Xilinx Embedded Target RDMA Enabled NIC v1.1

LogiCORE IP Product Guide

Vivado Design Suite

PG294 June 6, 2018

XILINX

Table of Contents

Appendix A: Debugging

Appendix B: Additional Resources and Legal Notices

Introduction

The Xilinx® ETRNIC™ (Embedded Target RDMA enabled NIC) IP is a target only implementation of RDMA over Converged Ethernet (RoCE v2) enabled NIC functionality. This parameterizable soft IP core can work with a wide variety of Xilinx hard and soft MAC IP implementations providing a high through-put, low latency and completely hardware offloaded reliable data transfer solution over standard Ethernet. The ETRNIC IP allows simultaneous connections to multiple remote hosts running RoCE v2 traffic.

Features

- Support for Endpoint RDMA functionality
  - RoCE v2
  - Packet retransmission on errors in hardware
- 100 Gb/s data path
- Support for hardware based reliable connection
- Hardware handshake on user interface
- Supports incoming, and outgoing RDMA SEND
- Supports RDMA READ, and RDMA WRITE
- Does not support incoming RDMA READ, RDMA WRITE, and atomic operations
- Designed to scale up to 127 RDMA Queue pairs
- Support for IPv4 and IPv6 packets
- Support for Explicit Congestion Notification (ECN)

LogiCORE™ IP Facts Table	
Core Specifics	
Supported Device Family(1)	Kintex UltraScale+™, Virtex® UltraScale, Virtex UltraScale+, Zynq® UltraScale+
Supported User Interfaces	AXI4-Lite, AXI4-full, and AXI4-Streaming
Resources	Performance and Resource Utilization
Provided with Core	
Design Files	Encrypted RTL
Example Design	Verilog
Test Bench	Not provided
Constraints File	Xilinx Design Constraints (XDC)
Simulation Model	Not provided
Supported S/W Driver(2)	Indicate the supported software driver type: N/A/ Stand-alone/Stand-alone and Linux
Tested Design Flows(2)	
Design Entry	Vivado® Design Suite Vivado IP Integration
Simulation	For supported simulators, see the Xilinx Design Tools: Release Notes Guide.
Synthesis	Vivado Synthesis
Support	
Provided by Xilinx at the Xilinx Support web page	

Notes:

1. For a complete list of supported devices, see the Vivado IP catalog.
2. For the supported versions of the tools, see the Xilinx Design Tools: Release Notes Guide.
3. For UltraScale devices, only 40G data path is supported.

Overview

This chapter contains an overview of the core and details of applications, licensing, and standards. ETRNIC is a soft IP implementing RDMA over a Converged Ethernet (RoCE v2) protocol for Endpoint devices. This implementation is based on the specifications described in InfiniBandTM Architecture Specification Volume 1, Annex A16 RoCE and Annex 17 RoCE V2.

Figure 1-1 shows the ETRNIC and its connections to other IPs in the subsystem.

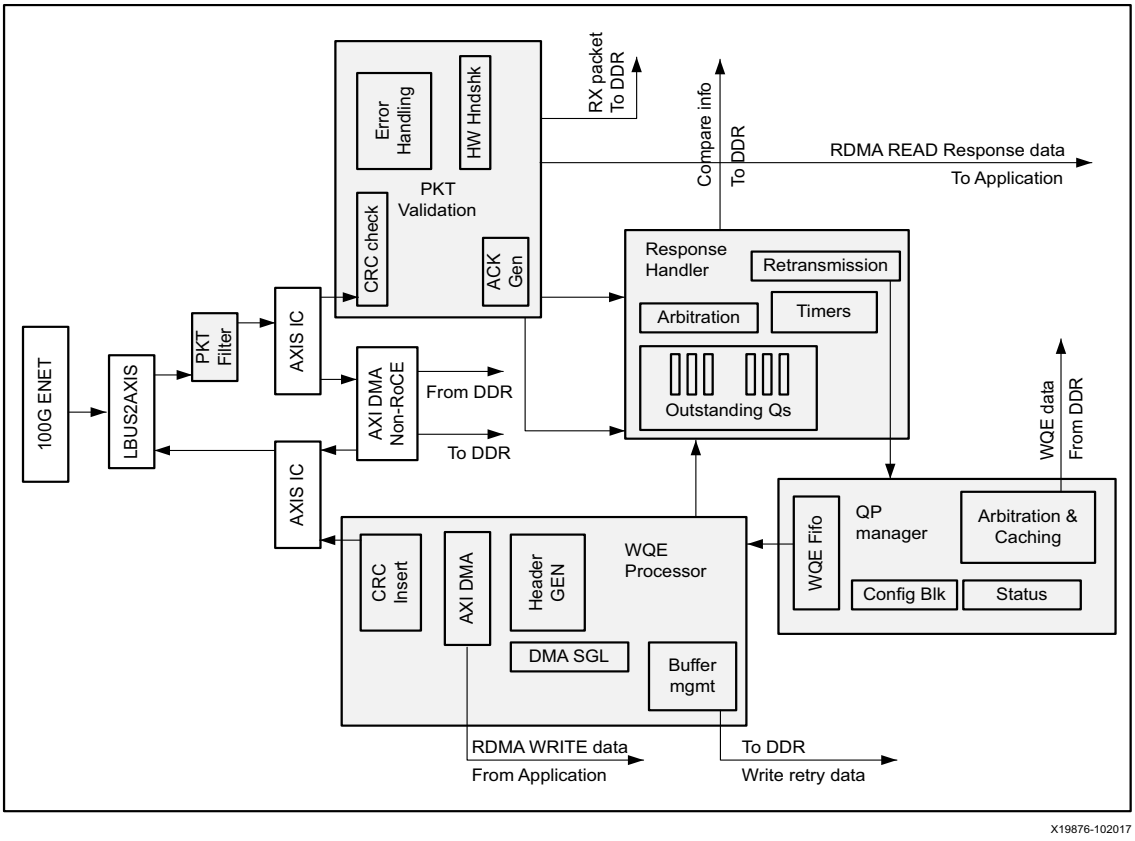

Figure 1-1: **ETRNIC IP and Subsystem**

Note: The user logic or target IP that connects to ETRNIC is referred to as application and the direction of the arrows is from master to slave.

The primary components of an ETRNIC subsystem are Xilinx Ethernet IP, AXI DMA, and AXI interconnect. On the user application front, the ETRNIC IP exposes side band interfaces to allow efficient doorbell exchanges without going through the interconnect.

Each queue is identified with a set of read and write pointers called the Producer Index (write pointer) and Consumer Index (read pointer). The register address locations for these pointers are termed as doorbells in this document. A doorbell exchange or doorbell ringing indicates that the corresponding register location is updated.

Feature Summary

The ETRNIC IP interfaces with a Network Interface using an AXI Streaming interface. Access to DDR is necessary for reading and writing various data structures for RDMA packet processing. This connection is achieved using multiple AXI4 Full interfaces. The ETRNIC IP works on a 512 bit internal datapath that is completely hardware accelerated and does not require any software intervention for data transfer. All recoverable faults like retransmission due to packet drops are also handled entirely using hardware.

The ETRNIC IP implements Endpoint RNIC functionality. As a result, the following subset of RoCE v2 functionality is implemented (as opposed to full-blown initiator RNIC):

- Support for RDMA SEND for incoming and outgoing packets
- Support for RDMA READ, RDMA WRITE, and RDMA SEND work requests
- Support for up to 128 connections
- No support for dynamic memory registration
- Hardware handshake mechanism for efficient doorbell exchange with the user application

ETRNIC Modules

The ETRNIC IP consists of four main modules that are explained in this section.

- QP Manager
- WQE Processor Engine
- RX PKT Handler
- Response Handler

QP Manager

The QP Manager module houses the configurations for all the QPs and provides an AXI Lite interface to the processor. It also arbitrates across the various SEND Queues and caches the SEND Work Queue Entries (WQEs). These WQEs are then provided to the WQE processor module for further processing. This module also handles the QP pointer updates in the event of retransmission.

WQE Processor Engine

The WQE Processor Engine reads the cached WQEs from the QP Manager module and handles the following tasks:

- Validates the incoming WQE packets for any invalid opcode
- Creates the header for the packets based on the Payload Max Transfer Unit (PMTU) and programs the Scatter Gather Lists (SGLs) for the internal DMA engine
- It also triggers the DMA to start the outgoing packet transfers

The WQE Processor Engine is also responsible for sending the outgoing acknowledgment packets for the incoming RDMA SEND requests.

RX PKT Handler

The RX PKT Handler module receives the incoming packets. The ETRNIC IP handles the following types of incoming RoCE v2 packets:

- RDMA SEND and response packets for RDMA READ (request sent from ETRNIC)
- Acknowledgement packets for RDMA WRITE/RDMA SEND (request sent from ETRNIC)
- Communication management (Management Datagram) packets to QP1

The RX PKT Handler module is responsible for validating the incoming packets. It also triggers outgoing acknowledgment packets for incoming RDMA SEND requests and pushes the packets that pass the validation to the RQ buffer. The RDMA READ responses are channeled to the target application directly.

Response Handler

The Response Handler module manages the outstanding queues. These queues hold the information about all packets sent to the remote host but have not yet been acknowledged or responded to. In addition, this module triggers a re-transmission if the remote host sends a Negative Acknowledgment (NAK). If this module does not receive a response from the remote host within a specified time (timeout value), it triggers a timeout related retransmission.

Applications

The ETRNIC IP can be used in these applications:

- Sensor Data Acquisition and transfer over RoCE V2
- Video/Image capture and transfer over RoCE V2
- Remote storage nodes over RoCE V2

Unsupported Features

The ETRNIC IP does not support these operations:

- Incoming RDMA READ, RDMA WRITE, and ATOMIC operations
- Outgoing ATOMIC operations and OFED Stack APIs
- Incoming/outgoing RDMA SEND packets of 0 length
- Incoming/outgoing RDMA SEND with invalidate (NA)
- Incoming/outgoing RDMA SEND with immediate
- Outgoing RDMA WRITE with immediate
- Maximum RQ buffer size supported is 8MB

Licensing and Ordering

This Xilinx LogiCORE™ IP module is provided under the terms of the Xilinx Core License Agreement. The module is shipped as part of the Vivado® Design Suite. For full access to all core functionalities in simulation and in hardware, you must purchase a license for the core. To generate a full license, visit the product licensing web page. Evaluation licenses and hardware timeout licenses might be available for this core or subsystem. Contact your local Xilinx sales representative for information about pricing and availability.

License Checkers

If the IP requires a license key, the key must be verified. The Vivado design tools have several license checkpoints for gating licensed IP through the flow. If the license check succeeds, the IP can continue generation. Otherwise, generation halts with an error. License checkpoints are enforced by the following tools:

- Vivado synthesis
- Vivado implementation
- write_bitstream (Tcl command)

IMPORTANT: *IP license level is ignored at checkpoints. The test confirms a valid license exists. It does not check IP license level.*

Product Specification

The ETRNIC IP provides Endpoint implementation of a RoCE v2 enabled NIC. The RDMA technology allows for faster movement of data over standard Ethernet while completely offloading CPU bandwidth. The ETRNIC IP core comes with SW drivers that can be ported to any Zynq MPSoC or FPGA devices thereby allowing the ETRNIC IP to function independent of any external processor.

Target RDMA enabled NIC

The Xilinx Embedded Target RDMA enabled NIC (ETRNIC) controller can interface with any user application using one of the following methods:

- Closely integrated HW handshake mechanism which provides a low latency, light weight interface to the target system
- Light weight, proprietary SW API interface that integrates with any target application running on ARM and communicates with the ETRNIC HW

Figure 2-1 shows sample end-to-end system with multiple host CPUs and multiple native NVMe devices talking over a network fabric through the Xilinx ETRNIC + NVMEOFABRIC IP subsystem.

