WP363 (v1.0) April 19, 2010

Spartan-6 FPGA Connectivity Targeted Reference Design Performance

By: Sunita Jain

This white paper discusses the observed performance of the Spartan®-6 FPGA Connectivity targeted reference design. The design uses PCI Express®, Ethernet, and an integrated memory controller along with a packet DMA for moving data between system memory and the FPGA.

The design demonstrates these applications:

- Network interface card (movement of Ethernet frames between system memory and Tri-Mode Ethernet MAC (TEMAC))
- System memory to external memory (DDR3 SDRAM) data transfer

The goal of the targeted reference design is to achieve maximum performance on the PCIe® and Ethernet links.

About the Targeted Reference Design

The Spartan-6 FPGA Connectivity targeted reference design demonstrates the key integrated components in a Spartan-6 FPGA, namely, the endpoint block for PCI Express, the GTP transceivers, and the memory controller. The targeted reference design integrates these components in an application along with additional IP cores, including a third-party packet direct memory access (DMA) engine (from Northwest Logic), the Xilinx® Platform Studio LocalLink Tri-Mode Ethernet MAC (XPS-LL-TEMAC), and a Xilinx memory controller block (MCB) delivered through the Xilinx memory interface generator (MIG) tool. A block diagram of the targeted reference design is shown in Figure 1.

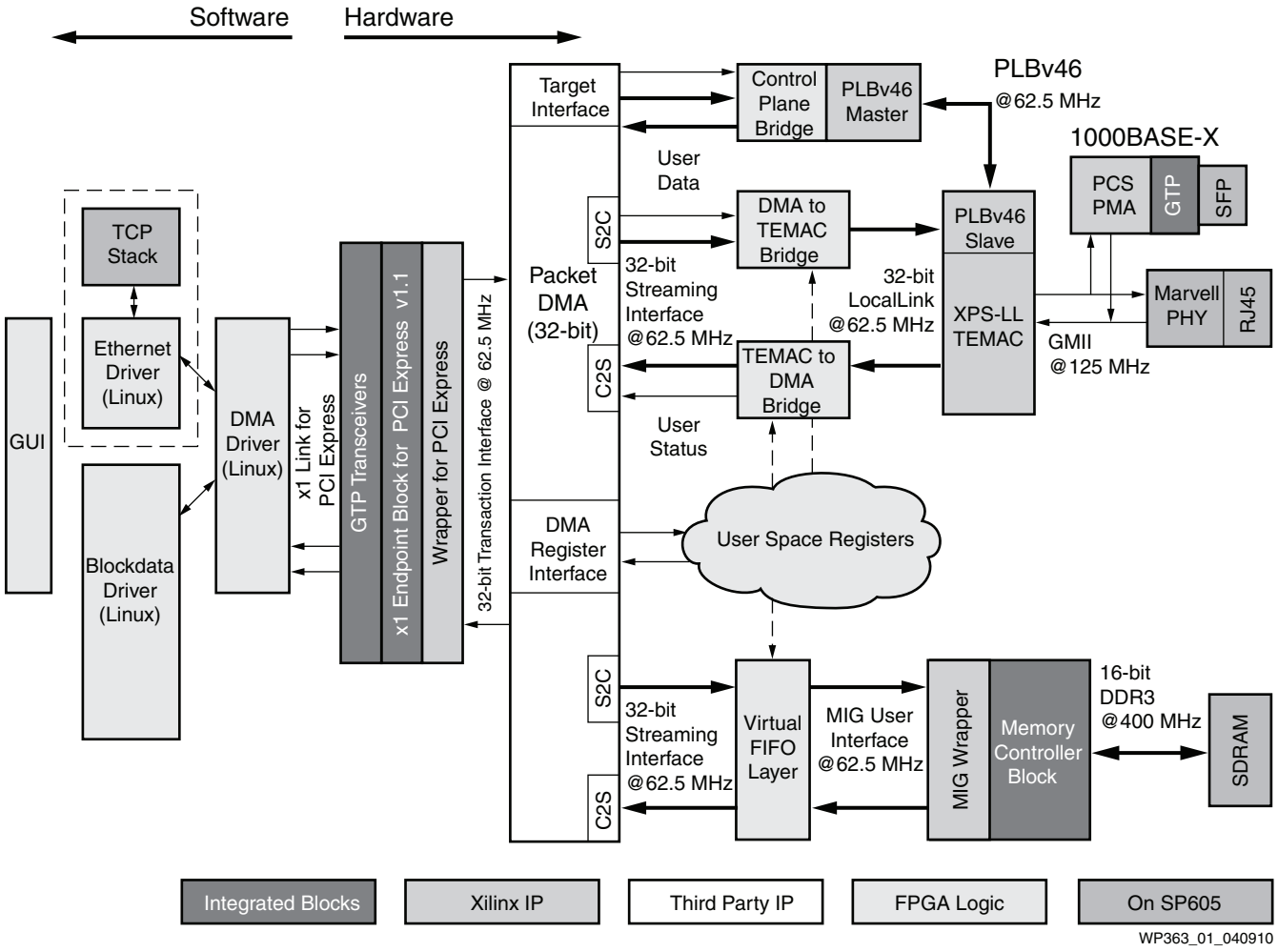

Figure 1: **Design Block Diagram**

This design is a v1.1-compliant x1 Endpoint block for PCI Express showcasing these independent applications:

- A network interface card (NIC) that enables movement of Ethernet packets between the TCP/IP stack running on the host and the external network via the FPGA. The NIC provides two modes of operation:
  - GMII mode using external Ethernet PHY (typically used to connect to copper Ethernet networks)

- 1000BASE-X mode using GTP transceivers (typically used to connect to optical fiber Ethernet networks)
- An external memory interface over PCI Express that enables data movement between host memory and external DDR3 SDRAM via the FPGA.

