Performance of System Interconnect using PCI Express® Switches

Application Note AN-547

Notes

By Kwok Kong

Introduction

A multi-peer system using a standard-based PCI Express® (PCIe) multi-port switch as the System Interconnect was described in an IDT white paper by Kwok Kong [1]. Since the release of that white paper, IDT has designed and implemented a multi-peer system using a x86-based system as the Root Processor (RP) and Endpoint Processor (EP) connecting through IDT's PES16NT8 Non-Transparent Bridge (NTB) port [2] and IDT's PES64H16 [3] device as the multi-port PCIe switch for the System Interconnect. A detailed description of the software architecture can be found in application note AN-571 [4]. This paper presents the measured system data transfer performance of such a system.

System Description

A multi-peer system topology is shown in Figure 1. A x4 PCIe interface is used to connect each Root Processor and Endpoint Processors to the PES64H16 System Interconnect PCIe switch. This is the topology that is used to measure the system data transfer performance.

A PES16NT2 is used to provide the NTB function in order to connect a x86-based Endpoint Processor to a downstream port of a PES64H16 PCIe switch. The System Interconnect software provides a virtual Ethernet over PCIe interface. The Linux Operation System (OS) detects a network interface and "sees" an Ethernet interface. The Linux OS sends Ethernet packets to the PCIe interface as if it were an Ethernet interface. The PCIe interface is hidden from the Linux OS as far as data transfer is concerned. All current networking protocol stacks, such as TCP/IP protocol stack, and user applications that are able to run on top of the TCIP/IP stack will work without any modification.

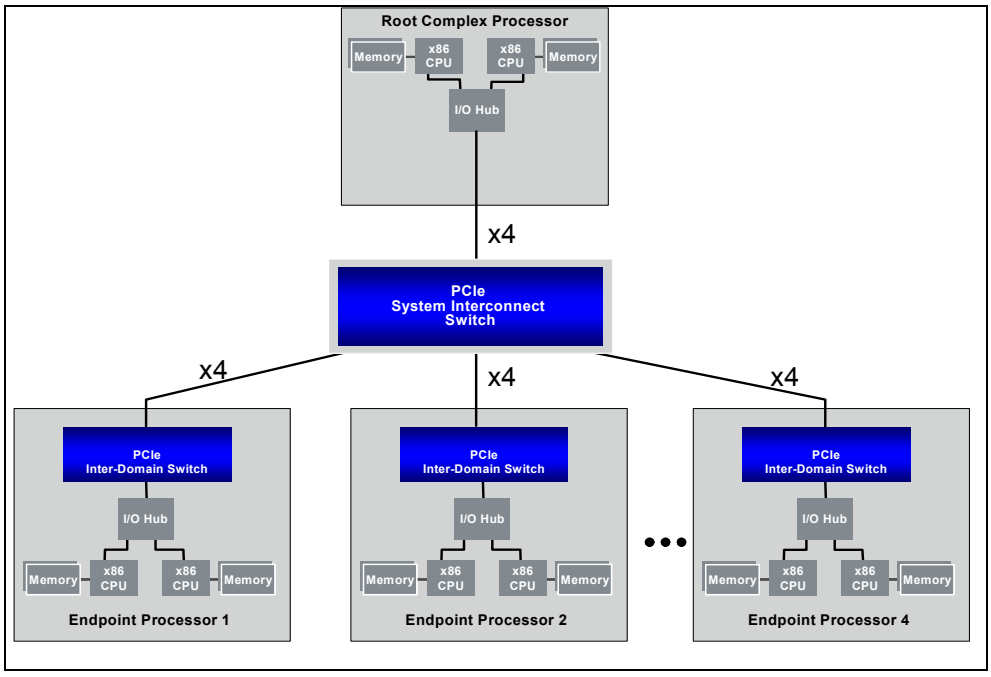

Figure 1 System Interconnect Topology

Notes

The configuration of the Root Processor and Endpoint Processors is shown in Table 1. Both the Bensley and Stoakley platforms support Direct Memory Access (DMA) engines (also known as IOAT technology) in the Northbridge to transfer data between the local memory to memory and from local memory to the PCIe Memory Mapped IO (MMIO) address space. The System Interconnect software utilizes the DMA engine to transfer data from local memory to the MMIO address space on the transmit path. When an incoming packet is received, the DMA engine is utilized to transfer data from local memory to local memory.