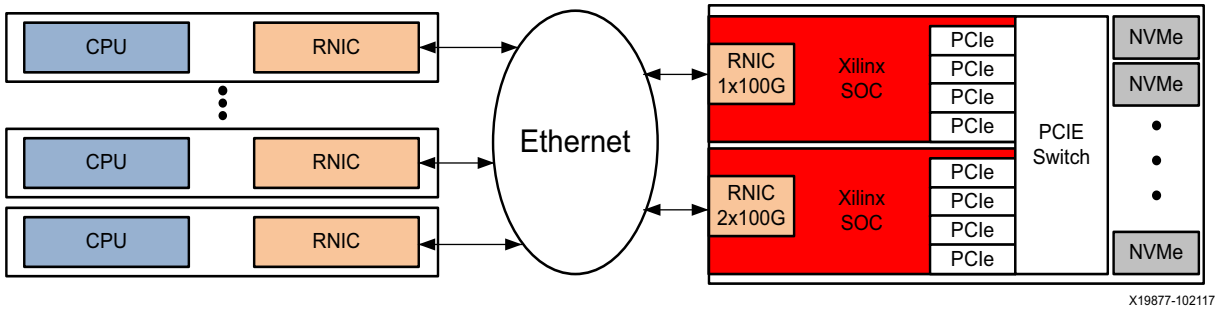

Figure 2-1: **Xilinx ETRNIC + NVMe over Interconnect**

The host side RDMA enabled NICs (RNIC) shown in Figure 2-1 need to support RoCE v2 technology. The following sections describe the key data structures used in the ETRNIC implementation.

Work Requests/Work Queue Entries (WQEs):

Work Requests are used to submit units of work to the ETRNIC IP. The operations which may be posted to the Send Work Queues by the target system are:

- Send
- RDMA Write
- RDMA Read

The following work requests are not allowed:

- ATOMIC
- Bind Memory Window
- Local Invalidate
- Fast Register Physical MR

You need not post Receive work requests, as the ETRNIC Hardware automatically re-posts consumed receive buffers as per the configured receive queue depth.

The structure of send work requests is shown in Table 2-1. Each Work Queue Entry (WQE) is 64 Bytes in size.

Table 2-1: **WQE Structure**

Bitwidth	Content	Size (B)	Comment
[15:0]	WRID	2	Work Request ID. Unique identifier for every WQE
[31:16]	RESERVED	2	
[95:32]	LADDR	8	Local address. Data Buffer address for RDMA READ and RDMA WRITE operations
[127:96]	LENGTH	4	Length of the transfer
[135:128]	OPCODE	1	RDMA_WRITE = 8'h00 RDMA_READ = 8'h04 RDMA_SEND = 8'h02 All other values are reserved.
[159:136]	RESERVED	3	3
[223:160]	ROFFSET	8	Remote offset address
[255:224]	RTAG	4	Remote TAG
[383:256]	SDATA	16	RDMA SEND Data. If the data to be sent is less than or equal to 16B, this field is used to represent the data.
[511:384]	RESERVED	16	

Work Completions:

Work completions are posted for every WQE posted on the Send Queue. Completions are not posted for Receive Queue (RQ) entries, instead doorbells are rung per Queue Pair (QP)

at the address pointed to by RQWPTRDBADDi when a new receive buffer is consumed by an incoming RDMA SEND request. The structure of a Completion Queue Entry (CQE) is given in Table 2-2. Each CQE is 4 bytes in size.

Table 2-2: **Completion Queue Entry structure**

Bitwidth	Content	Size (B)	Comment
[15:0]	WRID	2	Work Request ID. Unique identifier for every WQE
[23:16]	OPCODE	1	• 8'd0 = RDMA WRITE • 8'd1 = RESERVED • 8'd2 = RDMA SEND • 8'd3 = RESERVED • 8'd4 = RDMA READ • 8'd5 - 8'd255 = RESERVED
[31:24]	ERRFLAG	1	Error flag. Set to 8'b1 in case QP enters a fatal state

RDMA Queues

The ETRNIC IP implements RDMA queues like Receive Queue (RQ), Send Queue (SQ), and Completion Queue (CQ). These queues are referred to as Queue Pairs or QPs. SQ houses the send WQEs posted by the user application.

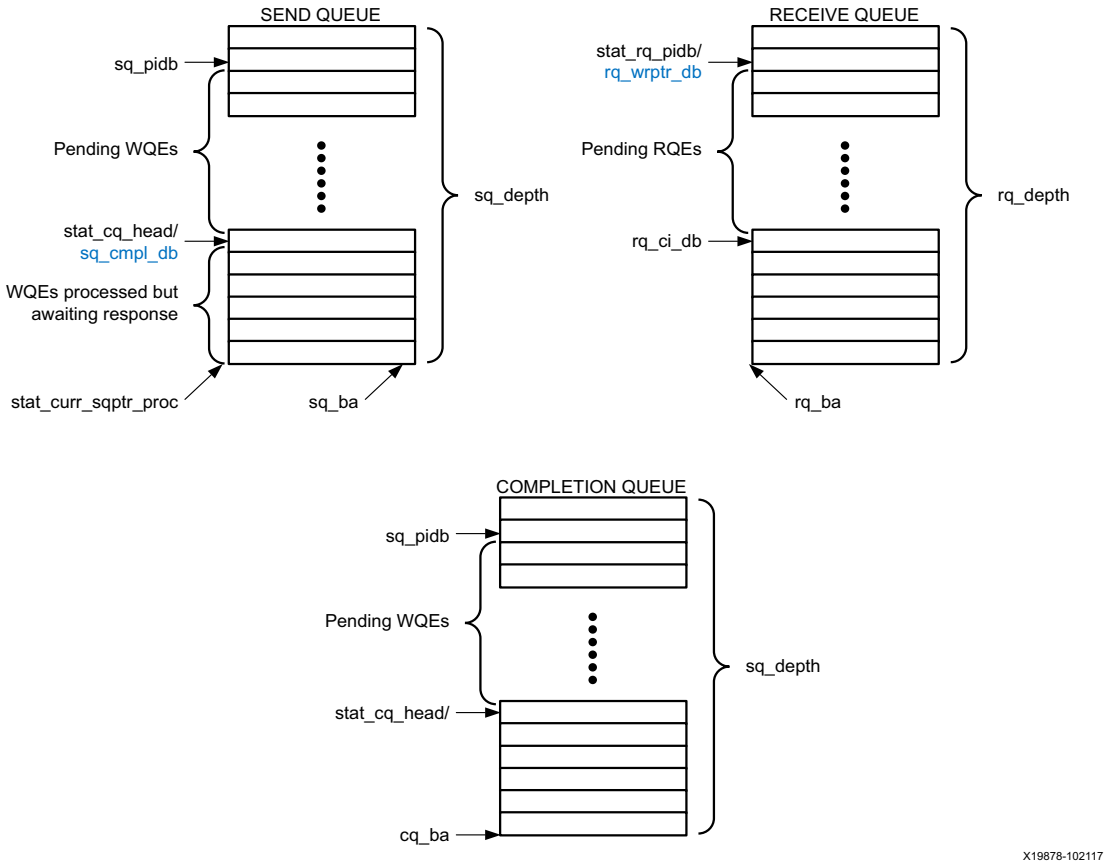

Figure 2-2: **RDMA Queues**

The RQ houses the incoming RDMA SEND packets. The completion queue informs the user application about the completed SEND WQEs. Each of these queues are implemented as circular buffers. Various doorbell and write pointer registers define the current state of these queues. Figure 2-2 shows the different queues and the variables/registers that define the state of the queues. The highlighted variables are indirectly accessed by the ETRNIC IP. The register CQDBADDi points to the address for sq_cmpl_db and the RQWPTRDBADDi register points to the address for the rq_wrptr_db.

These queues are implemented in memory regions outside the ETRNIC IP. The ETRNIC accesses the IP through the various AXI master interfaces. See Table 3-2 for details of ETRNIC memory requirements.

The next few sections provide a brief overview of the incoming (RX) and outgoing (TX) data flow of the ETRNIC IP.

ETRNIC RX Path

The ETRNIC RX Path gets the packet data from the MAC through the AXI streaming interface. All incoming packets are validated and all packet headers that fail packet validation are sent to the error buffer (base address specified by ERRBUFBA[31:0]) if the

value of XRNICCONF[5] is set to 1. The header is prefixed with an error syndrome by the ETRNIC RX packet handler module as per Table 2-3. These buffers provide useful debug information for incoming packet errors.

Table 2-3: **Packet Validation Error Syndrome**

Error syndrome bit index	Error related to	Precedence	Description and impact	Impact
0	MAC	4	MAC destination address received in the Ethernet header does not match the ETRNIC MAC address configured	Packet dropped
1	Reserved			
2	IPv4/IPv6	5	IP Version not as per ETRNIC configuration	Packet dropped
3	IPv4	7	IPv4 header length not equal to 20 bytes	Packet dropped
4	IPv4/IPv6	8	ECN bits in header are equal to 2'b11	Packet dropped
5	IPv4	8	Flag bits in IPv4 are not 3'b010	Packet dropped
6	IPv4	8	Fragment Offset in IPv4 are not 0	Packet dropped
7	Reserved			
8	IPv4/IPv6	8	IPv4 destination address in IPv4 header is not matching with ETRNIC IPv4 address configured	Packet dropped
9	IPv4	6	IPv4 header checksum error.	Packet dropped
10	IPv4/IPv6	8	IPv4 Total Length field value is not with in range. IPv6 Payload Length field value is not with in range	Packet dropped
11	Reserved			
12	UDP	9	UDP Length field is not consistent with IPv4 Total Length or IPv6 Payload Length	Packet dropped
13	BTH	10	BTH Version is not 4'b0000	Packet dropped
14	BTH	10	QP Destination address in BTH is not supported	Packet dropped
15	BTH	11	QP specified in Destination address of BTH is either not configured or not enabled	Packet dropped
16	BTH	14	Opcode sequence in not correct	NAK-Invalid sent and QP moved to FATAL state
17	BTH	10	Unsupported or reserved opcode request is received from remote QP	NAk-Invalid sent and QP moved to FATAL state
18	BTH	14	For FIRST and MIDDLE request/response received BTH Pad bits are not 2'b00	Packet dropped
19	Transport layer	14	Transport Layer Payload is inconsistent with PMTU configured for that QP	Packet dropped

Table 2-3: **Packet Validation Error Syndrome *(Cont'd)***

Error syndrome bit index	Error related to	Precedence	Description and impact	Impact
20	Internal error	13	Request Queue is full and the packet is dropped	RNR-NAK sent
21	Transport layer	12	Incoming request PSN sequence is not correct	NAK-Sequence error
22	Transport layer	14	AETH Syndrome is malformed.	Packet dropped
23	Transport layer	14	Bad response or NAK-Invalid response received	QP moved to FATAL state
24-26	Reserved			
27	IPv4/IPv6	5	IP4(6) source address in IP4(6) header is not matching with QP configured remote node source address	Packet dropped
28	MAC	4	MAC source address in the Ethernet header does not match the QP configured remote node MAC address	Packet dropped
29	Reserved			
30	Link layer	2	ICRC Error	Packet dropped
31	Link layer	1	FCS Error	Packet dropped

Most of the packet validation errors are handled entirely by hardware and no software intervention is required. However, if an incoming packet causes the QP to enter into a FATAL state, software intervention is required to process the error and to initiate a disconnection. Such errors are available for the SW in the incoming packet error status buffers defined by IPKTERRQBA, IPKTERRQSZ, and IPKTERRQWPTR registers. Each error status buffer entry is 64 bit wide. The format for the error status is as shown in Figure 2-3.

