



# Extending the Spartan-6 FPGA Connectivity TRD (PCIe-DMA-DDR3-GbE) to Support the Aurora 8B/10B Serial Protocol

Authors: Vasu Devunuri and Sunita Jain

## Summary

Targeted Reference Designs (TRDs) provide Xilinx designers with turn-key platforms to create FPGA based solutions in a wide variety of industries. This application note extends the Spartan-6 FPGA PCIe-DMA-DDR3-GbE TRD to support Aurora 8B/10B serial protocol. [Ref 1]

The PCIe-DMA base platform moves data between system memory and the FPGA. The data thus transferred, can be consumed within the FPGA, or forwarded to another FPGA, or sent over the backplane using a serial connectivity protocol such as the Aurora 8B/10B serial protocol. Similarly, data can be brought in to the FPGA from another FPGA or backplane through the Aurora protocol and sent to system memory for further processing or analysis. This enhancement retains the Ethernet operation *as is* and demonstrates PCIe-to-Ethernet and PCIe-to-Aurora bridging functionality.

## Connectivity TRD

The Spartan-6 FPGA Connectivity TRD, shown in [Figure 1](#), demonstrates the key integrated components in a Spartan-6 FPGA: the Endpoint block for PCI Express, the GTP transceivers, and the memory controller working together in an application along with additional IP cores including the third-party (Northwest Logic) Scatter Gather Direct Memory Access (DMA) engine, Xilinx Platform Studio LocalLink Tri-mode Ethernet MAC (XPS-LL-TEMAC), and the Xilinx Memory Interface Generator (MIG).

This TRD is a x1 Endpoint block for PCI Express (v1.1 compliant) showcasing the following independent applications:

- Network interface card (referred to as the network path) providing either:
  - GMII mode using external Ethernet PHY—typically used to connect to copper networks.
  - 1000BASE-X mode using FPGA GTP transceivers—typically used to connect to optical fiber Ethernet networks

The network path allows connection to an external network and running networking applications such as browsing web pages, Telnet, and FTP.

- External memory interface over PCI Express (referred to as the memory path)

The memory path showcases data movement between system memory and DDR3 SDRAM through the Spartan-6 FPGA.

The TRD uses a bus-mastering scatter-gather Direct Memory Access (DMA) engine to off-load processor data-transfer overhead. The DMA works in conjunction with the PCI Express Endpoint and enables data movement between system memory and the FPGA at high speed.

For details on the Spartan-6 FPGA PCIe-DMA-DDR3-GbE TRD, refer to UG392, *Spartan-6 FPGA Connectivity Targeted Reference Design User Guide*. [Ref 1]

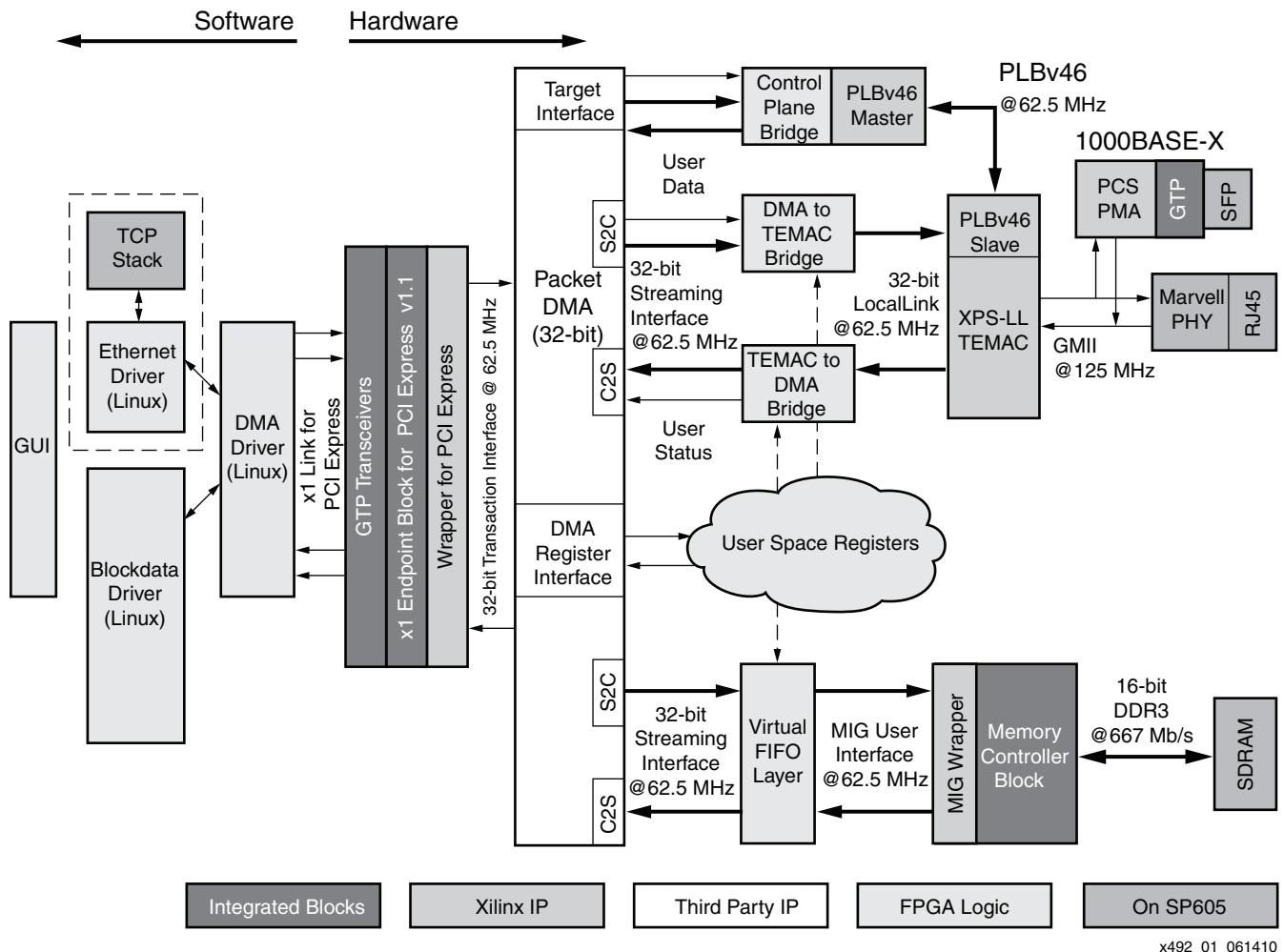


Figure 1: Spartan-6 FPGA Connectivity TRD

The entire TRD framework is built in a layered manner. PCIe and DMA blocks form the foundation of the entire platform. The network path and memory path are two applications developed around this foundation.