Fully independent transmit and receive path implementations coupled with prefetching buffer descriptors from packet DMA help maximize performance. For more details on the targeted reference design, refer to the *Spartan-6 FPGA Connectivity Targeted Reference Design User Guide* [Ref 1].

A brief description of the data flow in the design is provided here for both network and block data paths.

Network Data Flow

Ethernet packets prepared by the TCP/IP stack are queued up for direct memory access (DMA) by the software driver. The packets are transferred to the FPGA over PCIe and then routed to the Ethernet MAC for transmission on the network. Packets received from the external network by the Ethernet MAC are moved into system memory through PCIe, and the software driver passes them onto the TCP/IP stack.

Block Data Flow

Data to external memory is written in blocks. Software generates the data and queues it up for DMA transfer. The block data is then written to external memory, read back from external memory, and sent back to system memory through PCIe.

Test Setup

The test setup uses the Spartan-6 FPGA Connectivity Kit as the hardware and software platform, providing the SP605 board and the targeted reference design as well as the Fedora 10 LiveCD OS.

The systems used for the performance measurement have these specifications:

- Intel x58 2.67 GHz motherboard, 3 GB RAM, 256 KB L2 Cache, 8,192 KB L3 Cache, hyper-threaded eight-CPU server machine running Fedora 10 LiveCD 2.6.27.5. (MPS = 256B and MRRS = 512B for the PCIe slot used.)
- ASUS computer with single hyper-threaded 1 GHz Intel Pentium 4 processor, 1 MB cache, two-CPU desktop machine running Fedora 10 LiveCD 2.6.27.5. (MPS = 128B and MRRS = 512B for the PCIe slot used.)
- Dell machine with 1.80 GHz Intel Pentium dual CPU running Fedora Core 6 2.6.18. (MPS = 128B, MRRS = 512B for the PCIe slot used.)

Performance of PCI Express

PCI Express is a serialized, high-bandwidth, scalable point-to-point protocol that provides highly reliable data transfer operations. The maximum transfer rate of a v1.1 compliant PCIe core is 2.5 Gb/s. This rate is the raw bit rate per lane per direction and not the actual data transfer rate. The effective data transfer rate is lower due to protocol overhead and other system design trade-offs [Ref 2].

Theoretical PCI Express Performance Estimate

The maximum line rate supported by the PCIe protocol specification v1.1 is 2.5 Gb/s. This effectively becomes 2 Gb/s after accounting for 8B/10B encoding overhead.

The following contribute to the the protocol overhead associated with payload transfer:

- The transaction layer adds 12 bytes of header (considering only 32-bit addressable transactions without the optional end-to-end cyclic redundancy check (ECRC)).
- The data link layer adds 2 bytes of sequence number and 4 bytes of link layer cyclic redundancy check (LCRC).
- The physical layer adds 1 byte of start-framing and 1 byte of end-framing information.

The protocol overhead is illustrated in Figure 2. For more information, refer to *Understanding Performance of PCI Express Systems* [Ref 2]).

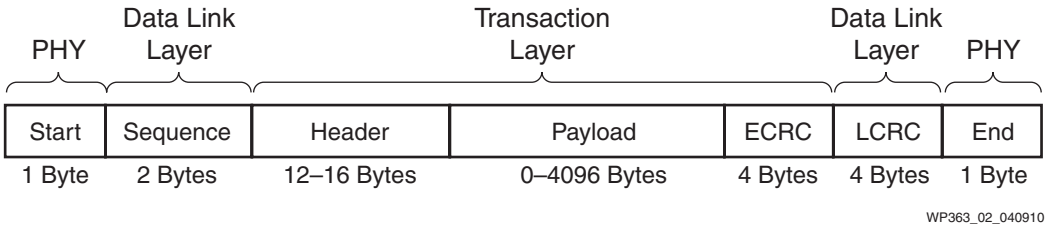

Figure 2: **PCI Express Protocol Packet Overhead**

The PCIe protocol also uses additional data link layer packets for flow control updates and ACK/NAK to help in link management. Neither is accounted for in this performance estimation.

Note: The following estimation does not represent achievable bandwidth, because it assumes that all lanes are transmitting transaction layer packets (TLPs) at all times without any gaps or physical or data link layer overhead. Therefore, it only provides an indication of the upper performance boundary.

Equation 1 and Equation 2 define the throughput estimate and efficiency, respectively.

$$\text{Throughput Estimate} = \left[\frac{\text{Payload}}{\text{Payload} + 12 + 6 + 2}\right] \times 2 \qquad \textit{Equation 1}$$

$$\text{Efficiency} = \left[\frac{\text{Payload}}{\text{Payload} + 12 + 6 + 2}\right] \times 100 \qquad \textit{Equation 2}$$

An estimate of PCIe performance is shown in Table 1 with values obtained using Equation 1 and Equation 2 for various payload sizes.