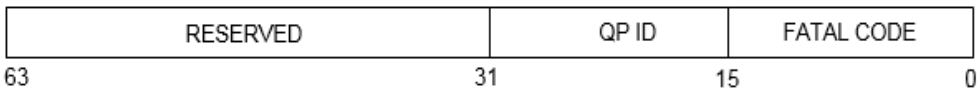

Figure 2-3: **Error status format**

Fatal table decoding is shown in Table 2-4

Table 2-4: **Decoding for FATAL codes**

FATAL Code	Description	Local/Remote error
5'b00001	Request opcode sequence check failed	Locally detected error
5'b00010	Request packet length is not as per PMTU configured	Locally detected error
5'b00011	Both opcode sequence error and packet length occurred simultaneously	Locally detected error

Table 2-4: **Decoding for FATAL codes**

FATAL Code	Description	Local/Remote error
5'b00100	Unsupported request opcode received but with correct PSN	Locally detected error
5'b00101	Reserved	
5'b00110	QP retry count expired	
5'b00111 – 5'b01001	Reserved	
5'b01010	NAK Invalid/RAE/ROE response received	Remote error
5'b01011	Reserved	
5'b01100	Acknowledge response opcode is not correct	Locally detected error
5'b01101	RNR-NAK down counter expired	Locally detected error
5'b01110 – 5'b11000	5'b01110 – 5'b11000	
5'b11y2y1y0	y0 – In READ Response FIRST/LAST/ONLY, AETH syndrome is other than ACK (READ Response considered as BAD)	Locally detected error
	y1 - Read response length is not correct	Locally detected error
	y2 - Read response opcode is not correct	Locally detected error

Only incoming RDMA SEND requests are expected on the RX side. All other types of packets are response packets for the outgoing requests. The data flow for incoming RDMA SEND requests is shown in Figure 2-4. The direction of arrows show the flow of data. On receiving a valid RDMA SEND incoming packet on a connected QP, the packet content (without the headers) is pushed into the RX Buffer for the relevant QP. The ETRNIC rings the RQ Producer Index Doorbell (RQPI DB00000), to indicate that a new packet is available, either using the side band interface or through the AXI interface. This depends on the configuration of QPCONFi[4]. An acknowledgment is also posted to the remote host at this point. The user application may inform the ETRNIC of having consumed the new packet by ringing the RQ consumer Index Doorbell (RQCI DB). On receiving this doorbell, the corresponding RX buffer is made available to be used for new incoming packets.

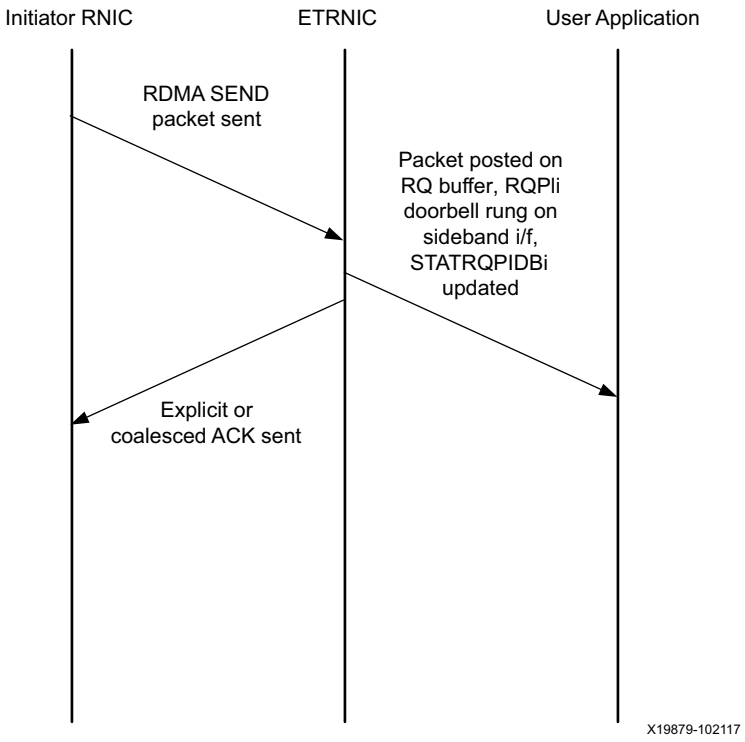

Figure 2-4: **RDMA SEND RX data flow**

ETRNIC TX Path

The TX Path consists of outgoing RDMA READ, RDMA WRITE transactions, and ACK packets for incoming RDMA SEND requests. Based on the SQPIi doorbell, the Send Work Queue requests are processed. The AXI DMA module is configured for data transfers for all outgoing transactions. The TX data flow for RDMA WRITE/SEND is shown in Figure 2-5 and Figure 2-6.

The user application may request the ETRNIC IP to transmit an RDMA WRITE/SEND packet by posting a WQE on the SEND Queue for a particular QP and ringing the corresponding SQ Producer Index Doorbell (SQPI DB). The ETRNIC processes the WQE and pulls data for RDMA WRITE/SEND commands based on the information provided in the WQE. This data along with relevant headers is pushed out on the link. Once an acknowledgment is received from the remote host, the ETRNIC informs the user application of the successful completion of the WQE by posting a CQE, based on the configuration of QPCONFi[5], and by posting the completion count through the side band interface. The CQHEAD for the corresponding QP is also updated.

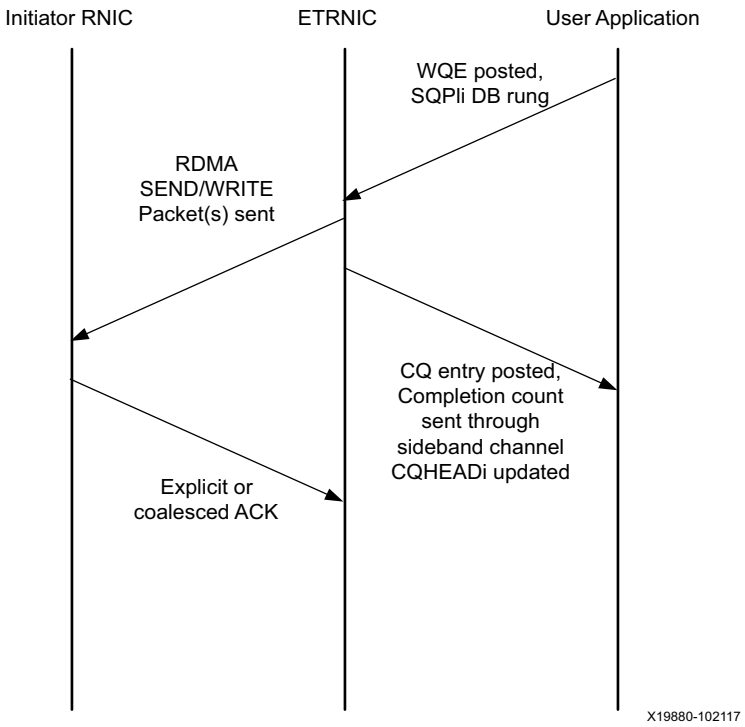

Figure 2-5: **RDMA SEND/WRITE data flow - 1**

Note: The direction of arrows shows the flow of data.

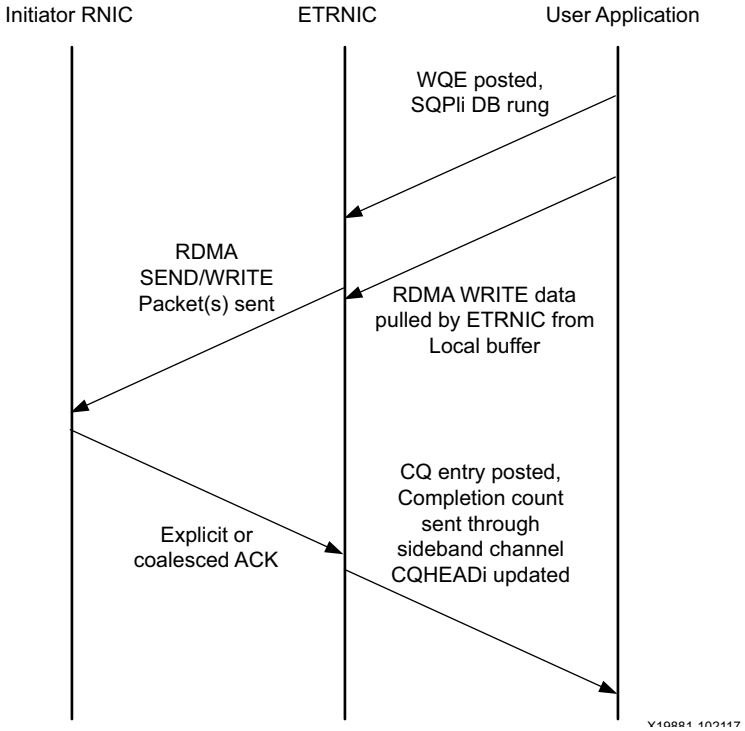

Figure 2-6: **RDMA SEND/WRITE data flow - 2**

Figure 2-7 shows the flow of RDMA READ requests to the remote host from the ERTNIC.

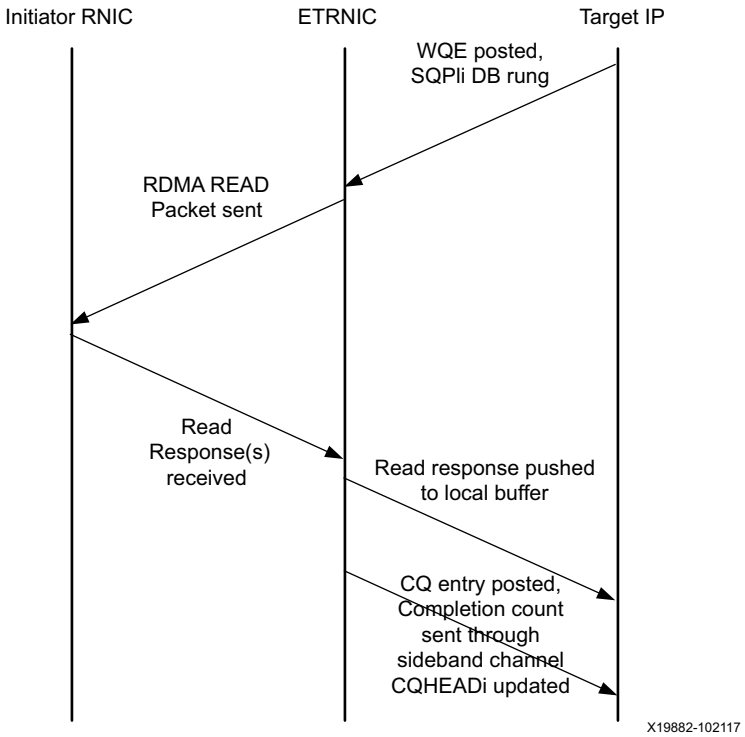

Figure 2-7: **RDMA READ data flow**

Figure 2-7 shows the flow for RDMA READ requests to the remote host from the ETRNIC. The user application may request the ETRNIC IP to transmit an RDMA READ request to the remote host by posting a WQE on the SEND Queue for a particular QP and ringing the corresponding SQ Producer Index Doorbell (SQPI DB). ETRNIC processes the WQE and creates the request packet with relevant headers and pushes it out on the link. Once the response data packets are received from the remote host, ETRNIC writes the data (after removing the headers) to the local buffer address provided in the WQE. It then informs the user application of the successful completion of the WQE by posting a CQE, based on the configuration of QPCONFi[5], and by posting the completion count through the side band interface. The CQHEAD for the corresponding QP is also updated.

Standards

This implementation is based on the standard and specifications described in InfiniBandTM Architecture Specification Volume 1, Annex A16 RoCE and Annex 17 RoCE V2.

Performance

The ETRNIC IP is designed with an internal data path throughput of up to 100 Gb per second at a frequency of 200 MHz.

For more details resource utilization, see, Resource Utilization.

Resource Utilization

This section summarizes the estimated maximum performance for various modules within the ETRNIC IP. The data is separated into a table per device family. Each row describes a test case. The columns are divided into test parameters and results. The test parameters include the part information and the core-specific configuration parameters. Any configuration parameters that are not listed have their default values; any parameters with a blank value are disabled or set automatically by the IP core. Consult the product guide for this IP core for a list of GUI parameter and user parameter mappings.

- Resource use numbers are taken from the utilization report issued at the end of an implementation using the Out-of-Context flow in Vivado Design Suite.
- The Out-of-Context IP constraints include HD.CLK_SRC properties as required to ensure correct hold timing closure. These properties are enabled using the following Tcl command: `set_param ips.includeClockLocationConstraints true`
- The frequencies used for clock inputs are stated for each test case.
- LUT numbers do not include LUTs used as pack-thrus, but include LUTs used as memory.
- Default Vivado Design Suite 2018.1 settings are used. You can improve on these numbers using different settings. However, because the surrounding circuitry will affect placement and timing, these numbers might not repeat in a larger design.

For more details about resource utilization, see Performance and Resource Utilization.