The TRD can be used as *is* or by using the guidelines provided in this application note to develop end-design versions using other serial protocols, video, or LVDS.

## Introduction

This application note builds on the Spartan-6 FPGA PCIe-DMA-DDR3-GbE TRD by extending it to connect to the Xilinx proprietary Aurora 8B/10B serial protocol. Aurora is a scalable, lightweight, link-layer protocol that is used to move data across point-to-point serial links. It provides a transparent interface to the physical serial links and supports both framing and streaming modes of operation. This application note uses Aurora in framing mode.

The network path functioning as the network interface card remains the same. The memory path is modified to support packet FIFO over the Spartan-6 FPGA memory controller and integrating the Aurora 8B/10B LogiCORE IP, which operates through the packet FIFO.

This application note also provides directions on making modifications and incorporating Aurora 8B/10B IP in the Spartan-6 FPGA Connectivity TRD to arrive at the design shown in the block diagram in [Figure 2](#).

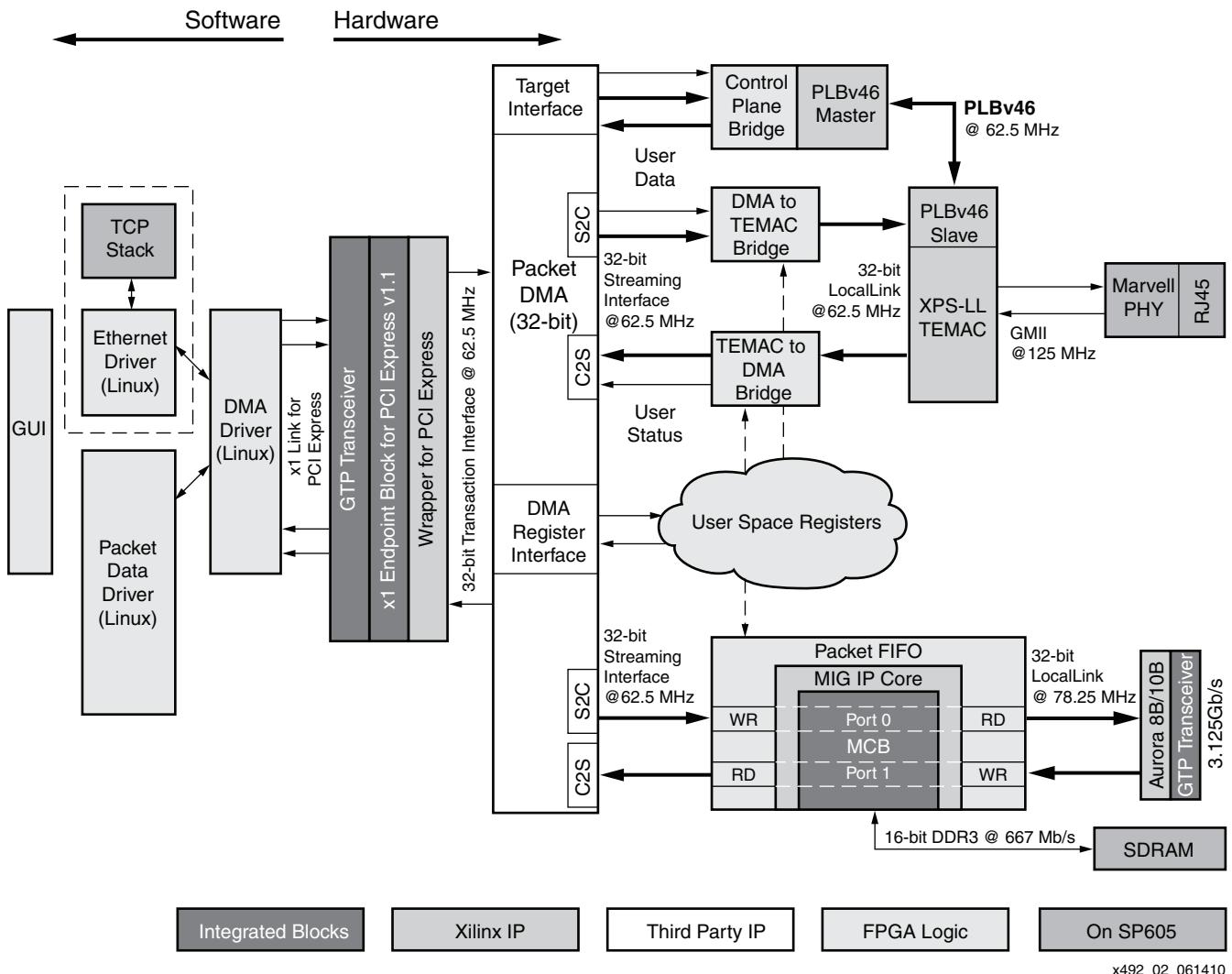


Figure 2: Spartan-6 FPGA Connectivity TRD Extension with Packet FIFO and Aurora IP

## Reference Design Features

The main feature changes in this reference design from the Spartan-6 FPGA Connectivity TRD are:

- Conversion of the DDR3 SDRAM to a LocalLink-based Packet FIFO
  - The design utilizes two fully bidirectional, 32-bit wide memory controller block (MCB) ports for data movement from:
    - DMA from PC system to Aurora Transmit (egress path)
    - Aurora receive to PC system through DMA (ingress path)
- An extensible Packet FIFO design that uses four fully bidirectional Spartan-6 FPGA MCB ports
- Integration of Aurora 8B/10B LogiCORE IP

**Note:** The 1000BASE-X mode of operation is not supported in this design.

## Requirements

- Spartan-6 FPGA Connectivity Kit
  - SP605 board with a Spartan-6 device (XC6SLX45T-3FGG484C)
  - Fedora 10 Linux OS LiveCD
- ISE® Design Suite 12.1 (Embedded or System Edition)
- Modelsim 6.5c or later
- PCI Express capable computer (PC)
- FMC Card (HW-FMC-XM104-G, if available)
- SMA cables