Table 1: **PCIe Performance (Theoretical Estimate)**

Payload Size (Bytes)	Throughput (Gb/s)	Efficiency (%)
64	1.52	76.19
128	1.72	86.48
256	1.85	92.75
512	1.92	96.24
1024	1.96	98.08
2048	1.98	99.03

Figure 3 shows that protocol overhead prevents the throughput from reaching the theoretical maximum. Beyond a payload size of 512 bytes, there is no significant gain in performance.

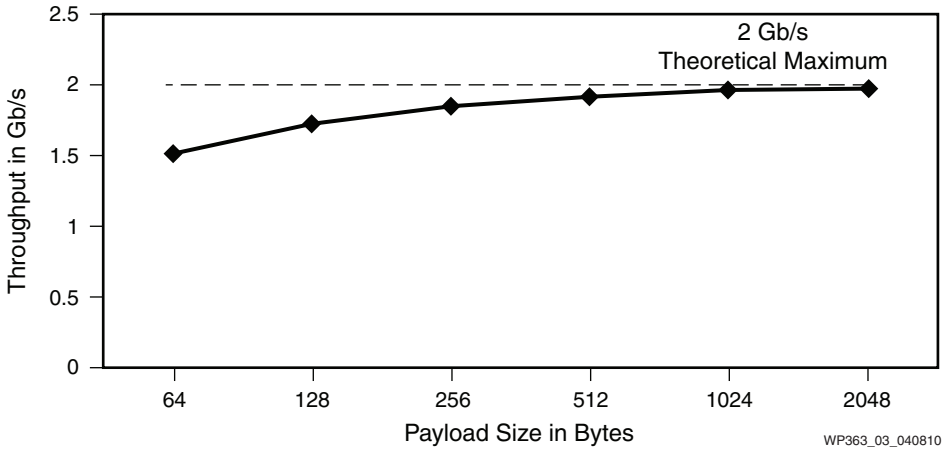

Figure 3: **Theoretical PCI Express Throughput**

Additionally, system parameters like maximum payload size, maximum read request size, and read completion boundary impact PCIe performance. For a detailed understanding of various factors impacting the PCIe link performance, refer to *Understanding Performance of PCI Express Systems* [Ref 2].

PCI Express Performance Results

This section presents the PCI Express performance results as seen on the memory path using the block data transfer to external memory. The Ethernet application (network path) provided in the design supports a maximum throughput of 1 Gb/s. Thus, block data transfer is used to maximize PCIe performance, which is measured against these metrics:

- DMA throughput: This includes a count of actual payload bytes available at the DMA user interface.
- PCIe transmit utilization: This includes utilization of the TRN-TX interface.
- PCIe receive utilization: This includes utilization of the TRN-RX interface.

The PCIe transmit and receive transaction utilization metrics include overhead due to DMA buffer descriptor management and register setup, in addition to data payload throughput. These metrics are the total of all PCIe transaction utilizations necessary to achieve the observed DMA throughput.

Counters in hardware are used to gather the above metrics. The DMA throughput estimate is provided by the counters present inside the DMA. For PCIe transaction utilization measurement, counters snooping over the TRN-TX and TRN-RX interfaces are implemented. These counters map to user space registers, which are periodically read by the software driver to get a real-time performance estimate.

Performance Variation with Maximum Payload Size

Figure 4 shows performance variation with change in maximum payload size (MPS) on a PCIe link for a fixed block size of 1,024 bytes. MPS is one of the many factors impacting performance.

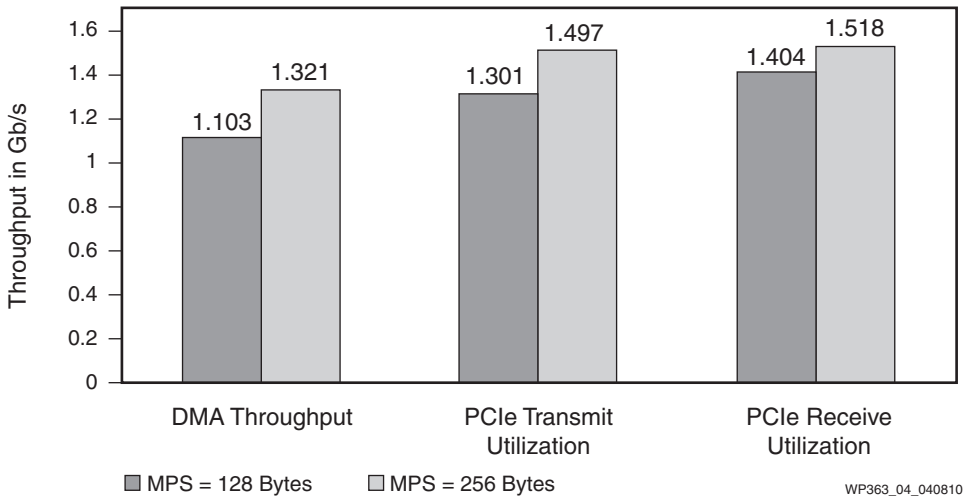

Figure 4: **Performance Variation with MPS**

From Figure 4, it can be seen that:

- Higher MPS improves performance, because more data can be sent when compared to TLP header overhead for a given transaction.
- The DMA throughput plotted is similar for both transmit and receive directions. The PCIe transaction utilization includes DMA setup overhead and register read/write operations. Thus, the throughput observed is higher than the actual payload throughput on DMA.
- PCIe receive utilization (downstream traffic due to reads) is slightly higher than PCIe transmit utilization (upstream traffic due to writes) due to the fact that an additional descriptor fetch of 32 bytes is counted in the PCIe receive direction as compared to a 4-byte or 12-byte descriptor update in the PCIe transmit direction.