Port Descriptions

Table 2-5 table describes the ports and their interface definitions

Table 2-5: **ETRNIC IP Ports**

Name	Direction	Width	Clock Domain	Description
Clocking and Reset				
m_axi_aclk	Input	1	AXI4	AXI4 clock
m_axi_aresetn	Input	1		AXI4 reset (active low)
s_axi_lite_aclk	Input	1	AXI4	AXI lite clock
s_axi_lite_aresetn	Input	1		AXI lite reset (active low)
system_resetn	Output	1		System reset
RX Packet Handler AXI4 master interface to DDR				
rx_pkt_hndler_ddr_m_axi*	I/O		AXI4	This interface is used by the RX Packet handler module to push the incoming RDMA SEND packets to RQ buffer. See Appendix A of the *Vivado Design Suite: AXI Reference Guide* (UG1037) [Ref 7] for more information on the AXI4 signal.
RX Packet Handler AXI4 master read response interface				
rx_pkt_hndler_rd rsp_m_axi*	I/O		AXI4	This interface Is used by the RX Packet handler module to push RDMA READ Response packets to the local buffer as specified in the WQE. See Appendix A of the *Vivado Design Suite: AXI Reference Guide* (UG1037) [Ref 7] for more information on the AXI4 signal.
RX Packet Handler AXI4 streaming slave interface to ENET controller				
rx_pkt_hndler_s_axis_*	I/0		AXI stream	This interface provides the incoming stream of data from the network interface to the RX Packet handler module. See Appendix A of the *Vivado Design Suite: AXI Reference Guide* (UG1037) [Ref 7] for more information on the AXI4 signal.
WQE Processor AXI master interface				
wqe_proc_top_m_axi_*	I/O		AXI4	This interface is used by WQE Processor engine to read RDMA SEND/WRITE data from the local buffer. This interface is also used to update the Outgoing Error Packet Queue in case of an error. See Appendix A of the *Vivado Design Suite: AXI Reference Guide* (UG1037) [Ref 7] for more information on the AXI4 signal.

Table 2-5: **ETRNIC IP Ports *(Cont'd)***

Name	Direction	Width	Clock Domain	Description
WQE Processor AXI4 master interface to DDR				
wqe_proc_wr_ddr_m_axi_*	I/O		AXI4	This interface is used by the Response Handler module to write the Completion Queue entries and access the Submission Queue entries. See Appendix A of the *Vivado Design Suite: AXI Reference Guide* (UG1037) [Ref 7] for more information on the AXI4 signal.
AXI Lite slave interface for register programming				
s_axi_lite_*	I/O		AXI4 Lite	This interface is used by the processor to configure the ETRNIC registers. See Appendix A of the *Vivado Design Suite: AXI Reference Guide* (UG1037) [Ref 7] for more information on the AXI4 signal.
QP manager AXI4 master interface				
qp_mgr_m_axi_*	I/O		AXI4	This interface is used by the QP Manager module to read the Submission Queue entries. See Appendix A of the *Vivado Design Suite: AXI Reference Guide* (UG1037) [Ref 7] for more information on the AXI4 signal.
HW handshake ports for RQ Doorbells				
rx_pkt_hndler_o_rq_db_data	O	32	AXI4	RDMA-SEND Producer Index Doorbell Value from ETRNIC
rx_pkt_hndler_o_rq_db_addr	O	10	AXI4	RDMA-SEND Producer Index Doorbell Address (4 Bytes per QP; for 127 QPs)
rx_pkt_hndler_o_rq_db_data_valid	O	1	AXI4	RDMA-SEND Producer Index Doorbell Valid. When this signal is asserted HIGH, rx_pkt_hndler_o_rq_db_addr and rx_pkt_hndler_o_rq_db_data are valid. Until rx_pkt_hndler_i_rq_db_rdy is not sampled HIGH, this signal remains asserted and, rx_pkt_hndler_o_rq_db_addr and rx_pkt_hndler_o_rq_db_data signals will hold the same values.
rx_pkt_hndler_i_rq_db_rdy	I	1	AXI4	Ready from the target signaling data and address are accepted
HW handshake ports for CQ Doorbells				
resp_hndler_o_send_cq_db_cnt	O	32	AXI4	Send WQE Completion queue Doorbell count from ETRNIC
resp_hndler_o_send_cq_db_addr	O	10	AXI4	Send WQE Completion queue Doorbell address (4 Bytes per QP; for 127 QPs)

Table 2-5: **ETRNIC IP Ports *(Cont'd)***

Name	Direction	Width	Clock Domain	Description
resp_hndler_o_send_cq_db_cnt_valid	O	1	AXI4	Send WQE Completion Doorbell Valid. When this signal is asserted HIGH, resp_hndler_o_send_cq_db_addr and resp_hndler_o_send_cq_db_cnt are valid. Until resp_hndler_i_send_cq_db_rdy is not sampled HIGH, this signal remains asserted and, resp_hndler_o_send_cq_db_addr and resp_hndler_o_send_cq_db_cnt signals will hold the same values.
resp_hndler_i_rq_db_rdy	I	1	AXI4	Send WQE Completion Doorbell Ready. Ready signal should go HIGH when the target application accepts the current doorbell transaction.
HW handshake ports for SQ PI Doorbells				
i_qp_sq_pidb_hndshk	I	16	AXI4	Send WQE Producer Index Doorbell Value from target application
i_qp_sq_pidb_wr_addr_hndshk	I	32	AXI4	Send WQE Producer Index Doorbell Address
i_qp_sq_pidb_wr_valid_hndshk	I	1	AXI4	Send WQE Producer Index Doorbell Valid. After the target application posts WQE(s), it should assert this signal HIGH with valid i_qp_sq_pidb_wr_addr_hndshk and i_qp_sq_pidb_hndshk. Target application should keep this signal asserted and hold i_qp_sq_pidb_wr_addr_hndshk and i_qp_sq_pidb_hndshk signals until it samples o_qp_sq_pidb_wr_rdy as HIGH.
o_qp_sq_pidb_wr_rdy	O	1	AXI4	Send WQE Producer Index Doorbell Ready. Ready signal asserted HIGH when ETRNIC accepts the Doorbell value
HW handshake ports for RQ CI Doorbells				
i_qp_rq_cidb_hndshk	I	16	AXI4	RDMA-SEND Completion Index Doorbell value from target application
i_qp_rq_cidb_wr_addr_hndshk	I	32	AXI4	RDMA-SEND Completion Index Doorbell register address
i_qp_rq_cidb_wr_valid_hndshk	I	1	AXI4	RDMA-SEND Consumer Index Doorbell Valid. After target application processes incoming RDMA-SEND command(s), it should assert this signal HIGH with valid i_qp_rq_cidb_wr_addr_hndshk and i_qp_rq_cidb_hndshk. Target application should keep this signal asserted and hold i_qp_rq_cidb_wr_addr_hndshk and i_qp_rq_cidb_hndshk signals until it samples o_qp_rq_cidb_wr_rdy as HIGH.

Table 2-5: **ETRNIC IP Ports *(Cont'd)***

Name	Direction	Width	Clock Domain	Description
o_qp_rq_cidb_wr_rdy	O	1	AXI4	RDMA SEND Completion Index Doorbell value is accepted by ETRNIC
Interrupts				
xrnic_intr0	O	1	AXI4	Level sensitive interrupt. Asserted HIGH when incoming IB packet fails Header validation rules.
xrnic_intr1	O	1	AXI4	Level sensitive interrupt. Asserted HIGH when MAD packet received from remote host.
xrnic_intr2	O	1	AXI4	Reserved
xrnic_intr3	O	1	AXI4	Level sensitive interrupt. Asserted HIGH whenever RNR-NAK acknowledge is sent to remote host.
xrnic_intr4	O	1	AXI4	Level sensitive interrupt. Asserted HIGH when WQE completion is updated to RDMA.
xrnic_intr5	O	1	AXI4	Level sensitive interrupt. Asserted HIGH when RDMA posts WQE with illegal opcode.
xrnic_intr6	O	1	AXI4	Level sensitive interrupt. Asserted HIGH when RDMA-SEND command received from remote host.
xrnic_intr7	O	1	AXI4	Level sensitive interrupt. Asserted HIGH when any QP moved to fatal state either due remote/local error.
xrnic_intr8	O	1	AXI4	Reserved

Parameter Descriptions

As many features in the ETRNIC controller design are controlled using parameters, the controller implementation can be uniquely tailored using only the resources required for the desired functionality. This approach also achieves the best possible performance with the lowest resource usage.

Table 2-6 lists the parameter descriptions and default values.

Table 2-6: **ETRNIC Parameters**

Parameter name	Description	Default Value
C_NUM_QP	Maximum number of Queue Pairs	8 Minimum - 8 Maximum - 128
C_EN_HW_HHDSHK	Enables HW handshake mechanism for QPs based on the configuration	1

Table 2-6: **ETRNIC Parameters *(Cont'd)***

Parameter name	Description	Default Value
C_M_AXI_ADDR_WIDTH	AXI Master address width (not user changeable)	32
C_M_AXI_ID_WIDTH	AXI Master ID width	1
C_MAX_SGL_DEPTH	Maximum number of SGLs to be stored for the TX DMA	128
C_MAX_WR_RETRY_DATA_BUF_DEPTH	Maximum number of data buffers for RDMA WRITE in case of a retransmission. Each RDMA data buffer is 4.	128
C_S_AXI_LITE_ADDR_WIDTH	AXI-Lite slave address width	32
C_EN_WR_RETRY_DATA_BUF	Enable storing RDMA WRITE data in DDR for retransmission	1

Register Space

All the ETRNIC registers are synchronous to the AXI4-Lite domain. Any bits not specified in register tables below are considered reserved and return the values as 0 upon read. The power-on reset values of control registers are 0 unless specified in the definition. You should always write the reserved locations with a 0 unless stated otherwise. Only address offsets are listed in the table below and the base address is configured by the AXI interconnect at system level.

Table 2-7: **ETRNIC Registers details**

Address	Access	Register Name	Details
0x0000	RW	XRNIC Configuration (XRNICCONF)	This register provides the basic global (not QP specific) configuration controls for the ETRNIC IP • [0]: XRNIC Enable • [1]: Reserved • [2]: Reserved • [4:3]: TX Ack generation. ◦ 00 – (default) ACK only generated on explicit ACK request in the incoming packet or on timeout ◦ 01 – ACK generated only on timeout ◦ 10 – ACK generated only on explicit ACK request in the incoming packet • [7:5]: Reserved • [15:8]: Number of QPs enabled. Maximum sequential number • [31:16]: UDP source port for outgoing packets
0x0004	RW	XRNIC Advance Configuration (XRNICADCONF)	This register provides advanced configuration controls for the ETRNIC IP • [0]: SW override enable. Allows SW write access to the following Read Only Registers – CQHEADn, STATCURRSQPTRn and STATRQPIDBn (where is the QP number) • [1]: Reserved • [2]: retry_cnt_fatal_dis • [15:3]: Reserved • [19:16]: Base count width ◦ Approximate number of system clocks that make 4096us. ◦ For 400 MHz clock -->Program decimal 11 ◦ For 200 MHz clock --> Program decimal 10 ◦ For 125 MHz clock --> Program decimal 09 ◦ For 100 MHz clock --> Program decimal 09 ◦ For N MHz clock ---> Value should be CLOG2(4.096 *N) • [20:23]: Reserved • [31:24] Software Override QP Number