**Note:** The FMC card is not shipped as part of the Spartan-6 FPGA Connectivity Kit.

## Hardware

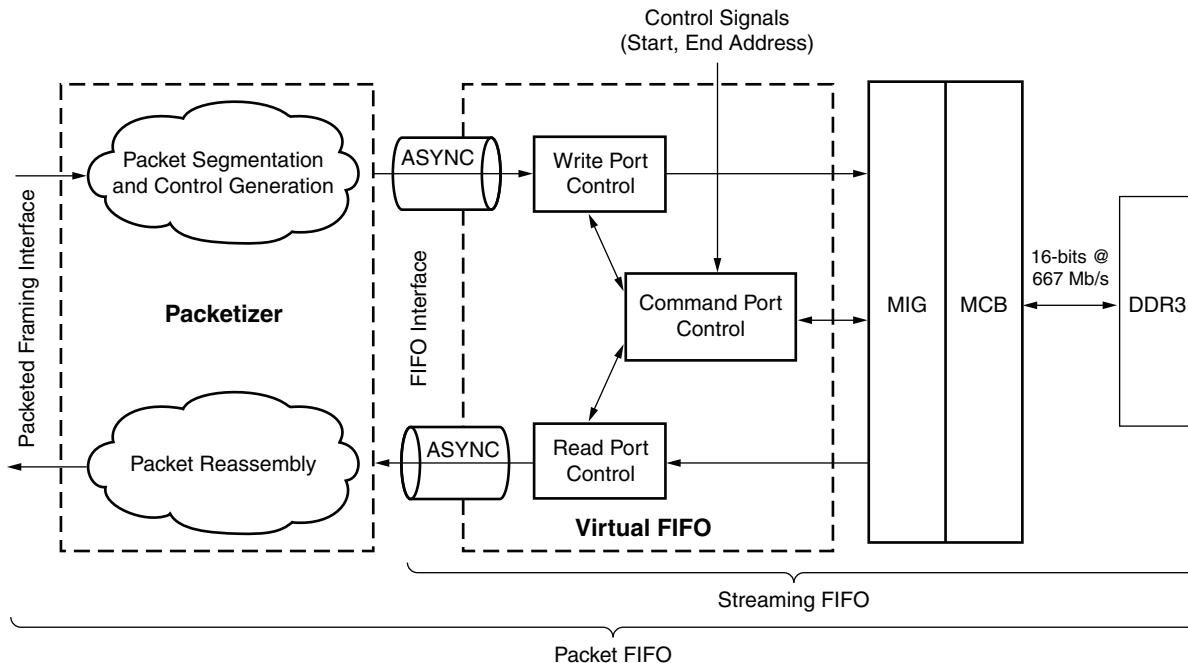
This section lists the hardware design details for building a packet FIFO over the virtual FIFO provided in the Spartan-6 FPGA Connectivity TRD and additional flow control logic for the Aurora 8B/10B IP.

### Multi-port Packet FIFO

The Spartan-6 FPGA Connectivity TRD provides a virtual FIFO which supports one memory controller port (fully bidirectional, 32-bit interface). The virtual FIFO is used in streaming mode i.e., it has no knowledge of packet context or packet boundary delimiters; it only understands blocks of data.

As suggested in the Spartan-6 FPGA Connectivity TRD User Guide [Ref 1], Chapter 5, multiple instances of the streaming FIFO can be used along with multiple memory controller ports to build a multi-port streaming FIFO.

This application note adds the functionality to make the streaming FIFO function as a packet FIFO. The block diagram of the packet FIFO design is shown in [Figure 3](#).



*Figure 3: Packet FIFO Block Diagram*

A packetizer module is designed to embed packet context information into the data stream and interface to existing streaming FIFO. A packet FIFO is created using a packetizer, a virtual FIFO, and the MIG generated memory controller wrapper.

[Figure 2](#) shows a single packet FIFO block interfacing to one MCB port. For multiple MCB ports (controlled by the NUM\_PORTS parameter), multiple instances of the packet FIFO block (connected to independent MCB ports) are generated. Asynchronous FIFOs are used to manage clock domain crossing between the user design and virtual FIFO logic.

### Packetizer

Based on a simple store and forward scheme, the packetizer block embeds the packet context information into the streaming data. The packetizer inserts a control word containing packet context information periodically in the data stream. The periodicity of control word insertion is defined by the BLOCK\_SIZE parameter or end of packet, whichever occurs first.

To understand the packetization scheme, consider the following example which uses the Xilinx LocalLink as user interface.

Since DDR3 memory can not store the framing sidebands (SOP, EOP, REM, etc.) separately, the packetizer segments the frame and introduces control words at block boundaries (as programmed by the BLOCK\_SIZE parameter). The control word provides information on length and various LocalLink sidebands.

The control word format is described in [Table 1](#).

**Table 1: Control Word Format**

Bit Location	Field	Description
0	SOP Status	Start of packet status
1	EOP Status	End of packet status
3:2	Remainder Status (REM)	Remainder value stored in the control word (uses a REM width of two bits)
15:3	Reserved	Unused, currently contain zeros
31:16	Length	Length in double word (DW) of data payload following control word (1 DW = 4 bytes)

Inserting a control word consumes one location in DDR3 SDRAM as overhead; hence, the packet FIFO mode of operation provides slightly lesser throughput than using the streaming FIFO mode of operation.

### Packet FIFO Parameters

The [Table 2](#) defines the design parameters available for packet FIFO and the default values.

**Table 2: Design Parameters**

Parameter	Default Value	Description
NUM_PORTS	2	Defines the number of memory controller ports. The maximum value supported is four since the memory controller supports four fully bidirectional ports.
BLOCK_SIZE	64	Data block size (in double words) for the packetizer block that decides the packet segmentation boundaries. Allowed values are 64, 128, and 256.

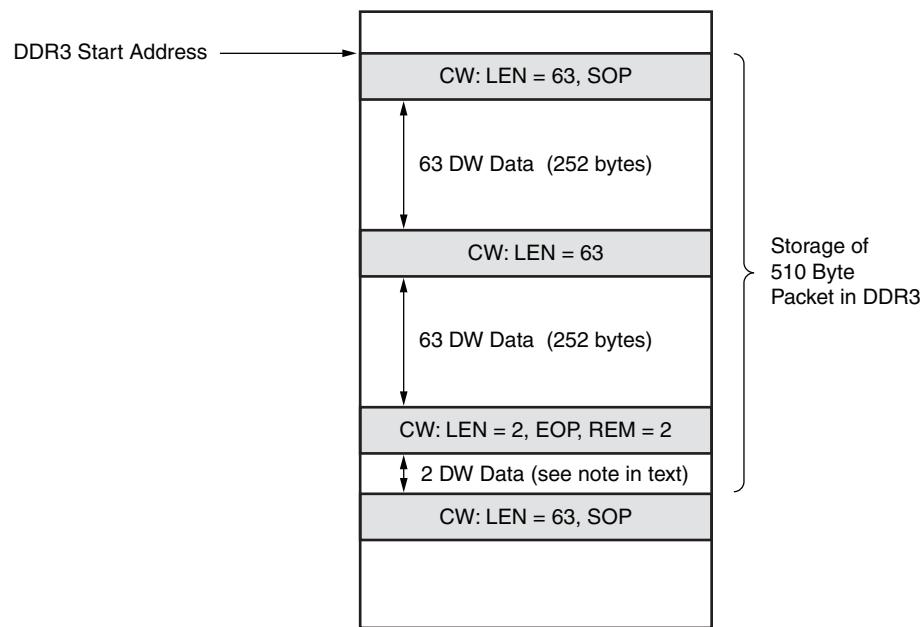
**Note:** The design with packet FIFO wrapper for all four MCB ports and Ethernet together does not fit on a Spartan-6 LX45T device; however, this is a feasible design in a larger Spartan-6 device.