The memory path loops back data—that is, it reads data from system memory, writes it to external memory, and then reads data back from external memory and writes it to system memory. This essentially indicates that the performance is controlled by the PCIe read transactions, because they are slower than the write transactions. The data available through reads is written back to the system memory. Because the data for this test is generated by the software driver itself, care is taken to generate data buffer addresses aligned to system page boundaries to optimize performance.

Performance Variation with Data Block Size

As data block size increases, throughput improves. This can be seen in Figure 5. The results shown are obtained with MPS = 256 bytes.

The test is set up so that one data block is defined by one descriptor to reflect the true impact of data block size. If one descriptor had defined a fixed block size (e.g., 256 bytes), then fetching a 256-byte data block would result in a fetch of one descriptor (the result of just one read), while fetching a 1,024-byte data block would result in a fetch of four descriptors (the result of four reads). The performance would, in the latter case, be characteristic of an aggregate 256-byte data block size rather than a true 1,024-byte block size.

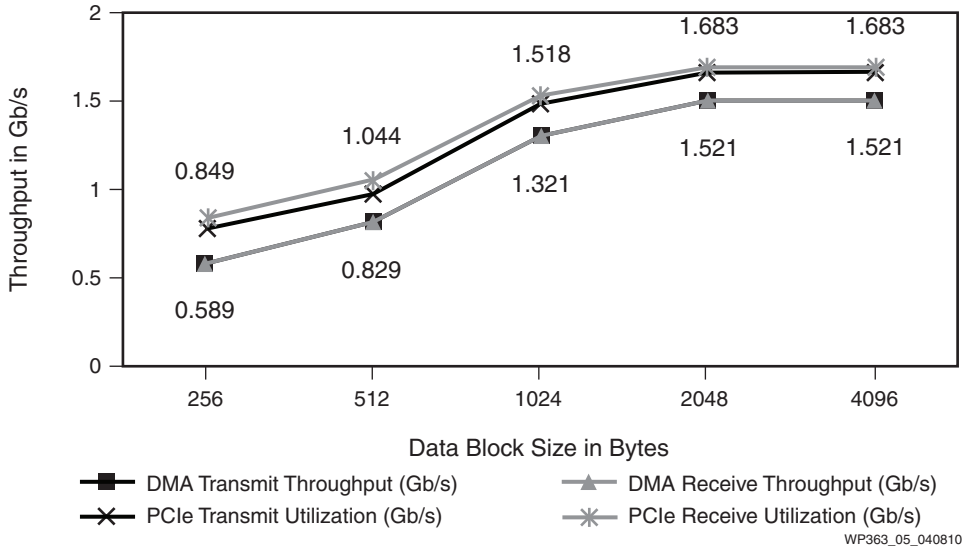

Figure 5: **Throughput Variation with Data Block Size**

The PCIe utilization plots in Figure 5 also indicate additional overhead required to set up data transfers using DMA. The DMA throughput measures the actual payload transferred between system memory and hardware. The PCIe utilization in the receive direction is more than in the transmit direction, as can be seen in Figure 5. This is because an additional descriptor fetch of 32 bytes is counted in the PCIe receive direction, as compared to a 4-byte or 12-byte descriptor update in the PCIe transmit direction. This per-packet difference is seen more for smaller block sizes than for larger block sizes.

Ethernet Performance

The raw line rate of the Ethernet link is 1.25 Gb/s. This is commonly referred to as Gigabit Ethernet after accounting for encoding overhead. The 1000BASE-X link uses 8B/10B encoding, while the 1000BASE-T link uses 4D PAM-5 encoding. The actual data rate thus comes out to be 1,000 Mb/s [Ref 3].

Theoretical Ethernet Performance Estimate

The performance of various Ethernet applications at different layers is less than the throughput seen at the software driver and the Ethernet interface. This is due to the various headers and trailers inserted in each packet by the layers of the networking stack. Ethernet is only a medium to carry traffic. Various protocols, such as transmission control protocol (TCP) and user datagram protocol (UDP) implement their own header/trailer formats.

To estimate theoretical performance, consider raw Ethernet traffic not accounting for network protocol overheads, and traffic accounting for TCP. The protocol header includes:

- TCP/IP Overhead: 20-byte TCP header + 20-byte IP header
- Ethernet Overhead: 14-byte Ethernet header + 4-byte trailer (as a frame check sequence)

Additionally, there are 8 bytes of preamble with a start-of-frame delimiter and 12 bytes of interframe gap on the wire.

Based on the above information, the throughput calculations can be formulated as shown in Equation 3 and Equation 4, where Payload is the Ethernet frame payload.

$$\text{Raw Throughput} = \left[\frac{\text{Payload}}{\text{Payload} + 18 + 20}\right] \times 1000 \qquad \textit{Equation 3}$$

$$\text{TCP/IP Throughput} = \left[\frac{\text{Payload}}{\text{Payload} + 18 + 20 + 40}\right] \times 1000 \qquad \textit{Equation 4}$$

Table 2 shows the theoretical raw Ethernet throughput (without network protocol overhead) and TCP/IP throughput, which does account for network protocol overhead [Ref 4].

Note: This computation is purely for Ethernet and does *not* account for the PCIe protocol or DMA management overhead.