The AMD system does not support any DMA engine. Write Combining is configured on the AMD system to transfer data from its local memory to the MMIO address space. Whenever the AMD CPU writes to the PCIe MMIO address space, data is not sent to the PCIe interface but cached in the write combining buffer. When it has accumulated 64 bytes of data, all 64 bytes data is sent out to the PCIe interface as a single PCIe packet. This improves the write performance to the PCIe interface.

	Root Processor	Endpoint Processor 1 (Bensley)	Endpoint Processor 2 (Stoakley)	Endpoint Processor 3 and 4 (AMD)
CPU	Intel Xeon E5310	Intel Xeon E5310	Intel Xeon E5310	AMD Athlon 64 3500+
Number of cores	4	4	4	1
Speed (GHz)	1.6	1.6	1.6	2.2
RAM (G Bytes)	2	2	2	1
Mother board	SuperMicro X7DBE	SuperMicro X7DBE	SuperMicro X7DWA-N	MSI K8N NE04 Platinum SLI
North Bridge	Intel 5000P	Intel 5000P	Intel 5400	nVidia CK804
OS (Linux)	Fedora 6 (2.6.18)	Fedora 6 (2.6.18)	Fedora 6 (2.6.18)	Fedora 6 (2.6.18)
System Interconnect Release	2.0	2.0	2.0	2.0

Table 1 Root Processor and Endpoint Processors Configuration

How Performance is Measured

The benchmarking software Netperf Version 2.4.3 [5] is used for all the performance measurements. Netperf was chosen for the following reasons:

- it is widely used as a data transfer performance measurement tool
- it is freely available and all the tests can be duplicated easily
- It runs as a user level application that closely represents the actual system performance of a typical user application

Netperf is designed around a basic client-server model. There are two programs, netperf and netserver. Netserver is the server program running continuously and waiting for incoming requests from the netperf client program. The TCP/IP protocol was chosen to measure the unidirectional data transfer performance. Once a TCP connection is established between netperf and netserver, netperf sends data to the netserver as fast as possible without any data lost. Netperf keeps sending data for 10 seconds, and the average unidirectional data transfer rate is calculated. This test is repeated for user data sizes of 16K, 8K, 4K, 2K, 1K, and 1/2 K bytes.

Notes

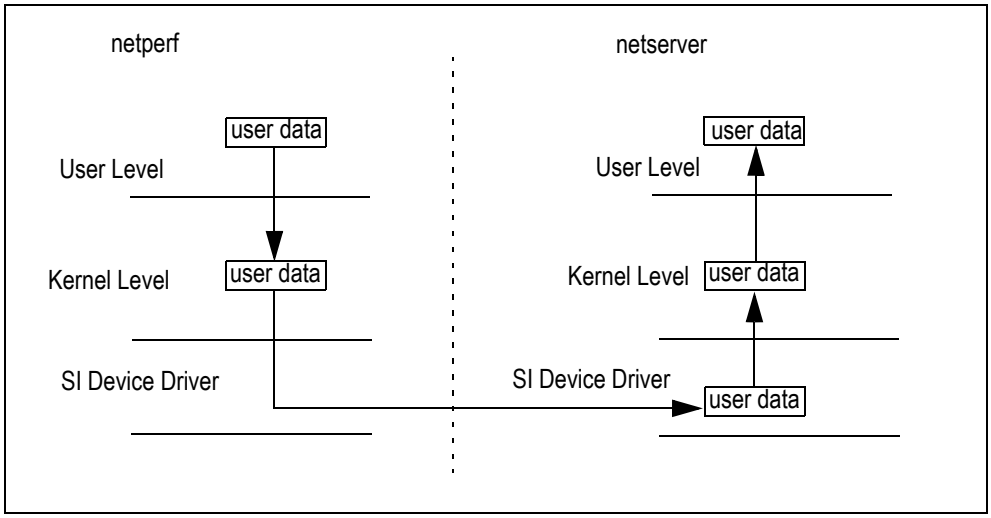

Figure 2 Netperf User Data Movement

The data movement of netperf is shown in Figure 2. Netperf first creates user data in a transfer buffer in the user level address space. It then sends the data to the kernel using the BSD socket API. The Linux kernel copies the user data from the user level address space to its internal TCP/IP data transfer buffer in the kernel address space. This is a local memory to memory copy. The System Interconnect device driver copies the user data in the kernel address space to the receive buffer of the System Interconnect host which is running netserver. This is a local memory to PCIe Memory Mapped IO copy.

When the System Interconnect device driver receives the user data in its receive buffer, the data is copied to a buffer in the kernel address space. This is a local memory to memory copy. When the TCP/IP stack receives the incoming user data, the user data is copied to the user level receive buffer in order to pass the incoming data to netserver. This is also a local memory to memory copy.