By default, the MIG core is generated with four fully bidirectional 32-bit data width ports using a round-robin arbitration scheme. When required, the arbitration scheme can be customized during MIG IP core generation.

Packet segmentation and control word insertion boundaries are controlled by the BLOCK\_SIZE parameter. If packets are greater than BLOCK\_SIZE, packets are segmented into blocks equal to BLOCK\_SIZE-1 and a control word is inserted at the start of the data block. On reads, the control words are stripped from the stream of data and corresponding packet framing signals are built and passed over to the user interface (LocalLink in the current example). When a larger BLOCK\_SIZE is used, reading one control word provides more data payload, while increasing the storage requirement for segmentation while writing. The current design provides a 512 locations deep segmentation buffer. To increase BLOCK\_SIZE beyond allowed values, this segmentation buffer depth must be increased however, it also increases overall block RAM utilization.

To understand the segmentation scheme, consider an example of a 510-byte packet size with BLOCK\_SIZE = 64. Figure 4 shows the way this packet is stored in DDR3 SDRAM (CW refers to control word). For a BLOCK\_SIZE of 64, a 510-byte packet is split in three blocks. The first two blocks carry one DW of control information followed by 63 DW of data. The last block carries one DW of control information and two DW of data.

**Note:** The 2 DW data packet is 8 bytes stored where a total of 6 bytes are valid. Only 2 bytes are valid of the last 4 bytes, as indicated by REM.



x492\_04\_061710

Figure 4: **Packet Storage in DDR3**

Only the module interfacing to the framing user interface in the packetizer is aware of the interface details (LocalLink in the current scenario). When there is control word misalignment, the packetizer error bit in the [Packet Error Register \(0x9300\), page 16](#) is set and can be polled by software. No misalignment is expected unless there is data corruption or misalignment in external memory. The reference design provided with the application note currently has no way to recover from a control word misalignment scenario, apart from requiring a system reset. Designers can add an application specific scheme for verifying the packet FIFO data integrity.

## Aurora Protocol Considerations

The Aurora protocol is a scalable, lightweight, link-layer protocol that is used to move data across point-to-point serial links.

The reference design uses a 1-lane framing, 4-byte user interface width, Aurora 8B/10B link running at 3.125 Gb/s and using one transceiver accessible through the FPGA Mezzanine Card (FMC). The 125 MHz differential clock available on SP605 board is used as a reference clock for the transceivers.

However, in absence of an FMC connection, the design can be tested by enabling the near-end PMA loopback mode in the transceivers.

### Native Flow Control

Since the Aurora 8B/10B IP does not provide destination ready control in the receive direction, the native flow control (NFC) interface is used which avoids packet loss by FIFO overflow. NFC in Aurora regulates the data transmission rate at the receiving end of a full-duplex channel. By specifying the number of idle dead beats that must be placed into the data stream, NFC allows the receivers to control the rate of the data sent to them. The data flow can also be turned off completely by requesting that the transmitter temporarily send only idles.[\[Ref 4\]](#)

The Aurora 8B/10B IP configuration used in this design operates at a user clock of 78.125 MHz. As per the Aurora protocol specification, the round trip delay through the Aurora interfaces between the NFC request and the first pause arriving at the originating channel partner must not exceed 256 symbol times.

For example:

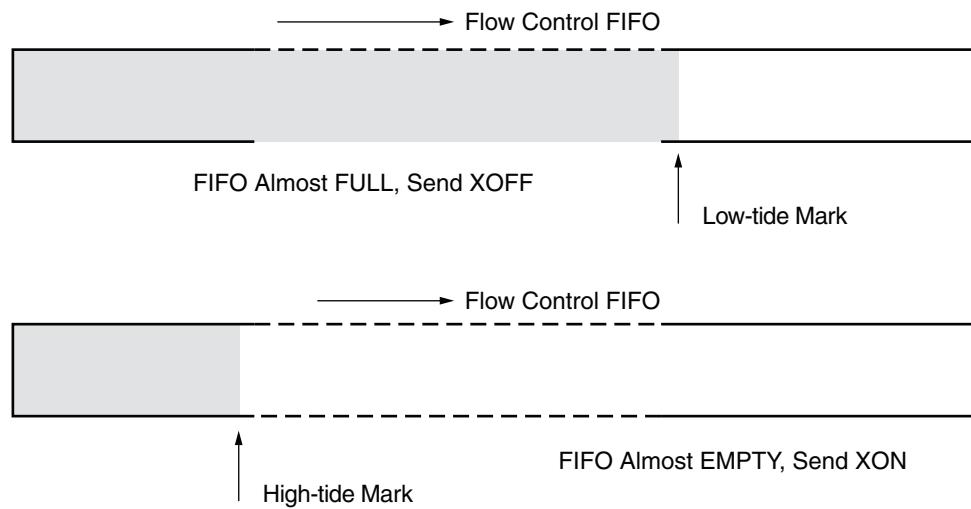
With a 3.125Gb/s rate, 1 symbol =  $10 \times 640 \text{ ps} = 6.4 \text{ ns}$ .

Using 256 symbol times:  $256 \times 6.4 = 1638.4 \text{ ns}$ .

Using a 78.125 MHz clock (12.8 ns period), the worst case delay is 128 clock cycles.

This means that the NFC should be asserted when only 128 locations are available in the receive FIFO.

A 512 locations deep-flow control FIFO is used on the receive interface of the Aurora 8B/10B IP to control the NFC as shown in [Figure 5](#).