Table 2: **Theoretical Ethernet Throughput Calculation**

Payload Size (Bytes)	Ethernet Traffic Throughput (Mb/s)	
	Raw	TCP/IP
46	547.61	370.96
64	627.45	450.70
128	771.08	621.35
256	870.74	766.46
512	930.90	867.79
1,024	964.21	929.21
1,442	974.32	948.68
1,472	974.83	949.67
1,496	975.22	950.44

A plot of the theoretical throughput in Figure 6 shows that protocol overhead prevents the gigabit Ethernet NIC from even theoretically approaching the wire speeds. The protocol overhead negatively impacts the throughput for payload sizes less than 512 bytes.

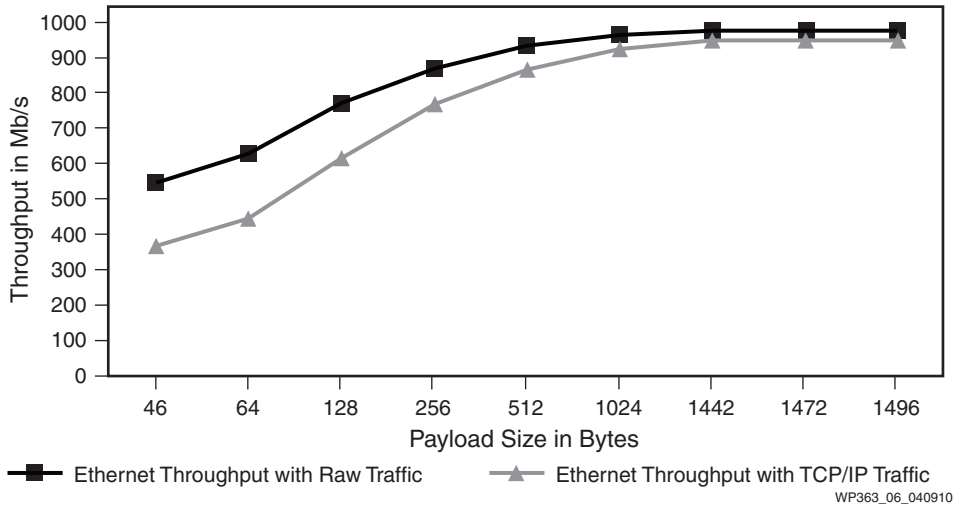

Figure 6: **Raw Traffic Vs TCP/IP Traffic Theoretical Throughput**

Ethernet Performance Results

This section describes the Ethernet performance results as observed with Netperf.

Test Methodology

Netperf v2.4 was used as the testbench for measuring Ethernet performance [Ref 5]. Netperf is a data transfer application running on top of the TCP/IP stack, and it operates around a basic client-server model. The client could be configured for different message sizes and to open a TCP or UDP connection. After establishing a connection, either the client or server (depending on the direction in which performance is being measured) transmitted a stream of full-size 1,514-byte packets. Only ACK packets were sent in the opposite direction.

For performance measurement, a socket size of 16 KB was used. The two systems were connected in a private LAN connection to avoid other external LAN traffic.

For Ethernet performance measurement, the setup consisted of two PC machines connected in a private LAN environment, as shown in Figure 7. The Spartan-6 FPGA Connectivity Kit's NIC was hosted on an ASUS desktop machine.

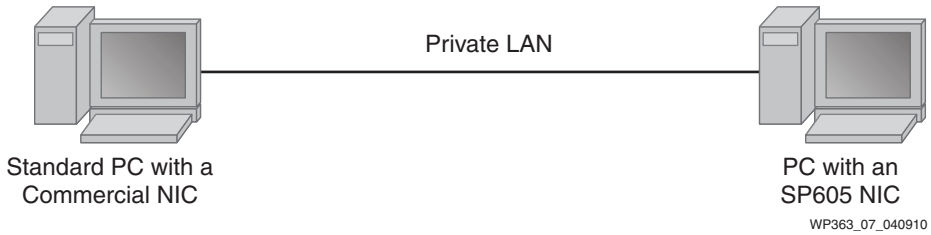

Figure 7: **Ethernet Performance Measurement Setup**

To measure the CPU utilization as accurately as possible, no other applications (other than the standard ones) were run during the test. The measurements shown were obtained with the Netperf TCP Stream test. This test measures unidirectional transmission of TCP data and does not include connection establishment time. This measurement indicates performance for processing outbound packets, i.e., from the Netperf client running on the host PC towards the net server running on a remote PC.

Throughput Analysis

Throughput was measured while varying the message size in Netperf. The maximum transmission unit (MTU) was kept at 1,500 bytes, and a socket size of 16,384 bytes was used.

The message size is the send buffer size, which indirectly impacts the size of the packets sent over the network. MTU size decides the maximum size of a frame that can be sent on the wire. Figure 8 depicts the Ethernet throughput measured for outbound traffic.