Table 2-7: **ETRNIC Registers details *(Cont'd)***

Address	Access	Register Name	Details
0x0010	RW	MAC XRNIC Address LSB (MACXADDLSB)	This is the MAC Address for local (ETRNIC) device and should be configured before the XRNICCONF[0] is set to 1. • [31:0]: MAC local address LSB
0x0014	RW	MAC XRNIC Address MSB (MACXADDMSB)	MAC Address for local (ETRNIC) device. Should be configured before the XRNICCONF[0] is set to 1. • [15:0]: MAC local address MSB • [31:16]: Reserved
0x0020	RW	IPv6 Address 1 (IPv6XADD1)	IPv6 address [32:0] for local (ETRNIC) device. Should be configured before the XRNICCONF[0] is set to 1. [31:0]: IP address [31:0]
0x0024	RW	IPv6 Address 2 (IPv6XADD2)	IPv6 address [63:32] for local (ETRNIC) device. Should be configured before the XRNICCONF[0] is set to 1. [31:0]: IP address [63:32]
0x0028	RW	IPv6 Address 3 (IPv6XADD3)	IPv6 address [95:64] for local (ETRNIC) device. Should be configured before the XRNICCONF[0] is set to 1. [31:0]: IP address [95:64]
0x002C	RW	IPv6 Address 4 (IPv6XADD4)	IPv6 address [127:96] for local (ETRNIC) device. Should be configured before the XRNICCONF[0] is set to 1. [31:0]: IP address [127:96]
0x0060	RW	Error Buffer Base Address (ERRBUFBA)	This register provides the base address for Error buffers. The ETRNIC IP updates these buffers with incoming packets that fail validation. The writes to this buffer for all validation errors are enabled by writing a 1 to XRNICCONF[5]. If this bit is disabled, only packets that cause the QP to move to a FATAL state are written to the error buffer. • [31:0]: Error buffer base address
0x0068	RW	Error Buffer Size (ERRBUFSZ)	This register provides the size of the error buffer • [15:0]: Number of Error buffers • [31:16]: Size of each Error buffer
0x006C	RO	Error Buffer write pointer (ERRBUFWPTR)	This register is updated by the ETRNIC IP when any error packet is written to the error buffer. The pointer informs the driver of the number of error packets received. • [15:0]: Write pointer doorbell for Error Buffer
0x0070	RW	IPv4 XRNIC Address (IPv4XADD)	IPv4 address for local (ETRNIC) device. • [31:0]: IPv4 address
0x0088	RW	Incoming Pkt Error Status Queue Base address (IPKTERRQBA)	This register is used to configure the base address for incoming packet error status queue. See, ETRNIC RX Path for details of the incoming packet error status queue. [31:0]: Outgoing Pkt Error Status Queue base address
0x0090	RW	Incoming Pkt Error Status Queue Size (IPKTERRQSZ)	This register is used to configure the size of the incoming packet error status queue. • [15:0]: Number of incoming error pkt status queue entries • [31:16]: Reserved

Table 2-7: **ETRNIC Registers details *(Cont'd)***

Address	Access	Register Name	Details
0x0094	RO	Incoming Pkt Error Status write pointer (IPKTERRQWPTR)	This status register provides information on the number of incoming error packets received by the ETRNIC IP. [15:0]: Write pointer doorbell for incoming error status queue. ETRNIC IP writes to the queue in a circular manner without taking care of overflow. This needs to be handled in SW.
0x00A0	RW	Data Buffer Base Address (DATBUFBA)	These data buffers are used by the ETRNIC to save all outgoing RDMA WRITE data until it is acknowledged by the remote host. In the event of retransmission, the retried data is pulled from these buffers. [31:0]: Data Buffer base address
0x00A8	RW	Data Buffer Size (DATBUFSZ)	This register is used to configure the size of the data buffers. • [15:0]: Number of data buffers • [31:16]: Data buffer size in bytes
0x00AC	RO	Reserved	
Global status registers			
0x0100	RO	Incoming SEND/ Read Response Pkt count (INSRRPKTCNT)	This is a status register which provides information about incoming RDMA SEND and RDMA READ response packets. • [15:0]: Incoming SEND packet count • [31:16]: Incoming Read Response packet count
0x0104	RO	Incoming ACK/MAD Pkt count (INAMPKTCNT)	This is a status register which provides information about incoming acknowledgement and Management Datagram (MAD) packets. • [15:0]: Incoming ACK packet count • [31:16]: Incoming MAD packet count
0x0108	RO	Outgoing IO (SEND/READ/ WRITE) Pkt count (OUTIOPKTCNT)	This is a status register which provides information about outgoing RDMA SEND and RDMA READ/WRITE packets. • [15:0]: Outgoing SEND packet count • [31:16]: Outgoing RDMA READ/WRITE packet count
0x010C	RO	Outgoing ACK/ MAD Pkt count (OUTAMPKTCNT)	This is a status register which provides information about outgoing acknowledgment and MAD packets. • [15:0]: Outgoing ACK packet count • [31:16]: Outgoing MAD packet count
0x0110	RO	Last incoming pkt (LSTINPKT)	This status register provides details of the last incoming packet received. • [7:0]: Opcode of the last incoming packet • [15:8]: QPID of the last incoming packet • [31:16]: PSN LSB bits of the last incoming packet
0x0114	RO	Last outgoing pkt (LSTOUTPKT)	This status register provides the details of the last outgoing packet. • [7:0]: Opcode of the last outgoing packet • [15:8]: QPID of the last outgoing packet • [31:16]: PSN LSB bits of the last outgoing packet

Table 2-7: **ETRNIC Registers details *(Cont'd)***

Address	Access	Register Name	Details
0x0118	RO	Incoming Invalid/ Duplicate pkt count (ININVDUPCNT)	This status register provides information on incoming invalid or duplicate packets. • [15:0]: Incoming Invalid packet count • [31:16]: Incoming Duplicate packet count
0x011C	RO	Incoming NAK pkt Status (INNCKPKTSTS)	This status register is provides details of incoming Responder Not Ready (RNR) NAK, Retry Count expired or other NAK received. • [7:0]: QPID for incoming NAK packet • [15:8]: Reserved • [31:16]: Incoming NAK packet count
0x0120	RO	Outgoing RNR pkt Status (OUTRNRPKTSTS)	This status register is provides details of outgoing Responder Not Ready (RNR) NAK sent. • [7:0]: QPID for outgoing RNR Packet • [15:8]: Reserved • [31:16]: Count of RNR Packets sent
0x0124	RO	WQE Processor Status (WQEPROCSTS)	This status register provides the status of WQE processor engine and is used for debug purposes. • [2:0]: WQE processor state • [11:3]: Last error opcode received in WQE • [12]: reserved • [15:13]: Buffer manager status • [23:16]: SGL buffer tail pointer • [31:24]: SGL buffer head pointer
0x012C	RO	QP Manager Status (QPMSTS)	This status register provides the status of QP manager and is used for debug purposes. • [0]: WQE Cache full • [1]: WQE Cache empty • [15:2]: Reserved • [31:16]: Count of WQEs processed
0x0130	RO	Incoming all/ dropped pkt count (INALLDRPPKTCNT)	This status register is provides details of all incoming and dropped packets. • [15:0]: All incoming packet count • [31:16]: Incoming dropped packet count
0x0134	RO	Incoming NAK pkt count (INNAKPKTCNT)	This status register is provides details of all incoming NAK packets. • [15:0]: Incoming NAK packet count • [31:16]: Reserved
0x0138	RO	Outgoing NAK pkt count (OUTNAKPKTCNT)	This status register is provides details of all outgoing NAK packets. • [15:0]: Outgoing NAK packet count • [31:16]: Reserved

Table 2-7: **ETRNIC Registers details *(Cont'd)***

Address	Access	Register Name	Details
0x013C	RO	Response handler Status (RESPHNDSTS)	This status register provides the status of Response Handler module and is used for debug purposes. • [7:0]: Arbitrated QP Index • [13:8]: Response handler FSM state • [16:14]: Reserved • [17]: Acks to process • [31:18]: Reserved
0x0140	RO	Retry count status (RETRYCNTSTS)	This status register provides details of retransmitted packets. [31:0] Retry count
0x0180	RW	Interrupt Enable (INTEN)	This register provides the configuration control for interrupts generated by the ETRNIC IP. • [0]: Incoming packt validation error interrupt enable • [1]: Incoming MAD packet received interrupt enable • [2]: Bypass packet received interrupt enable • [3]: RNR NACK generated interrupt enable • [4]: WQE completion interrupt enable (asserted for QPs for which QPCONF[3] bit is set) • [5]: Illegal opcode posted in SEND Queue interrupt enable • [6]: RQ Packet received interrupt enable (asserted for QPs for which QPCONF[2] bit is set • [7]: Fatal error received interrupt enable
0x0184	W1C	Interrupt Status (INTSTS)	This status register provides the cause of an asserted interrupt. • [0]: Incoming packt validation error interrupt • [1]: Incoming MAD packet received interrupt • [2]: Bypass packet received interrupt • [3]: RNR NACK generated interrupt • [4]: WQE completion interrupt (asserted for QPs for which QPCONF[3] bit is set) • [5]: Illegal opcode posted in SEND Queue interrupt • [6]: RQ Packet received interrupt (asserted for QPs for which QPCONF[2] bit is set) • [7]: Fatal error received interrupt
0x0190	RO	Rcv Q interrupt status (bitwise) QP 0-31 (RQINTSTS1)	This registers provides the RQ interrupt status for QPs 0 to 31. Bit [i] provides the interrupt status for RQi where i=0 to 31. [i]: Interrupt status for RQ i
0x0194	RO	Rcv Q interrupt status (bitwise) QP 32-63 (RQINTSTS2)	This registers provides the RQ interrupt status for QPs 32 to 63. Bit [i] provides the interrupt status for RQ(i+32) where i=0 to 31. [i]: Interrupt status for RQ(i+32)
0x0198	RO	Rcv Q interrupt status (bitwise) QP 64-95 (RQINTSTS3)	This registers provides the RQ interrupt status for QPs 64 to 95. Bit [i] provides the interrupt status for RQ(i+64) where i=0 to 31. [i]: Interrupt status for RQ(i+64)

Table 2-7: **ETRNIC Registers details *(Cont'd)***

Address	Access	Register Name	Details
0x019C	RO	Rcv Q interrupt status (bitwise) QP 96-127 (RQINTSTS4)	This registers provides the RQ interrupt status for QPs 96 to 127. Bit [i] provides the interrupt status for RQ(i+96) where i=0 to 31. [i]: Interrupt status for RQ(i+96)
0x01A0	RO	Rcv Q interrupt status (bitwise) QP 128-159 (RQINTSTS5)	This registers provides the RQ interrupt status for QPs 128 to 159. Bit [i] provides the interrupt status for RQ(i+128) where i=0 to 31. [i]: Interrupt status for RQ(i+128)
0x01A4	RO	Rcv Q interrupt status (bitwise) QP 160-191 (RQINTSTS6)	This registers provides the RQ interrupt status for QPs 160 to 191. Bit [i] provides the interrupt status for RQ(i+160) where i=0 to 31. [i]: Interrupt status for RQ(i+160)
0x01A8	RO	Rcv Q interrupt status (bitwise) QP 192-223 (RQINTSTS7)	This registers provides the RQ interrupt status for QPs 192 to 223. Bit [i] provides the interrupt status for RQ(i+192) where i=0 to 31. [i]: Interrupt status for RQ(i+192)
0x01AC	RO	Rcv Q interrupt status (bitwise) QP 224-255 (RQINTSTS8)	This registers provides the RQ interrupt status for QPs 224 to 255. Bit [i] provides the interrupt status for RQ(i+224) where i=0 to 31. [i]: Interrupt status for RQ(i+224)
0x01B0	RO	Cmpletion Q interrupt status (bitwise) QP 0-31 (CQINTSTS1)	This registers provides the CQ interrupt status for QPs 0 to 31. Bit [i] provides the interrupt status for CQi where i=0 to 31. [i]: Interrupt status for CQ i
0x01B4	RO	Cmpletion Q interrupt status (bitwise) QP 32-63 (CQINTSTS2)	This registers provides the CQ interrupt status for QPs 32 to 63. Bit [i] provides the interrupt status for CQ(i+32) where i=0 to 31. [i]: Interrupt status for CQ(i+32)
0x01B8	RO	Cmpletion Q interrupt status (bitwise) QP 64-95 (CQINTSTS3)	This registers provides the CQ interrupt status for QPs 64 to 95. Bit [i] provides the interrupt status for CQ(i+64) where i=0 to 31. [i]: Interrupt status for CQ(i+64)
0x01BC	RO	Cmpletion Q interrupt status (bitwise) QP 96-127 (CQINTSTS4)	This registers provides the CQ interrupt status for QPs 96 to 127. Bit [i] provides the interrupt status for CQ(i+96) where i=0 to 31. [i]: Interrupt status for CQ(i+96)
0x01C0	RO	Cmpletion Q interrupt status (bitwise) QP 128-159 (CQINTSTS5)	This registers provides the CQ interrupt status for QPs 128 to 159. Bit [i] provides the interrupt status for CQ(i+128) where i=0 to 31. [i]: Interrupt status for CQ(i+128)