*Figure 5: Flow Control FIFO and NFC*

Avoiding packet loss is accomplished by considering a low-tide and a high-tide mark for the flow control FIFO as almost full and almost empty FIFO states, respectively. When the write space

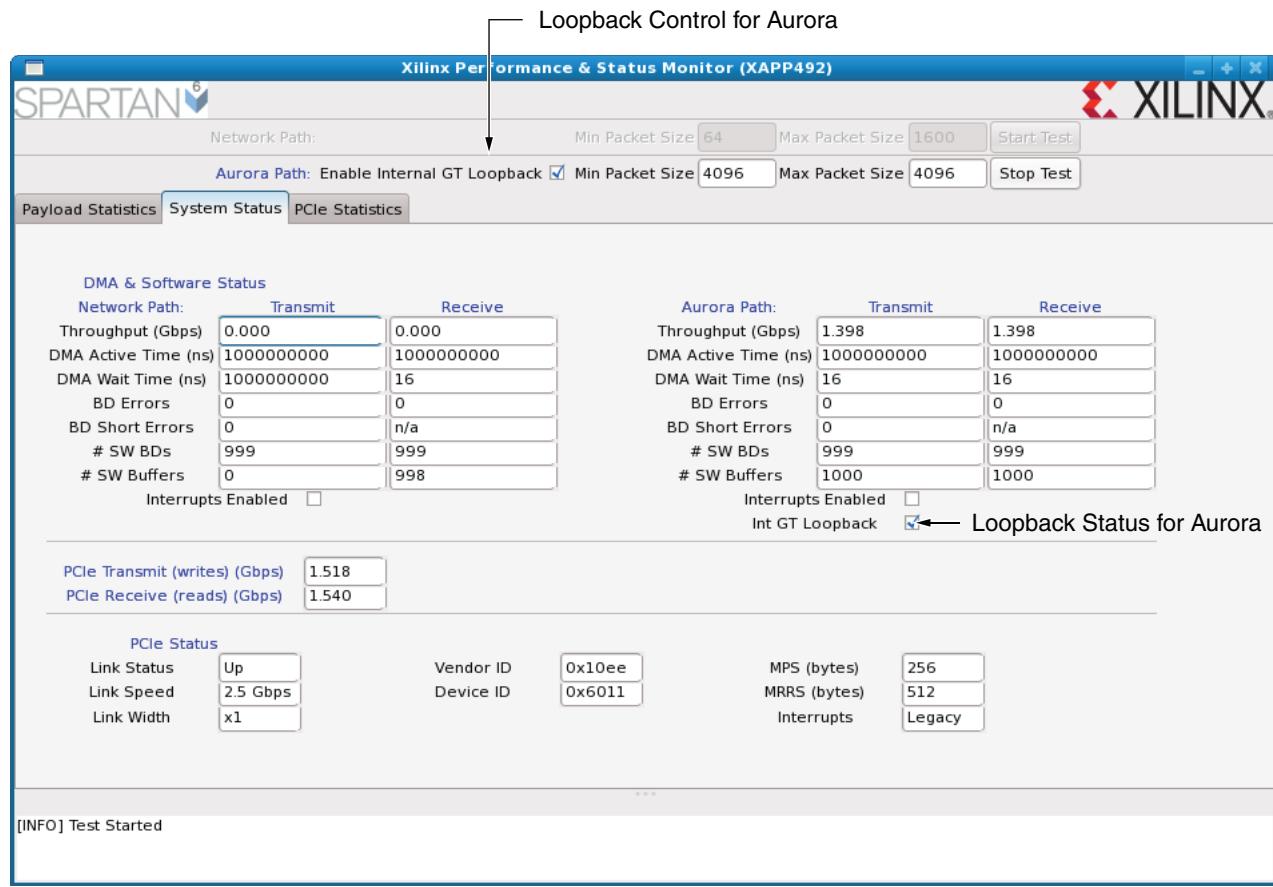
available status of the FIFO reaches the low-tide mark, an XOFF NFC request is sent to the channel partner, which stops any further data transmission from the remote link partner. Once the write space available status of the FIFO reaches the high-tide mark, an XON NFC request is sent, which enables data transmission from the remote partner. Packet loss is avoided by an effective flow-control mechanism. The high-tide and low-tide threshold values are provided as control registers. These values can be programmed through the software driver.

## Software

The software driver delivered with the Spartan-6 FPGA Connectivity targeted reference design requires minor changes on the memory path to include registers defined for the Aurora status and control logic as defined in [Aurora Status and Control Registers](#).

The driver is modified to check for Aurora status before initiating traffic on the Aurora path. In the case where there is no *channel up* or *lane up* from the Aurora link, traffic is not initiated from software and the GUI displays a warning message.

The GUI ([Figure 6](#)) is modified to include an option to enable near-end PMA loopback mode on the transceiver for Aurora operation in the absence of an FMC. This control from the GUI, when selected, programs the transceivers used by the Aurora 8B/10B IP to operate in near-end PMA loopback mode by changing the value of the LOOPBACK input signal.



*Figure 6: GUI*

The procedure to load/unload the software driver and test the design in hardware remains the same as explained in UG392 [\[Ref 1\]](#) or UG665 [\[Ref 2\]](#). To include the Aurora 8B/10B specific changes in software, a macro named AURORA is defined.

## Summary of Steps to Follow

This section provides a summary of the steps to reuse the existing Spartan-6 FPGA Connectivity TRD and extend it to support the Aurora protocol. Though the final reference design is provided as part of this application note, these steps are intended to provide a quick summary on the procedures followed as guidance for modifying the existing TRD.

From a software driver modification perspective, the incorporation of the Aurora 8B/10B IP on the memory path is straightforward since the block data driver can be modified to incorporate additional registers.

**Note:** The Core Generator™ files required to regenerate the IP cores, the design source code, simulation test-bench and implementation scripts are provided with the reference design.

### IP Generation

- Generate the Aurora 8B/10B IP with the following specifications:
  - 1 lane, 4-byte framing interface
  - 3.125 Gb/s line rate using a 125 MHz reference clock with immediate NFC and using the X1Y0 transceiver in Tile 1.
- Generate the Spartan-6 FPGA memory controller using the MIG tool with the following specifications:
  - Four fully bidirectional ports supporting DDR3 SDRAM with round-robin arbitration scheme.

The Aurora 8B/10B IP is provided with the [Reference Design](#) file and the MIG IP is generated when running the scripts provided. A golden set of `xco` files to regenerate the IP cores are also provided under the `design/reference/xco_files` folder.

### Design

- Design a packetizer block over the virtual FIFO as detailed in [Multi-port Packet FIFO](#).
- Design the native flow control logic for Aurora 8B/10B IP as explained in [Native Flow Control](#).
- Add additional control registers as required. A brief description of registers is provided in [Appendix A: Register Description](#).