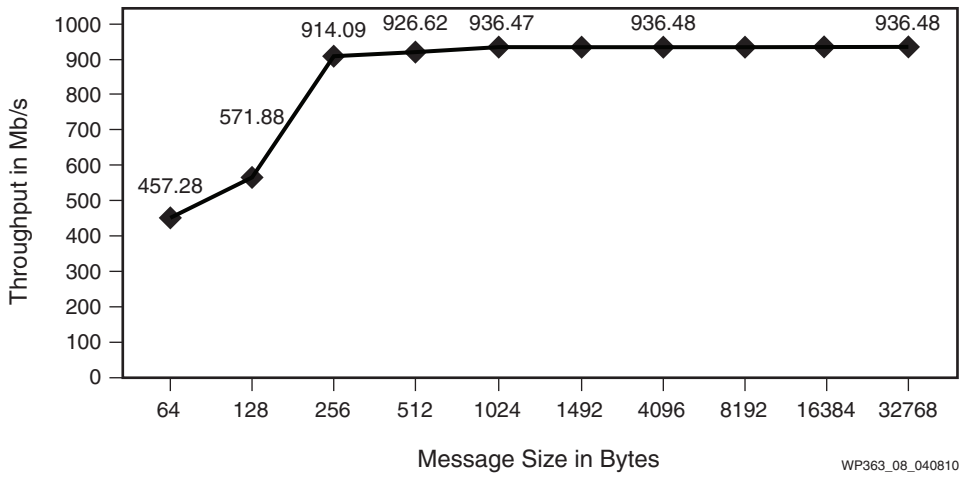

Figure 8: **Outbound Ethernet Throughput**

It can be deduced from Figure 8 that when the message size is 256 bytes or less, the TCP/IP protocol overhead has a larger impact and thus reduces the throughput. With larger message sizes, the protocol overhead decreases, thereby increasing the throughput. Beyond a message size of 1,024 bytes, throughput stabilizes at a value of around 936 Mb/s. The result also demonstrates that performance scales with variation in packet size.

Frames greater than 1,514 bytes are categorized as "jumbo" frames. The impact of jumbo frames on throughput with increasing MTU size is shown in Figure 9. The results are reported for a message size of 8,192 bytes.

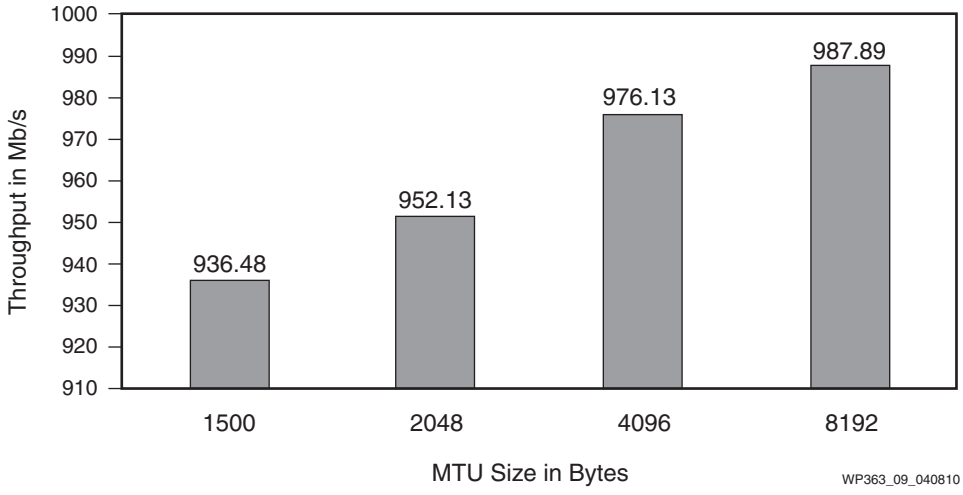

Figure 9: **Frame Throughput (Outbound)**

Thus, increasing MTU size improves performance because larger size packets can be sent. This means that a packet can have more payload for the same header. Using a large MTU size allows the operating system to send fewer packets of a larger size to reach the same data throughput.

Note: For higher MTU sizes, the highest performance seen is greater than 980 Mb/s, compared to approximately 936 Mb/s, as seen with an MTU of 1,500 bytes.

CPU Utilization and Checksum Offload

Checksum is used to ensure the integrity of data during transmission. The transmitter calculates the checksum of data and transmits the data with its checksum. The receiver calculates the checksum using the same algorithm as the transmitter and compares the checksum calculated against the checksum received in the data stream. If the calculated and received checksums do not match, a transmission error occurs.

Figure 10 shows the results for TCP data payload checksum offload. (This design supports only partial checksum offload for TCP/UDP packets.) As seen in the figure, checksum offload improves CPU utilization for large size packets.

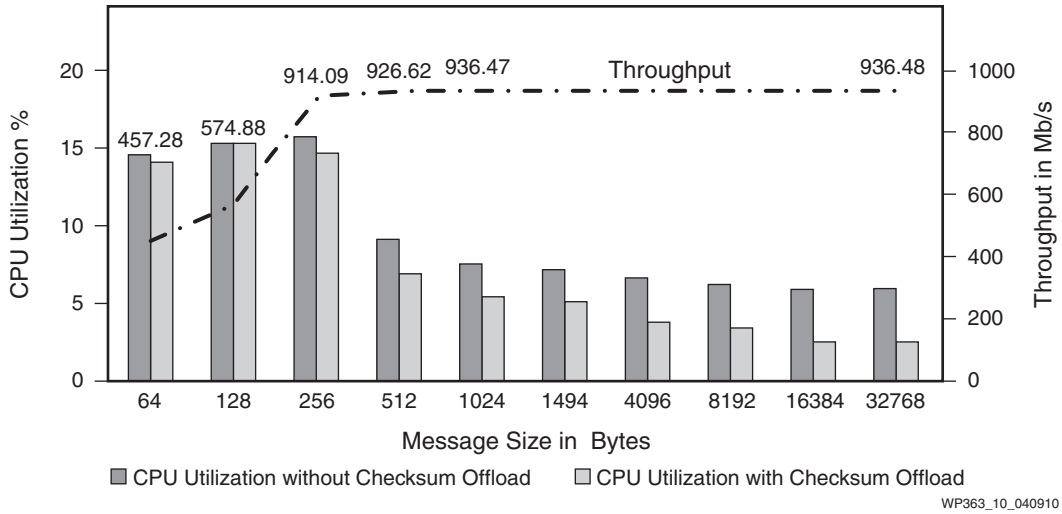

Figure 10: **Utilization with Checksum Offload (Outbound Traffic)**

Operations like checksum calculation, which are expensive when done in software on large packets, should thus be offloaded to hardware so that CPU time can be efficiently utilized elsewhere. With small packets, checksum computation does not contribute as much to CPU utilization overhead as does the protocol overhead of using small packets.