Table 2-7: **ETRNIC Registers details *(Cont'd)***

Address	Access	Register Name	Details
0x01C4	RO	Cmpletion Q interrupt status (bitwise) QP 160-191 (CQINTSTS6)	This registers provides the CQ interrupt status for QPs 160 to 191. Bit [i] provides the interrupt status for CQ(i+160) where i=0 to 31. [i]: Interrupt status for CQ(i+160)
0x01C8	RO	Cmpletion Q interrupt status (bitwise) QP 192-223 (CQINTSTS7)	This registers provides the CQ interrupt status for QPs 192 to 223. Bit [i] provides the interrupt status for CQ(i+192) where i=0 to 31. [i]: Interrupt status for CQ(i+192)
0x01CC	RO	Cmpletion Q interrupt status (bitwise) QP 224-255 (CQINTSTS8)	This registers provides the CQ interrupt status for QPs 224 to 255. Bit [i] provides the interrupt status for CQ(i+224) where i=0 to 31. [i]: Interrupt status for CQ(i+224)
Per QP registers			
0x0200 + ((i-1) x 0x0100)	RW	QP Configuraion QPi (QPCONFi)	This register defines the basic configuration of QP. This register is generated based on the parameter C_NUM_QP (where i = 1 to C_NUM_QP). • [0]: QP enable – Should be set to 1 for all active QPs. A disabled QP will not be able to receive or transmit packets. • [2]: RQ interrupt enable – When enabled, allows the receive queue interrupt to be generated for every new packet received on the receive queue • [3]: CQ interrupt enable – When enabled, allows the completion queue interrupt to be generated for every send work queue entry completion • [4]: HW Handshake disable – This bit when reset to 0 enables the HW handshake ports for doorbell exchange. If set, all doorbell values are exchanged through writes through the AXI or AXI lite interface. • [5]: CQE write enable – This bit when set, enables completion queue entry writes. The writes are disabled when this bit is reset. CQE writes can be enabled to debug failed completions. • [7]: QP configured for IPv4 or IPv6 ◦ 0 - IPv4 ◦ 1 - IPv6 • [10:8]: Path MTU ◦ 000 – 256B (default) ◦ 001 – 512B ◦ 010 – 1024B ◦ 011 – 2048B ◦ 100 - 4096B ◦ 101 to 111 - Reserved • [31:16]: RQ Buffer size (in multiple of 256B)

Table 2-7: **ETRNIC Registers details *(Cont'd)***

Address	Access	Register Name	Details
0x0204 + ((i-1) x 0x0100)	RW	QP advanced configuration QPi (QPADVCONFi)	This register is generated based on the parameter C_NUM_QP (where i = 2 to C_NUM_QP). This register is not present for QP1. • [5:0]: Traffic class (keep default reset value) • [15:8]: Time to live • [31:16]: Partition Key
0x0208 + ((i-1) x 0x0100)	RW	Rcv Q Buffer base address QPi (RQBAi)	This register is generated based on the parameter C_NUM_QP (where i = 1 to C_NUM_QP). This register provides the base address of the RDMA Receive queue buffer. [31:8]: Receive Q Buffer Base address addr (256 B aligned)
0x0210 + ((i-1) x 0x0100)	RW	SEND Q base addres QPi (SQBAi)	This register is generated based on the parameter C_NUM_QP (where i = 1 to C_NUM_QP). This register provides the base address of the RDMA Send queue buffer. [31:5]: Send Q base address (32 B aligned)
0x0218 + ((i-1) x 0x0100)	RW	CQ base address QPi (CQBAi)	This register is generated based on the parameter C_NUM_QP (where i = 1 to C_NUM_QP). This register provides the base address of the RDMA Completion queue buffer. [31:5]: Send CQ base address (32 B aligned)
0x0220 + ((i-1) x 0x0100)	RW	Rcv Q Write pointer DB address QPi (RQWPTRDBADDi)	This register is generated based on the parameter C_NUM_QP (where i = 1 to C_NUM_QP). This register provides the address of the Receive Queue doorbell register. Upon reception of a new incoming RDMA SEND packet, the ETRNIC IP updates the RQ doorbell values in the address pointed to by this register. [31:0]: Rcv Q write pointer Doorbell address
0x0228 + ((i-1) x 0x0100)	RW	CQ DB address QPi (CQDBADDi)	This register is generated based on the parameter C_NUM_QP (where i = 1 to C_NUM_QP). this register provides the address of the Completion Queue doorbell register. Upon completion of a new SEND Work queue entry, the ETRNIC IP updates the CQ doorbell values in the address pointed to by this register. [31:0]: Send CQ Doorbell address
0x0230 + ((i-1) x 0x0100)	RO	CQ head pointer QPi (CQHEADi)	This register is generated based on the parameter C_NUM_QP (where i = 1 to C_NUM_QP). This status register gives the Send completion Queue doorbell value and provides information about the Send WQEs that have been completed. This is the doorbell value that is written by the ETRNIC IP to the address pointed to by the CQDBADDi register. • [15:0]: CQ head pointer • [31:0]: Reserved

Table 2-7: **ETRNIC Registers details *(Cont'd)***

Address	Access	Register Name	Details
0x0234 + ((i-1) x 0x0100)	RW	RQ Consumer Index QPi (RQCIi)	This register is generated based on the parameter C_NUM_QP (where i = 1 to C_NUM_QP). This register is updated by the target application on consuming a new receive queue entry. Once consumed, the RQ entry can be over written by the ETRNIC IP. The RQCI index can also be updated through the side band interface. • [15:0]: Rcv Q Consumer Index Doorbell • [31:16]: Reserved
0x0238 + ((i-1) x 0x0100)	RW	SQ Producer index QPi (SQPIi)	This register is generated based on the parameter C_NUM_QP (where i = 1 to C_NUM_QP). This register is updated by the target application when it posts a new Send Work Queue entry. The SQPI can also be updated through the side band interface. The Send Queue is a circular buffer. The CQHEAD register for the corresponding QP provides information about the Work Queue entries that have been completed and can be over written by the target application. • [15:0]: Send Q Producer Index Doorbell • [31:16]: Reserved
0x023C + ((i-1) x 0x0100)	RW	Q depth QPi (QDEPTHi)	This register is generated based on the parameter C_NUM_QP (where i = 1 to C_NUM_QP). This register defines the queue depths for send, completion and receive queues. • [15:0]: Send Q depth (Send CQ will of the same depth) • [31:16]: Receive Q depth
0x0240 + ((i-1) x 0x0100)	RW	SEND Q PSN for QPi (SQPSNi)	This register is initiatialized at connection time by the SW. After that the HW updates it for every outgoing packet and should not be updated by the SW. This register is generated based on the parameter C_NUM_QP (where i = 2 to C_NUM_QP). This register does not exist for QP1. [23:0]: Send Q PSN
0x0244 + ((i-1) x 0x0100)	RO	Last RQ req for QPi (LSTRQREQi)	This register provides the last incoming RQ packet details. This register is generated based on the parameter C_NUM_QP (where i = 2 to C_NUM_QP). This register does not exist for QP1. • [23:0]: Rcv Q PSN • [31:24]: Rcv Q opcode
0x0248 + ((i-1) x 0x0100)	RW	Destination QP configuration for QPi (DESTQPCONFi)	This register is generated based on the parameter C_NUM_QP (where i = 1 to C_NUM_QP). This register is configured at connection time by the SW and provides the remote QPID connected to this QP. All outgoing packets from this QP are sent with this QPID as the destination QPID. [23:0]: Destination Connected QPID

Table 2-7: **ETRNIC Registers details *(Cont'd)***

Address	Access	Register Name	Details
0x0250 + ((i-1) x 0x0100)	RW	MAC destination address LSB QPi (MACDESADDLSBi)	This register is generated based on the parameter C_NUM_QP (where i = 1 to C_NUM_QP). This register is configured at connection time by the SW and provides the MAC address of the remote host connected to this QP. All outgoing packets from this QP are sent with this MAC address (LSB) as the destination MAC address. [31:0]: MAC destination address LSB
0x0254 + ((i-1) x 0x0100)	RW	MAC destination address MSB QPi (MACDESADDMSBi)	This register is generated based on the parameter C_NUM_QP (where i = 1 to C_NUM_QP). This register is configured at connection time by the SW and provides the MAC address of the remote host connected to this QP. All outgoing packets from this QP are sent with this MAC address (MSB) as the destination MAC address. [15:0]: MAC destination address MSB
0x0260 + ((i-1) x 0x0100)	RW	IP destination address 1 QPi (IPDESADDR1i)	This register is generated based on the parameter C_NUM_QP (where i = 1 to C_NUM_QP). If the remote host for this QP is configured using the IPv4 protocol, then this register indicates an IPv4 address. If the remote host is configured using the IPv6 protocol, this register then indicates theIPv6 destination address. [31:0]. [31:0]: IP Destination address LSB1
0x0264 + ((i-1) x 0x0100)	RW	IP destination address 2 QPi (IPDESADDR2i)	This register is generated based on the parameter C_NUM_QP (where i = 1 to C_NUM_QP). If the remote host is configured using the IPv6 protocol, this register indicates an IPv6 destination address [63:32]. If the remote host of this QP is configured using the IPv4 protocol, then this register is not used. [31:0]: IP Destination address LSB2
0x0268 + ((i-1) x 0x0100)	RW	IP destination address 3 QPi (IPDESADDR3i)	This register is generated based on the parameter C_NUM_QP (where i = 1 to C_NUM_QP). If the remote host is configured using the IPv6 protocol, this register indicates an IPv6 destination address [95:64]. IF the remote host of this QP is configured using the IPv4 protocol, then this register is not used. [31:0]: IP Destination address MSB1
0x026C + ((i-1) x 0x0100)	RW	IP destination address 4 QPi (IPDESADDR4i)	This register is generated based on the parameter C_NUM_QP (where i = 1 to C_NUM_QP). If the remote host is configured using the IPv6 protocol, this register indicates an IPv6 destination address [127:96]. If the remote host of this QP is configured using the IPv4 protocol, then this register is not used. [31:0]: IP Destination address MSB2

Table 2-7: **ETRNIC Registers details *(Cont'd)***

Address	Access	Register Name	Details
0x024C + ((i-1) x 0x0100)	RW	Timeout Configuration Register for QPi (TIMEOUTCONFi)	This register is generated based on the parameter C_NUM_QP (where i = 1 to C_NUM_QP). It provides the timeout configuration QPi. This register does not exist for QP1. • [5:0]: Timeout value • [7:6]: Reserved • [10:8]: Maximum retry count • [13:11]: Maximum RNR retry count • [15:14]: Reserved • [20:16]: RNR NACK Timeout value for outgoing packets • [31:21]: Reserved
0x0280 + ((i-1) x 0x0100)	RO	Status Sender sequence number QPi (STATSSNi)	This register is generated based on the parameter C_NUM_QP (where i = 2 to C_NUM_QP). This register does not exist for QP1. [23:0]: Current outgoing SSN (same as outgoing MSN)
0x0284 + ((i-1) x 0x0100)	RO	Status Message sequence number QPi (STATMSN)	This register is generated based on the parameter C_NUM_QP (where i = 2 to C_NUM_QP). This register does not exist for QP1. [23:0]: Current expected incoming MSN
0x0288 + ((i-1) x 0x0100)	RO	Status QPi (STATQPi)	This register provides the QP status for every QP. This register is generated based on the parameter C_NUM_QP (where i = 1 to C_NUM_QP). • [0]: QP in fatal status • [1]: Rcv Q ovfl (outgoing RNR NACK) • [2]: Send Q full • [3]: Outstanding Q full • [4]: CQ fifo full • [8:5]: Reserved • [9]: Send Q empty. This bit signifies that there are no SQEs left to process. However, it does not imply that all the SEND WQEs were acknowledged by the remote host. • [10]: Outstanding Q empty • [11]: QP packet was retired • [15:12]: Reserved • [22:16]: NACK syndrome received (Read/Write) • [23]: Reserved • [26:24]: Current retry count (Read/Write) • [27]: Reserved • [30:28]: Current RNR nack count for inc resp packets (Read/Write) • [31]: Reserved
0x028C + ((i-1) x 0x0100)	RO	Status Current SQ ptr under process (STATCURSQPTRi)	This register is generated based on the parameter C_NUM_QP (where i = 2 to C_NUM_QP). [15:0]: Current SQ pointer that is under process. This shows the number of WQEs that are outstanding and awaiting a response from the remote host.