### Integration

- Integrate the packet FIFO and Aurora 8B/10B IP at the top-level design.
- Modify the top level of the test bench to include newer ports and add an Aurora serial interface loopback
- Modify simulation scripts to include newer sources and simulate

### Software Driver Modification

- Modify the block data driver (`driver/xblockdata/user.c`) to include control register programming for the Aurora path and a checkpoint to check for Aurora link status before traffic initiation.
- Modify the GUI to include a transceiver loopback option for the Aurora path.

### Design Implementation

Update the implementation script and UCF by including newer sources and constraints. The script to support the Project Navigator flow is provided with the design. The golden reference design `bit` and `mcs` files are also provided in the `design/reference/configuration` folder.

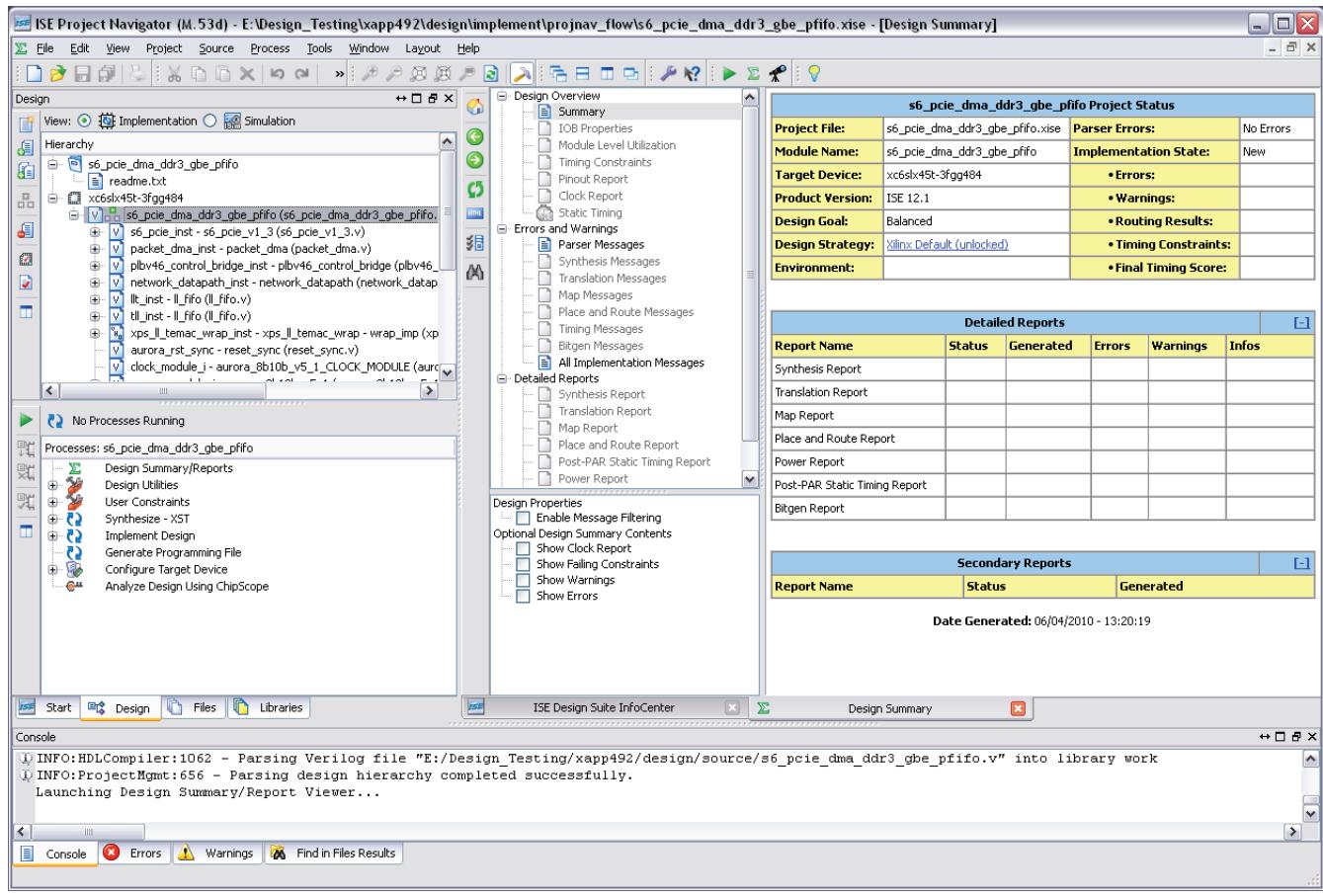
## Hardware Testing

The procedure to test the design in hardware remains the same as explained in UG392 [Ref 1] (Chapter 2) or UG665 [Ref 2].

## Simulation, Implementation, and Project Navigator Flow

The readme.txt file contains details on the use of simulation and implementation scripts.

The Project Navigator GUI is shown in [Figure 7](#). `s6_pcie_dma_ddr3_gbe_pfifo` is the top-level module.



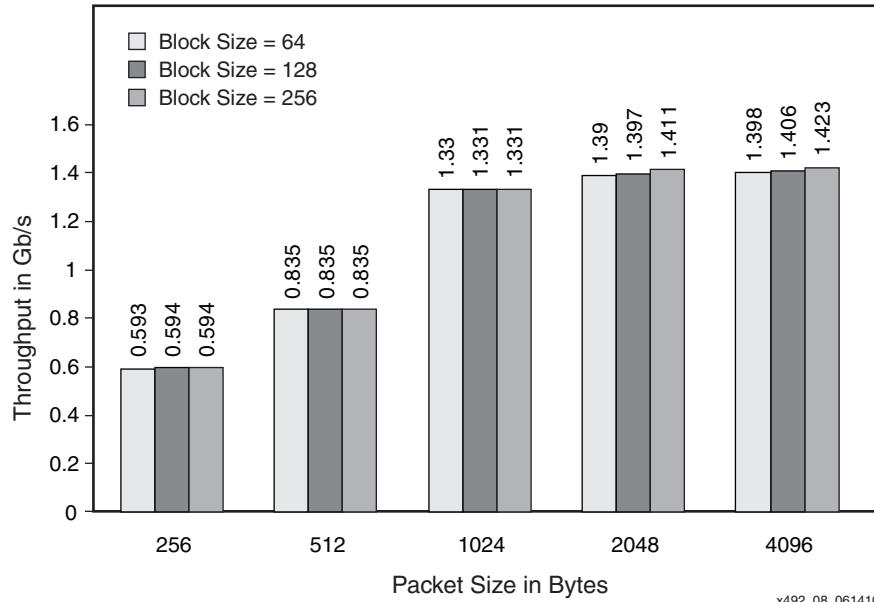
[Figure 7: Project Navigator GUI](#)

## Observations

Key observations on performance and resource utilization of the design are summarized in this section.