Beyond a message size of 512 bytes, it can be seen that CPU utilization decreases appreciably. For the same throughput, the number of TCP/IP packets processed decreases with an increase in packet size. As a result, processing fewer packets consumes fewer CPU cycles.

On the Fedora 10 LiveCD Linux operating system (kernel version 2.6.27.5), it is observed that enabling checksum offload results in the TCP/IP stack providing Ethernet frames in fragments for transmission, i.e., one frame is split across multiple buffer descriptors. The results described here indicate that the performance improvement gained by checksum offload is offset by the fragmentation of frames. No significant improvement in throughput is seen with checksum offload.

Throughput Variation with Descriptor Count

Figure 11 shows throughput variation with descriptor count per descriptor ring. Each DMA engine has one descriptor ring. With a smaller number of descriptors in the ring, transmit performance is low, because only a small number of packets can be queued for transmission. Similarly, reduced descriptor count in the receive ring drains the incoming packets at a slower rate. It is also seen that with smaller descriptor counts, a larger number of retransmissions occur, which explains the reduced inbound throughput. With fewer descriptors, the software overhead required to manage

descriptors is higher. Thus, the DMA drains packets at a slow rate from the Ethernet MAC. This causes the receive buffer in the Ethernet MAC to overflow, resulting in dropped packets. These dropped packets are responsible for the retransmissions.

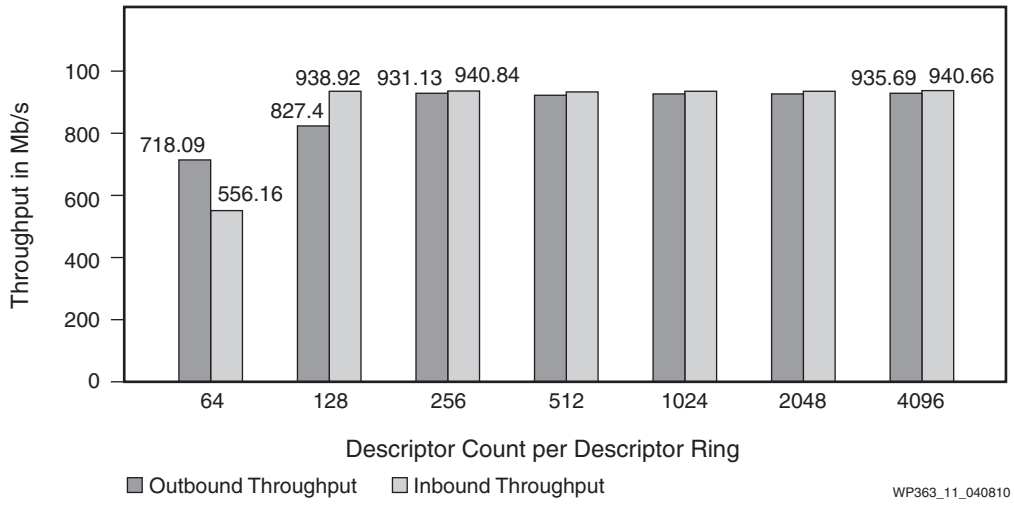

Figure 11: **Throughput Variation with Descriptor Count**

Beyond a descriptor count of 512, the performance remains stable because enough descriptors are available in both transmit and receive directions.

Interrupt Mode Performance

The software driver supports both interrupt and polled modes of operation. For interrupt mode operation, the processor is interrupted on a packet-count basis instead of on a per-packet basis. This count is known as the *coalesce count*.

Figure 12 shows the interrupt mode performance as seen with legacy interrupts obtained by varying the coalesce count.

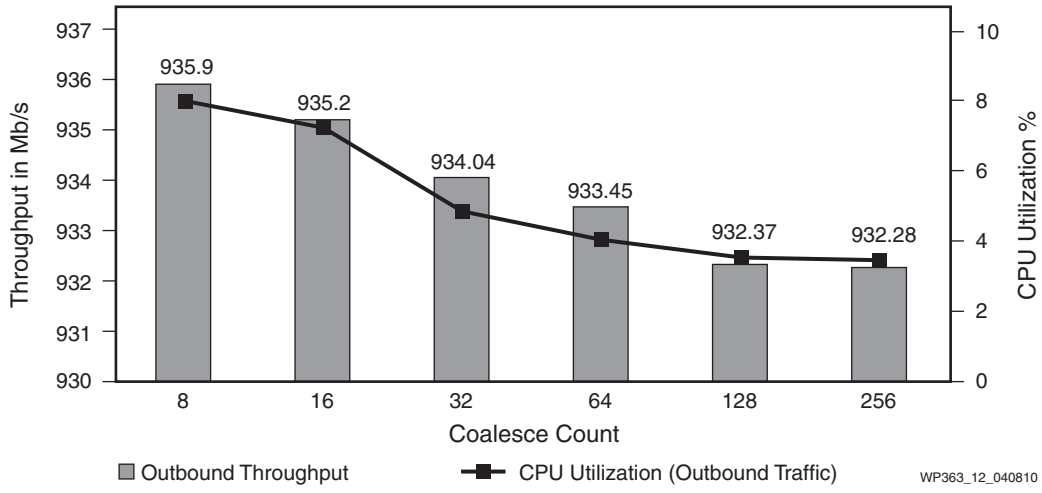

Figure 12: **Throughput and CPU Utilization in Interrupt Mode (Outbound Traffic)**