Table 2-7: **ETRNIC Registers details *(Cont'd)***

Address	Access	Register Name	Details
0x0290 + ((i-1) x 0x0100)	RO	Status Response PSN for QPi (STATRESPSNi)	This register is generated based on the parameter C_NUM_QP (where i = 2 to C_NUM_QP). This register does not exist for QP1. [23:0]: Expexted response PSN
0x0294 + ((i-1) x 0x0100)	RO	Status RQ buffer current address for QPi (STATRQBUFCAi)	This register is generated based on the parameter C_NUM_QP (where i = 2 to C_NUM_QP). [31:8]: Receive Q Buffer current addr (256 B aligned)
0x0298 + ((i-1) x 0x0100)	RO	Status of WQEs posted to QPi (STATWQEi)	This register is generated based on the parameter C_NUM_QP (where i = 2 to C_NUM_QP). [15:0]: Count of WQEs pushed by QP MGR for this QP
0x029C + ((i-1) x 0x0100)	RO	Status RQ Producer index DB QPi (STATRQPIDBi)	This register is generated based on the parameter C_NUM_QP (where i = 2 to C_NUM_QP). [15:0]: Rcv Q Producer Index DB

Designing with the Core

This chapter includes guidelines and additional information to facilitate designing with the ETRNIC IP core.

General Design Guidelines

A typical ETRNIC subsystem would include one or more MAC hard/soft IPs. An AXI interconnect would also be required to connect the various AXI interfaces exposed by the ETRNIC IP. A DRAM may be a part of the solution which would require a DRAM controller to be instantiated as well.

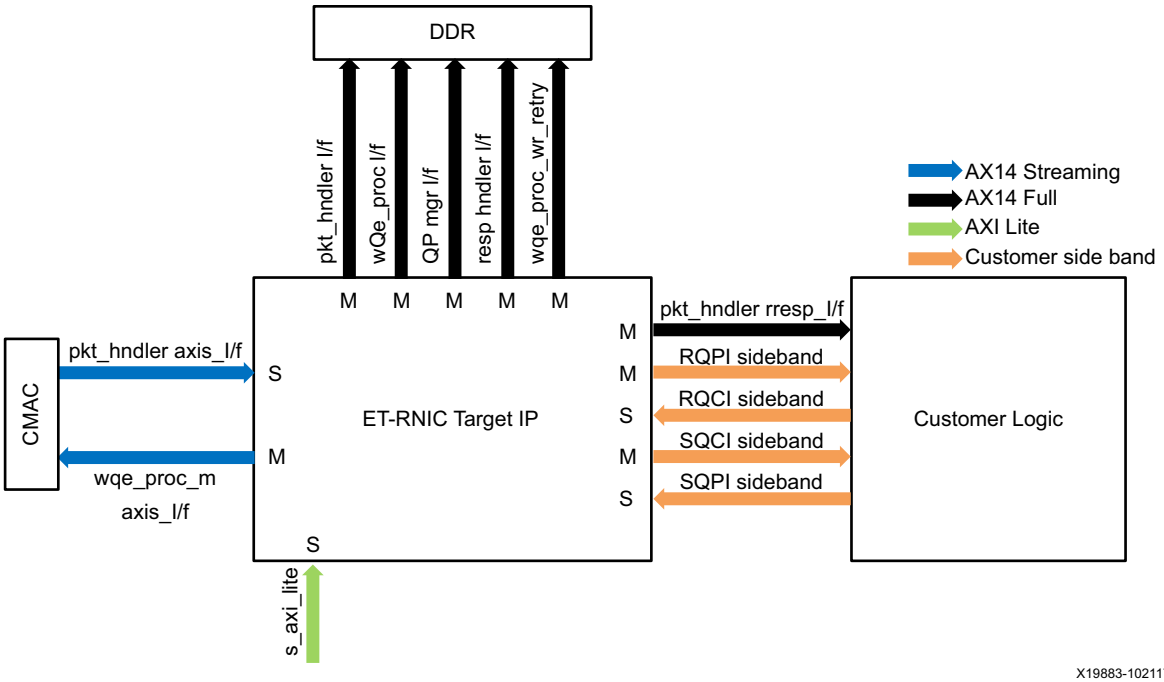

Figure 3-1: **ETRNIC Interfaces**

Figure 3-1 shows the various ETRNIC IP interfaces. The interfaces shown as interfacing with DDR may interface with any memory mapped region. Refer to Table 2-5 for details on each of these interfaces. The AXI4 full and the AXI4 streaming interfaces are 512 bits wide and are mainly used for data transfers. The ETRNIC IP provides sideband interfaces to allow for

efficient exchange of queue pair related doorbells. These side band interfaces can be enabled or disabled for each queue pair (QP) based on the configuration.

The ETRNIC IP has one AXI Lite slave interface to access the register space. The details of the memory map required for this slave interface is shown in the following table.

Table 3-1: **Address Space Allocation Requirement for Slave Interfaces**

Slave Interface	Size	Unit	Description
ETRNIC AXI Lite slave interface	64	KB	ETRNIC register interface

Apart from these, the IP also requires some memory regions to be allocated for some specific data structures. These memory regions may be mapped to a local DRAM or an AXI BRAM or any other memory mapped slave. Ensure that there is adequate bandwidth on these memory interfaces based on the line rate that ETRNIC should achieve.

Table 3-2: **ETRNIC IP Memory Requirement**

Memory Region	Size	Unit	Description
TX Header	16	KB	128 Headers of 128 bytes each
Dropped packet buffer	16	KB	64 packets of 256 bytes each
Bypass packet buffer	16	KB	32 packets of 512 Bytes each
Send Queue	1	MB	128 QPs of depth 128 locations and each SQE is of 64 bytes each
Receive Queue	16	MB	128 QPs of depth 128 locations and each SQE is of 1024 bytes each
Send completion Queue	16	KB	128 QPs of depth 128 locations and each CQE is of 4 bytes each
Write Retry buffers	8	MB	Buffers of 4K size each for 128 QPs and each QP with upto 16 outstanding transactions

Interrupts

The ETRNIC IP provides nine interrupt lines which are active-High. A brief description of these interrupts and their functionality is given below.

Table 3-3: **ETRNIC Interrupts**

Interrupts	Description
xrnic_intr0	Asserted HIGH when incoming IB packet fails Header validation rules.
xrnic_intr1	Asserted HIGH when MAD packet received from remote host.
xrnic_intr2	Reserved (Unused)
xrnic_intr3	Asserted HIGH whenever RNR-NAK acknowledge is sent to remote host

Table 3-3: **ETRNIC Interrupts *(Cont'd)***

Interrupts	Description
xrnic_intr4	Asserted HIGH when WQE completion is updated to RDMA. This interrupt can be enabled/disabled on a per QP basis
xrnic_intr5	Asserted HIGH when RDMA posts WQE with illegal opcode.
xrnic_intr6	Asserted HIGH when RDMA-SEND command received from a remote host. This interrupt can be enabled/disabled on a per QP basis
xrnic_intr7	Asserted HIGH when any QP moved to fatal state either due remote/ local error.
xrnic_intr8	Reserved (Unused)

These interrupts can be ORed at the system level to have a single interrupt line to the processor. Each of these interrupt lines can be enabled by writing a 1 to the corresponding bits of the Interrupt Enable INTEN register. On receiving an ORed interrupt the SW can read the INTSTS register to know the cause of the interrupt.

Interrupt lines 4 and 6 inform the drivers about a WQE completion or an incoming RDMA SEND respectively. These interrupts can be enabled or disabled on a per QP basis. Bit [2] of the QPCONFi registers allows selective enabling of Receive Queue interrupts per QP. Simlarly, bit [3] of the QPCONFi registers allows selective enabling of Send Completion Queue interrupts. In general QPs that require SW handling should have this option enabled. The QPs that are directly handled by the hardware will be informed through the hardware handshake ports and corresponding interrupt enable bits can be disabled. Such QPs should have the hardware handshake disable QPCONFi[4] bit reset to 0.

The RQINTSTSn and CQINTSTSn registers provide bitwise information about the QPs that have a pending RQ or CQ entry to be serviced.

These registers should be read by the SW on receiving an RQ or CQ interrupt respectively to know the QPs that are required to be serviced. These interrupt status should be cleared upon successful handling by writing a 1 to the respective bits.

Clocking

Two clocks are exposed at the top of the ETRNIC IP. These are: AXI4 clock and AXI lite clock. All the registers accesses work on the AXI lite clock while the rest of the logic works on AXI4 clock. Typically, the AXI4 clock would be of higher frequency (upto 200 MHz) while the AXI lite interface could be clocked at a lower frequency (divided AXI4 clock with the divided clock edges aligning with the AXI4 clock). However, these clocks are treated as synchronous inside the ETRNIC IP and are expected to be generated from the same clock source.

Resets

The ETRNIC IP requires two active-Low resets that are synchronized to the two clock domains respectively.

Design Flow Steps

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

- *Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator* (UG994) [Ref 1]
- *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 2]
- *Vivado Design Suite User Guide: Getting Started* (UG910) [Ref 3]
- *Vivado Design Suite User Guide: Logic Simulation* (UG900) [Ref 4]

Customizing and Generating the Core

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

If you are customizing and generating the core in the Vivado IP integrator, see the *Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator* (UG994) [Ref 1] for detailed information. IP integrator might auto-compute certain configuration values when validating or generating the design. To check whether the values change, see the description of the parameter in this chapter. To view the parameter value, run the `validate_bd_design` command in the Tcl console.

You can customize the IP for use in your design by specifying values for the various parameters associated with the IP core using the following steps:

1. Select the IP from the Vivado IP catalog.
2. Double-click the selected IP or select the **Customize IP** command from the toolbar or right-click menu.

For details, see the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 2] and the *Vivado Design Suite User Guide: Getting Started* (UG910) [Ref 3].

Note: Figures in this chapter are illustrations of the Vivado Integrated Design Environment (IDE). The layout depicted here might vary from the current version.

Output Generation

For details, see the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 2].

Simulation

For comprehensive information about Vivado simulation components, as well as information about using supported third-party tools, see the *Vivado Design Suite User Guide: Logic Simulation* (UG900) [Ref 4].

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

Synthesis and Implementation

For details about synthesis and implementation, see the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 2].

Example Design

This chapter contains information about the ETRNIC core example design. The example design consists of the following modules:

- Clock and Reset Generator (simulation only)
- Register Configuration Module
- Send, Read Response, and ACK Generator
- WQE generator
- TX Path Checker
- RX Path Checker

Figure 5-1 shows the top level example design architecture.