### Performance

- As expected, throughput scales with packet size; as packet size increases, throughput increases.
- As shown in [Figure 8](#), a slight variation in throughput is also observed with BLOCK\_SIZE. A higher value of BLOCK\_SIZE implies less frequent control word overhead, which implies availability of more data payload per control word read. A larger BLOCK\_SIZE is slightly better when large sized packets are transported.



*Figure 8: Throughput Variation Summary*

The performance shown in [Figure 8](#) is obtained with the native flow control FIFO thresholds for the Aurora 8B/10B IP derived for the worst-case condition as defined by the Aurora specification. Factors impacting throughput are:

- Store and forward scheme for Packet FIFO
- MCB port arbitration
- Throttling of traffic due to the Aurora NFC

Throughput can be improved by using an Aurora peer and better characterized NFC threshold levels depending upon the round-trip time between Aurora peers.

### Device Resource Utilization

[Table 3](#) provides resource utilization estimate for the design shown in [Figure 2](#). The resource utilization numbers are obtained using the default options for various parameters provided in the top-level design and the implementation options provided in the scripts. Any change of options would result in a change in utilization. The transceiver utilization is reported for the GTPA1\_DUAL; one transceiver in a pair is utilized in the design (not both).

**Table 3: Resource Utilization**

Resource	Utilization	Total Available	Percentage Utilization
Slice Registers	24,593	54,576	45
Slice LUTs	23,213	27,288	85
IOBs	84	296	28
RAMB16BWERS	42	116	36
DCMs	1	8	12
PLL_ADVs	3	4	75
BUFGs	12	16	75
GTPA1_DUALs	2	2	100

## Reference Design

The reference design files can be downloaded at:

<https://secure.xilinx.com/webreg/clickthrough.do?cid=147907>

The reference design checklist is shown in **Table 4**.

**Table 4: Reference Design Checklist**

Parameter	Description
Developer Name	Xilinx Inc.
Target Devices (stepping level, ES, production, speed grades)	Spartan-6 LXT devices
Source Code Provided	Yes
Source Code Format	Verilog
Design Uses Code/IP from an Existing Reference Design/Application Note, Third Party, or CORE Generator™ software	Yes: <ul style="list-style-type: none"><li>• Code from the Spartan-6 FPGA Connectivity TRD</li><li>• Core Generator IP for Aurora 8B/10B</li></ul>
<b>Simulation</b>	
Functional Simulation Performed	Yes
Timing Simulation Performed	No
Testbench Used for Functional Simulations Provided	Yes
Testbench Format	Verilog
Simulator Software Used	ModelSim 6.5c
SPICE/IBIS Simulations	No
<b>Implementation</b>	
Synthesis Software Tools Used	XST (ISE design suite: 12.1 Embedded or System Edition)
Implementation Software Tools Used	ISE design suite: 12.1 Embedded or System Edition
Static Timing Analysis Performed	Yes

**Table 4: Reference Design Checklist (Cont'd)**

Parameter	Description
<b>Hardware Verification</b>	
Hardware Verified	Yes
Hardware Used for Verification	SP605 development board

## Conclusion

This application note demonstrates the Spartan-6 FPGA Connectivity TRD in a platform solution. The example design provided uses an existing infrastructure (the TRD) as a base to create a newer design with limited effort.

## References

1. [UG392, Spartan-6 FPGA Connectivity Targeted Reference Design User Guide](#)
2. [UG665, Spartan-6 FPGA Connectivity Kit Getting Started Guide](#)
3. [UG388, Spartan-6 FPGA Memory Controller User Guide](#)
4. [UG353, LogiCORE™ IP Aurora 8B/10B User Guide](#)
5. [DS637, LogiCORE IP Aurora 8B/10B Data Sheet](#)
6. [SP002, Aurora 8B/10B Protocol Specification](#)

## Appendix A

### Register Description

This section defines the additional registers included for multi-port virtual FIFO and Aurora IP control and status.

**Table 5: User Application Register Range**

User Logic Register Group	Range (Offset from BAR0)
Multi-port Virtual FIFO Status and Control Registers	0x9100–0x91FF
Aurora Status and Control Registers	0x9200–0x92FF
Packetizer Registers	0x9300–0x93FF

The register bits not defined explicitly return a value of zero on read and are assumed to be reserved for all purposes.

### Multi-port Virtual FIFO Status and Control Registers

The multi-port virtual FIFO status and control registers define the registers specific to virtual FIFOs. As multiple ports of the memory controller are used, individual registers are defined for each port.

#### Status Register (0x9100)

The status register indicates the status of DDR3 calibration to the software driver. It enables software to determine if hardware is ready for operation.

**Table 6: Status Register**

Bit Location	Field	Mode	Default Value	Description
0	Calibration Status	RO	1'b0	Calibration Done. This bit indicates calibration done status from memory controller.

#### Write Threshold Register (0x9104)

The write threshold register programs the write threshold value to issue the write commands to the memory controller only when the write FIFO has space equal to or greater than the write threshold value. This register is applicable to all ports.

**Table 7: Write Threshold Register**

Bit Location	Field	Mode	Default Value	Description
7:0	Write Threshold	RW	8'h38	Write threshold for issuing write command to DDR3.

For multiple memory controller ports, the address ranges are defined in [Table 8](#).

**Table 8: Memory Port Registers**

Memory Controller Port	Range (Offset from BAR0)
Port 0	0x9110–0x913F
Port 1	0x9140–0x917F
Port 2	0x9180–0x91BF
Port 3	0x91C0–0x91FF

Though the TRD only supports two memory controller ports, additional registers and logic are provided to support future design scalability with increased memory controller ports.

### **Packet Length Register (0x9110, 0x9140, 0x9180, 0x91C0)**

The packet length register indicates the size of the packet (in bytes) to build in the receive direction. The default value is 1 KB.

**Table 9: Receive Packet Length Register**

Bit Location	Field	Mode	Default Value	Description
12:0	Packet Length	RW	12'h0400	DDR3 receive packet length. Indicates the size of the packet (in bytes) to build in the receive direction.

### **Start Address Register (0x9114, 0x9144, 0x9184, 0x91C4)**

The start address register indicates the start address for DDR3 partitioning. The default value is zero on reset for Port 0 register. For each port, a 1MB FIFO depth is provided and the start and end addresses are programmed accordingly. Software programming of this register is optional.

