Back-End Core Reference Design

A synthesizable reference design is provided to enable robust simulation and hardware verification of the AXI DMA Back-End independent of the user design. An additional goal is to provide enough flexibility over the packet generation to enable users to model their bandwidth requirements and test whether the targeted systems can sustain the desired throughputs. Different systems have better performance than others primarily due to larger PCI Express payload sizes and more efficient or higher-bandwidth system SDRAM controllers.

The reference design typically provided with AXI DMA Back-End Core licenses is illustrated in Figure 7-1.

The reference design consists of the following synthesizable components:

- Packet Generators to simulate providing DMA packet data for Card to System Packet DMA
- Packet Checkers to consume DMA packet data provided by System to Card Packet DMA
- Register Example to illustrate user register implementation
    - Re-creates the DMA Back-End Core Register Interface from the AXI Target Interface in order to illustrate generation of user registers and to simplify migration of user designs created with the DMA Back-End Core to the AXI DMA Back-End Core
- SRAM Example illustrating AXI Target Connection to SRAM
- Multi-ported AXI SRAM interface example to illustrate Addressable DMA RAM implementations

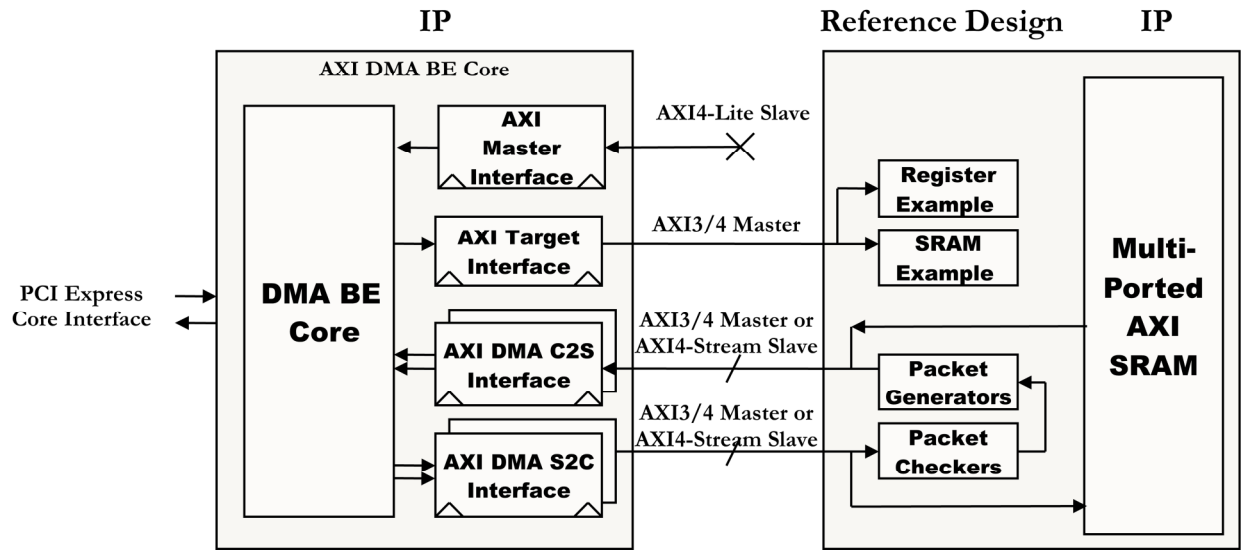

Figure 7-1 AXI DMA Back-End Core Reference Design

Please see the DMA Back-End Core User Guide for additional detail on the reference design components.

8 RAM Usage

This core contains some functions which require on-chip RAM arrays (also referred to as embedded memory or block RAM). The standard RTL code delivered by Northwest Logic implements these RAM arrays using generic Verilog or VHDL array constructs.

For FPGA targets, FPGA synthesis tools typically will automatically convert these arrays into the appropriate FPGA block RAMs or if necessary, Northwest Logic will pre-configure the IP Core with the proper RAM instantiations prior to delivery. In either case the designer does not need to be involved in this process. The FPGA designer only needs to ensure that there is enough block RAM available in their target part.

For ASIC targets, the designer needs to be actively involved in configuring the RAM arrays. ASIC flows generally do not support automatic generation of RAM arrays so the designer must first configure and create the RAM arrays using a memory generator tool and then substitute these in place of the inferred memory blocks provided in the IP core. To verify the actual size of the RAM, please follow the procedure outlined in the ASIC Integration Guide available from Northwest Logic. If there are any questions on the RAM configuration or handling of the substitution, please contact Northwest Logic.

An example report you will get when the simulation starts is as follows:

```
# tb_top.dut.pcie_complete_core_vc1.dma_back_end_pkt.s2c0_dma_engine_pkt.s2c_tx_pkt.cpl_reorder_
queue_pkt.caddr_ram: RAM Instance using ADDR_WIDTH=   4, DATA_WIDTH=  64, FAST_READ=   0
```

As you can see, this report shows the instance in the design hierarchy where the RAM is located, as well as the information on how the RAM is sized.

9 Speed & Size

For speed & size information, see the target device family document: ***Speed & Size Overview***

This document can be found on Northwest Logic's secure website. Request access to the secure website by sending an email to **nwl@nwlogic.com**.

10 Free Evaluation Core

To receive a free evaluation core, follow the following steps:

1. Request access to Northwest Logic's secure website by sending an e-mail to **nwl@nwlogic.com**
2. Log into the secure website section of Northwest Logic's website at **www.nwlogic.com**
3. Download the Evaluation Request Form for the core you are interested in
4. Fill in Table 3-2
5. E-mail the Evaluation Request Form to **nwl@nwlogic.com**
6. You should receive the Evaluation Core within 24 hours

11 For More Information

For more information including licensing options, pricing and the latest version of this document:

- Visit our website at **www.nwlogic.com**
- Send an e-mail to **nwl@nwlogic.com**
- Call us at **503-533-5800 x309**

Northwest Logic is located at:

Address: 1100 NW Compton Drive, Suite 100
Beaverton, Oregon 97006
United States

Phone: 503-533-5800

Fax: 503-533-5900