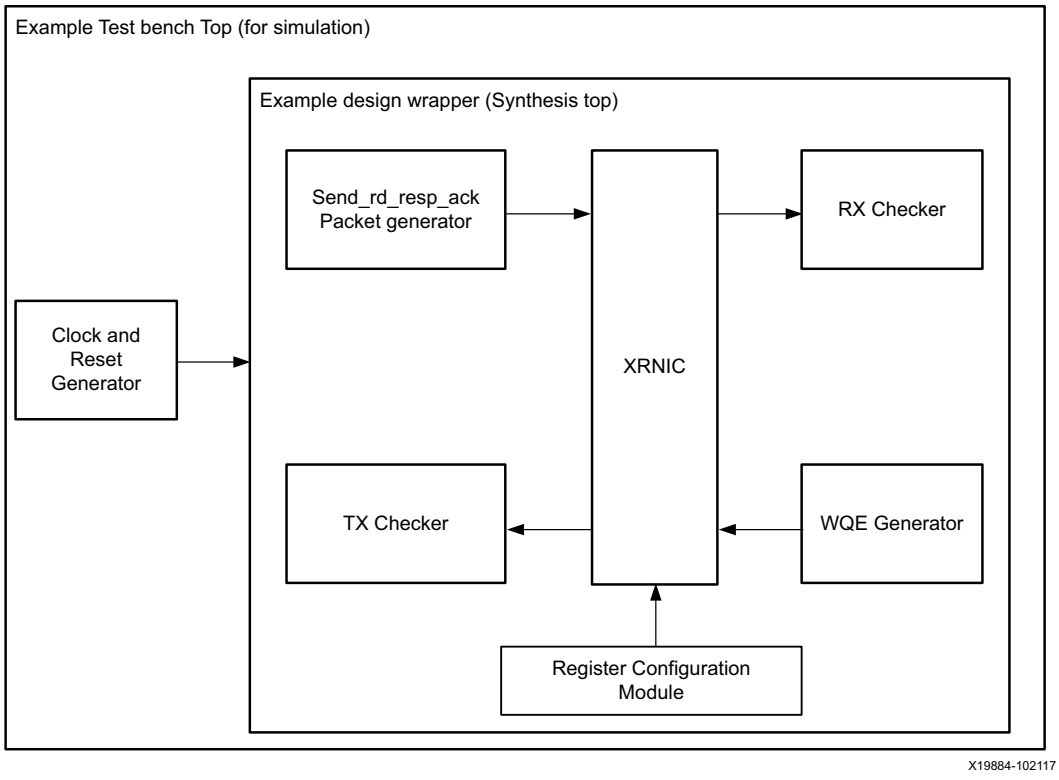

Figure 5-1: **Example Design Architecture**

Apart from the ETRNIC IP, the example design integrates the following modules:

- **Register Configuration Module**: This module of the example design configures all the required registers of the ETRNIC IP
- **Packet Generator Module**: This module of the example design generates the following types of packets
  a. SEND Packets
  b. RDMA Read Response Packets for all the Read Requests from the ETRNIC IP
  c. ACK Packets for all the RDMA Write requests from the ETRNIC IP
- **RX Checker Module**: This module of the example design checks the capsules (from SEND packets), and payload (from Read Response) received from the ETRNIC IP. This module also checks the number of door bells rang on the RQ side band interface. The payload size of Read Response packets, RDMA write requests is 256 bytes. Capsule size in the SEND packets is 80 bytes.
- **WQE Generator Module**: This module of the example design, is responsible for generating the work queue requests and SQ PI doorbell updates to the ETRNIC IP
- **TX Checker Module**: This module of the example design checks the data transmitted by the ETRNIC IP over the streaming interface.

Example Design Features

The example design exercises the following features of the ETRNIC IP:

- The following incoming packets are sent to the ETRNIC IP:
  a. RDMA SEND
  b. RDMA Read Response
  c. ACK Packets
- The following outgoing packets are checked by the example design:
  a. RDMA Read Requests
  b. RDMA Write Requests
- Hardware handshake paths of ETRNIC:
- Only IP4 Packets are exercised
- Only 6 QPs are exercised (QP2 to QP7)

Example Design Limitations

The ETRNIC example design has the following limitations:

- Example design does not exercise Retry path
- Example design exercises only Hardware handshake path (QPCONFi[4] is set to 0 for all QPs)
- Example design does not exercise IPV6 packets
- Example design does not generate any connection management (MAD) packets to QP1
- Example design limits the number of each supported packet type transactions to 8

Simulating the Example Design

For more information on Simulation, see the Vivado Design Suite User Guide: Logic Simulation (UG900) [Ref 4].

Simulation Results

The simulation script compiles the ETRNIC example design and supporting simulation files. It then runs the simulation and checks to ensure that it completed successfully. If the test passes, then the following message is displayed:

```
Test Completed Successfully
```

If the test fails, then the following message is displayed:

```
ERROR: Test Failed
```

If the test hangs, then the following message is displayed:

```
ERROR: Test did not complete (timed-out)
```

Example Sequence

The demonstration test bench performs the following tasks:

- All the required ETRNIC registers are configured by the Register Configuration Module through AXI lite interface.
- There are three operations being handled in example test bench:
  a. RDMA SEND:
     - Packet Generator module generates eight SEND packets to the QP2 to QP7
     - RX Path Checker of example design, checks for the data integrity on the capsule transferred from the ETRNIC along with the door bells rang
  b. RDMA RD REQUEST:
     - Example design has a predefined RDMA Read work Queue entry Packets, and the WQE Generator module of example design rings the SQPI doorbell, and sends the work queue entry packets when requested by the ETRNIC
     - The TX Checker Module of example design, checks the necessary fields of the RDMA read request received
     - After successful validation of the request, the Packet Generator module sends RDMA Read Response to the ETRNIC IP
     - The RX Path Checker of example design, checks for the data integrity on the payload transferred by the ETRNIC along with the door bells rang
  c. RDMA WRITE REQUEST:
     - Example design has a predefined RDMA Write work Queue entry Packets, and WQE Generator module of example design rings the SQPI doorbell, and sends the work queue entry packets when requested by the ETRNIC
     - The TX Checker Module of example design checks the necessary fields of the RDMA write request received
     - Upon successful validation of the request, the Packet Generator module sends ACK packet to the ETRNIC IP

Debugging

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

TIP: *If the IP generation halts with an error, there might be a license issue. See License Checkers in Chapter 1 for more details.*

Debugging the IP/System

The ETRNIC IP has a number of status registers implemented to allow for extensive debug in case of a failure. These registers along with counter information from the initiator RNIC, for example Mellanox, can provide vital clues to debug any failure. The debug tools available to the system debugger working with ETRNIC IP are:

- ETRNIC complete register dump
- Promiscuous or packet sniffer mode counters (from the initiator RNIC),
- HW counters (from the initiator RNIC).

The steps to dump the different logs are explained in the following sections.

Complete Dump of the ETRNIC Registers

If you are using smart cable, perform these steps:

1. Change the connect command directive with appropriate IP address of the Smart cable using the following command:

   ```
   $ connect -host 172.23.29.242
   ```

2. Change the target to 9 using the following command:

   ```
   :  $ ta 9
   ```

3. Connect to the `/proj/xbuilds and /wrk/released` folder in a Linux machine that is in the same LAN as the smart cable.
4. Execute the TCL script using the command:

```
$ /proj/xbuilds/2016.4_INT_daily_latest/installs/lin64/Vivado/HEAD/bin/xsdb
<PATH_to_tcl>/regDump.tcl | tee  <output_file_name>
```

Enable promisc mode on the initioator RNIC

1. Enable promisc mode on the initiator RNIC using the command:

```
$ ifconfig <interface name> promisc
$ netstat -i
```

2. At the end of the test, dump all the counter information from the initiator RNIC using the command:

```
$ ethtool –S <RNIC Card name> > ethtool.log
```

HW Debug Counters for Mellanox RNIC

To access and dump hardware debug counters for Mellanox RNIC, use the command:

```
$ cd /sys/class/infiniband/mlx5_0/ports/1/hw_counters/
$ for file in `ls .`; do echo -n "${file}:"; cat $file; done
```

Note: For more information on hardware debug counters for Mellanox, see DOC-2572 on the Mellanox community.

Some quick debug checks that you can do to ensure that the system is clean are listed here.

- Check the ethtool.log file for any link failures or CRC failures. Any non-zero value in these two counters points towards an unstable link and can be the cause of failure. Two snippets from the `ethtool.log` file are listed here.

  Sample 1:

```
rx_wqe_err: 0
rx_mpwqe_filler: 0
rx_mpwqe_frag: 0
rx_buff_alloc_err: 0
rx_cqe_compress_blks: 0
rx_cqe_compress_pkts: 0
link_down_events_phy: 4
rx_out_of_buffer: 0
rx_vport_unicast_packets: 5
rx_vport_unicast_bytes: 422
tx_vport_unicast_packets: 10
tx_vport_unicast_bytes: 714
rx_vport_multicast_packets: 46
rx_vport_multicast_bytes: 7608
```

  Sample 2:

```
rx_vport_rdma_multicast_bytes: 0
tx_vport_rdma_multicast_packets: 0
tx_vport_rdma_multicast_bytes: 0
tx_packets_phy: 320587336
rx_packets_phy: 324924215
```

```
rx_crc_errors_phy: 0
tx_bytes_phy: 27657272230
rx_bytes_phy: 467673988652
tx_multicast_phy: 59
tx_broadcast_phy: 32
rx_multicast_phy: 0
rx_broadcast_phy: 9
rx_in_range_len_errors_phy: 0
```

- Check the hw_counters on the initiator side. These counters give a picture of all fatal/ non-fatal errors seen by the initiator. A sample of the counters:

```
duplicate_request:0
implied_nak_seq_err:0
lifespan:10
local_ack_timeout_err:0
out_of_buffer:0
out_of_sequence:0
packet_seq_err:0
rnr_nak_retry_err:0
rx_atomic_requests:0
rx_read_requests:5
rx_write_requests:108307999
```

- Check the following register locations from the ETRNIC register dump. The QP Status (STATQPi) registers for all enabled QPs provide a status of the different QPs. Check if the QP FATAL status is set to 1 in any of the QP status registers. For example, the QP Status register for QP 5:

```
84000688:   30620601    • QP Fatal is set to 1
```

- If the QP is in FATAL state, no transactions are performed from this QP and the QP gets disconnected. Bits [22:16] in the same register provide the last AETH syndrome received from the initiator. In many cases the QP might go into FATAL state due to a NAK syndrome received from the initiator. The NAK syndrome helps you understand the failure being seen by the initiator RNIC card. In the above example, the AETH syndrome of 0x62 indicates a “Remote Access Error” from the initiator. The decoding of the AETH syndrome is provided in the InfinibandTM Architecture Specification Volume 1 (Release 1.2.1). For NAK code details in this specification, see Table 43: AETH Syndrome field and Table 44: NAK Codes.
- Check the Incoming and outgoing NAK count registers ((INNACKPKTCNT) and (OUTNACKPKTCNT)) at offset 0x134 and 0x138 for the number of incoming NAK syndromes seen and number of NAK syndromes sent out. This number should normally correlate with the number seen from the HW_counters seen at the initiator. In general not all NAK codes are fatal. However, all NAK codes lead to retries and can lower the overall performance of the system. A high number of NAK codes can be a cause of concern.
- The total number of retries initiated by the target can be known from the Retry count status register (RETRYCNTSTS) at offset 0x140. Normally this number will match with the incoming NAK count. In case this number is more than the incoming NAK count value, it may be due to timeouts. Timeouts happen when the responder (in this case, the initiator RNIC) does not respond to a request in a given time. The timeout value is

configured in the Timeout Configuration register (TIMEOUTCONF). This timeout interval is implemented as per the InfinibandTM Architecture Specification Volume 1 (Release 1.2.1) clause C9-141. It may be worthwhile to try and increase the timeout interval and check if the number of retries is reduced.

- ETRNIC register offset 0x6C (ERRBUFWPTR) indicates Error buffer write pointer. This register gives the number of error packets received. Each error packet will be stored in the address location provided in Error buffer base address (ERRBUFBA) register (offset 0x60). Each entry in this buffer will be given with error syndrome. See ETRNIC RX Path for details. The rows highlighted in yellow enlist the conditions that will cause the QP to go into FATAL state.

Appendix B

Additional Resources and Legal Notices

Xilinx Resources

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

For a glossary of technical terms used in Xilinx documentation, see the Xilinx Glossary.

Documentation Navigator and Design Hubs

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

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

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

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

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

References

These documents provide supplemental material useful with this product guide:

1. *Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator* (UG994)

2. *Vivado Design Suite User Guide: Designing with IP* (UG896)
3. *Vivado Design Suite User Guide: Getting Started* (UG910)
4. *Vivado Design Suite User Guide: Logic Simulation* (UG900)
5. *Vivado Design Suite User Guide: Programming and Debugging* (UG908)
6. *Vivado Design Suite User Guide: Implementation* (UG904)
7. *Vivado Design Suite User Guide:* AXI Reference Guide (UG1037)

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
06/06/2018	1.1	Support added for Explicit Congestion Notification (ECN).
04/04/2018	1.0	Initial Xilinx release.

Please Read: Important Legal Notices