### **End Address Register (0x9118, 0x9148, 0x9188, 0x91C8)**

The end address register indicates the end address for DDR3 partitioning. The default value is 32'h0010\_0000 on reset indicating 1MB FIFO depth for Port 0. Software programming of this register is optional.

### **Error Statistics Register (0x911C, 0x914C, 0x918C, 0x91CC)**

The DDR3 error statistics register records the status of various error bits from the memory controller. This register is cleared on reset.

**Table 10: Error Statistics Register**

Bit Location	Field	Mode	Default Value	Description
0	Write Underrun	RW	0	Memory controller port write underrun status.
1	Read Overflow	RW	0	Memory controller port read overflow status

## Aurora Status and Control Registers

The Aurora status and control registers group defines the specific Aurora IP status and control registers.

### Aurora Control and Status (0x9200)

The software driver is currently programmed to support near end PMA loopback. The driver can be changed to near-end PCS loopback if required in the `driver/xblockdata/user.c` file.

*Table 11: Aurora Control and Status Register*

Bit Location	Field	Mode	Default Value	Description
0	Lane Up	RO	0	Indicates Aurora lane up status.
1	Channel Up	RO	0	Indicates Aurora channel up status.
31:29	Loopback	RW	0	Control GT loopback: <ul style="list-style-type: none"><li>• 3'b000 - No loopback</li><li>• 3'b001 - Near end PCS loopback</li><li>• 3'b010 - Near end PMA loopback</li></ul>

### Aurora Low-tide Register (0x9204)

The Aurora low-tide register defines the lower bound for space available in the FIFO. Native flow control interface is driven based on the low-tide and high-tide values.

*Table 12: Aurora Low-tide Register*

Bit Location	Field	Mode	Default Value	Description
8:0	Low-tide mark	RW	9'd128	Low-tide mark for FIFO space availability.

### Aurora High-tide Register (0x9208)

The Aurora high-tide register defines the upper bound for space available in the FIFO. Native flow control interface is driven based on the low-tide and high-tide values.

*Table 13: Aurora High-tide Register*

Bit Location	Field	Mode	Default Value	Description
8:0	High-tide mark	RW	9'd384	High-tide mark for FIFO space availability.

## Packetizer Registers

### Packet Error Register (0x9300)

The packet error register provides packetization error status on each packet FIFO port. A packet error results due to the misalignment of the control word in the reassembly of packets. Design reset is required on a packet error.

*Table 14: Packet Error Register*

Bit Location	Field	Mode	Default Value	Description
1:0	Packet Error	RW	2'b0	Indicates packetization error for memory controller ports 0 and 1.

## Revision History

The following table shows the revision history for this document.

Date	Version	Description of Revisions
06/23/10	1.0	Initial Xilinx release.

## Notice of Disclaimer

Xilinx is disclosing this Application Note to you "AS-IS" with no warranty of any kind. This Application Note is one possible implementation of this feature, application, or standard, and is subject to change without further notice from Xilinx. You are responsible for obtaining any rights you may require in connection with your use or implementation of this Application Note. XILINX MAKES NO REPRESENTATIONS OR WARRANTIES, WHETHER EXPRESS OR IMPLIED, STATUTORY OR OTHERWISE, INCLUDING, WITHOUT LIMITATION, IMPLIED WARRANTIES OF MERCHANTABILITY, NONINFRINGEMENT, OR FITNESS FOR A PARTICULAR PURPOSE. IN NO EVENT WILL XILINX BE LIABLE FOR ANY LOSS OF DATA, LOST PROFITS, OR FOR ANY SPECIAL, INCIDENTAL, CONSEQUENTIAL, OR INDIRECT DAMAGES ARISING FROM YOUR USE OF THIS APPLICATION NOTE.

### **CRITICAL APPLICATIONS DISCLAIMER**

XILINX PRODUCTS (INCLUDING HARDWARE, SOFTWARE AND/OR IP CORES) ARE NOT DESIGNED OR INTENDED TO BE FAIL-SAFE, OR FOR USE IN ANY APPLICATION REQUIRING FAIL-SAFE PERFORMANCE, SUCH AS IN LIFE-SUPPORT OR SAFETY DEVICES OR SYSTEMS, CLASS III MEDICAL DEVICES, NUCLEAR FACILITIES, APPLICATIONS RELATED TO THE DEPLOYMENT OF AIRBAGS, OR ANY OTHER APPLICATIONS THAT COULD LEAD TO DEATH, PERSONAL INJURY OR SEVERE PROPERTY OR ENVIRONMENTAL DAMAGE (INDIVIDUALLY AND COLLECTIVELY, "CRITICAL APPLICATIONS"). FURTHERMORE, XILINX PRODUCTS ARE NOT DESIGNED OR INTENDED FOR USE IN ANY APPLICATIONS THAT AFFECT CONTROL OF A VEHICLE OR AIRCRAFT, UNLESS THERE IS A FAIL-SAFE OR REDUNDANCY FEATURE (WHICH DOES NOT INCLUDE USE OF SOFTWARE IN THE XILINX DEVICE TO IMPLEMENT THE REDUNDANCY) AND A WARNING SIGNAL UPON FAILURE TO THE OPERATOR. CUSTOMER AGREES, PRIOR TO USING OR DISTRIBUTING ANY SYSTEMS THAT INCORPORATE XILINX PRODUCTS, TO THOROUGHLY TEST THE SAME FOR SAFETY PURPOSES. TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CUSTOMER ASSUMES THE SOLE RISK AND LIABILITY OF ANY USE OF XILINX PRODUCTS IN CRITICAL APPLICATIONS.

### **AUTOMOTIVE APPLICATIONS DISCLAIMER**

XILINX PRODUCTS ARE NOT DESIGNED OR INTENDED TO BE FAIL-SAFE, OR FOR USE IN ANY APPLICATION REQUIRING FAIL-SAFE PERFORMANCE, SUCH AS APPLICATIONS RELATED TO: (I) THE DEPLOYMENT OF AIRBAGS, (II) CONTROL OF A VEHICLE, UNLESS THERE IS A FAIL-SAFE OR REDUNDANCY FEATURE (WHICH DOES NOT INCLUDE USE OF SOFTWARE IN THE XILINX DEVICE TO IMPLEMENT THE REDUNDANCY) AND A WARNING SIGNAL UPON FAILURE TO THE OPERATOR, OR (III) USES THAT COULD LEAD TO DEATH OR PERSONAL INJURY. CUSTOMER ASSUMES THE SOLE RISK AND LIABILITY OF ANY USE OF XILINX PRODUCTS IN SUCH APPLICATIONS.