Figure 12 shows that performance deteriorates slightly with increasing coalesce count values. This is attributed to the fact that larger coalesce counts result in retransmissions due to network protocol timeouts because packets wait longer in queue to be processed.

It can also be observed that CPU utilization is high for lower coalesce count values. Smaller coalesce count values indicate frequent interrupts.

Comparing polled-mode performance against interrupt-mode performance, it can be seen that CPU utilization for a coalesce count of 32 is 4.85% as compared to 6.02% measured in polled mode, but with a reduction in throughput (934.04 Mb/s versus 936.48 Mb/s in polled mode). On the PCIe link, the interrupts move upstream as TLPs consume link bandwidth. Although the amount of bandwidth consumed is small, it is still sufficient to impact the throughput as credits are consumed. There are no interrupts in polled mode, but a timer in the software driver frequently polls the status of various descriptor rings. CPU utilization is thus higher in polled mode than in interrupt mode.

Summary

A summary of Ethernet performance is shown in Figure 13.

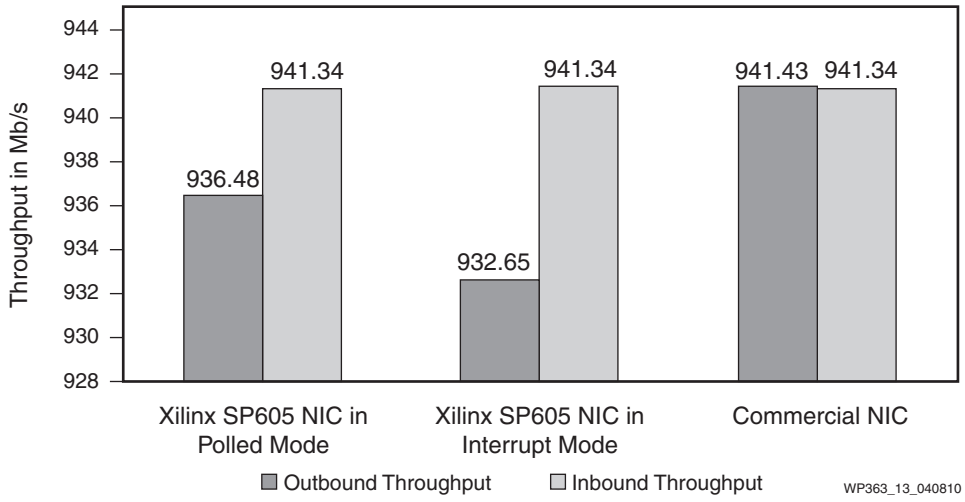

Figure 13: **Ethernet Performance Summary**

The Spartan-6 FPGA Connectivity Kit's NIC performance is measured against a commercial NIC so that true performance is reflected. The inbound performance clearly indicates that the Xilinx NIC can handle commercial NIC throughput. The outbound performance for the Xilinx NIC is lower when compared to the commercial NIC.

Conclusion

This white paper summarizes performance as measured on the Spartan-6 FPGA Connectivity targeted reference design available with the Spartan-6 FPGA Connectivity Kit. The results clearly demonstrate that:

- Throughput scales with variation in packet size.
- PCI Express performance improves with higher maximum payload size.
- CPU utilization improves with checksum offloading.
- An inverse relationship exists between packet size and CPU utilization.

The Spartan-6 FPGA Connectivity targeted reference design provides a high-quality, fully validated, and performance-profiled system that can help jump-start end user development, thereby reducing time to market.

References

1. UG392, *Spartan-6 FPGA Connectivity Targeted Reference Design User Guide*
2. WP350, *Understanding Performance of PCI Express Systems*
3. *Introduction to Gigabit Ethernet*, Cisco Technology Digest
   http://www.cisco.com/en/US/tech/tk389/tk214/tech_brief09186a0080091a8a.html
4. *Gigabit Ethernet Performance in rp7410 Servers*, Hewlett-Packard Performance Paper, September 2002
   http://docs.hp.com/en/2176/GigE_rp7410_perf.pdf
5. Netperf documentation
   http://www.netperf.org

Additional Information

1. *Small Traffic Performance Optimization for 8255x and 8254x Ethernet Controllers*, Intel Application Note AP-453, September 2003
   http://download.intel.com/design/network/applnots/ap453.pdf
2. *HP-UX IPFilter Performance*, Hewlett-Packard White Paper
   http://docs.hp.com/en/13758/ipfiltperf.pdf
3. *1000BASE-T Delivering Gigabit Intelligence*, Cisco Technology Digest
   http://www.cisco.com/en/US/tech/tk389/tk214/tech_digest09186a0080091a86.html

Revision History

The following table shows the revision history for this document:

Date	Version	Description of Revisions
04/19/10	1.0	Initial Xilinx release.

Notice of Disclaimer