The transmit side (netperf) needs to perform one local memory to memory copy and one local memory to PCIe MMIO address space copy. The receive side (netserver) needs to perform two local memory to memory copies.

The data transfer performance may be limited by the performance of the CPU or the speed of the interface between the two hosts that is running netperf and netserver. Netperf and TCP/IP stack require a certain number of CPU cycles per gigabit of data transfer. There is an upper limit on the data transfer performance that is independent of the interface's speed. For example, if the CPU can only handle a data transfer rate of X Gbps, then even if the interface speed is 10 times X Gbps, the data transfer performance cannot exceed the CPU rate of X Gbps. This restriction is called CPU-bound.

If the speed of the interface is less than X Gbps, then the data transfer performance cannot exceed the speed of the interface. This restriction is called interface-limited.

The System Interconnect data transfer performance should be as close to the CPU-bound performance as possible.

Data Transfer Performance

In order to determine if the data transfer performance is good or bad, a reference must first be established. The CPU-bound (or close to) data transfer performance is first measured as a reference. A loopback test is run on each system to determine the CPU-bound (or close to) data transfer performance. Both netperf and netserver are running on the same system (netperf sends data to netserver that is running on the same system). The data never leaves the local system's local memory and no data is sent to any external interface. Performance is not limited by the speed of any external interface; it is limited by the CPU speed. User data movement for loopback is shown in Figure 3. Netperf only needs to copy data from its local user level memory address space to the kernel memory address space. There is only a single local memory to memory copy. Because netserver is running in the same CPU system as netperf, netserver

Notes

copies the user data from the kernel address space to its receive buffer which is in its user level space address space. Only a single local memory to memory copy is required. To start the data transfer performance measurement:

– run “netserver -L <ip-address of local interface>
  ip-address = 127.0.0.1 for loopback address
– run “netperf -H <ip-address of netserver> -- -m <data size>
  ip-address = 127.0.0.1 for loopback address, data size = 16K, 8K, 4K, 4K, 1K and 1/2K Bytes

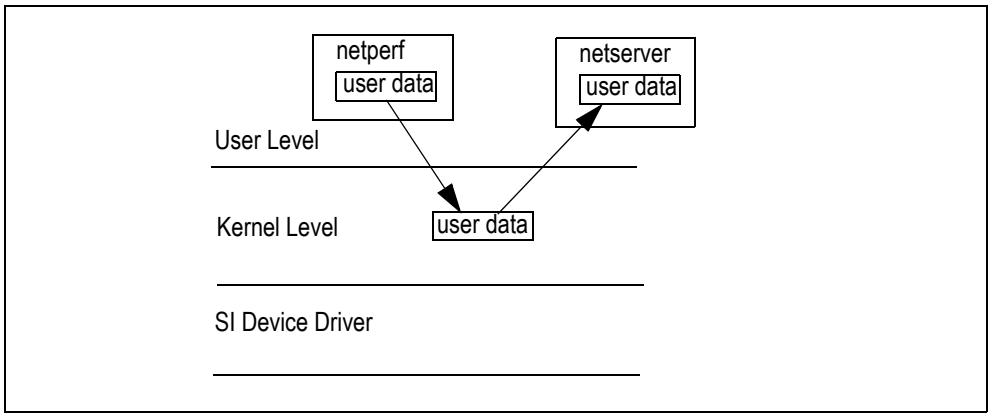

Figure 3 Netperf User Data Movement on Loopback

Loopback Performance

The loopback data transfer performance is shown in Figure 4. The performance for both Stoakley and Bensley are very similar since they use the same CPU. The best performance is the one with the biggest data size as the overhead to run the TCP/IP stack is reduced. i.e. A single TCP/IP packet is required to send 16K Bytes of data when the data size is 16 Bytes and two TCP/IP packets are required when the data size is 8K Bytes. The DMA engine is not used in the loopback test since no data is sent out to the PCIe interface. As there are 4 cores in the Stoakley and Bensley system, and netperf and netserver are running in separated cores. The best performance for both Stoakley and Bensley are expected to be about the same when netperf and netsever are running on two separated CPU systems.

The performance of the AMD system is less than half of the Stoakley and Bensley systems for large data size. This is due to the fact that there is a single core in the AMD system. A single core has to run both netperf and netserver programs. Thus, the peak performance for the AMD system when running netperf and netserver in two separated AMD system is expected to be double. For a fair comparison, the peak performance for an AMD system is about 7 Gbps compared to over 8 Gbps for the Bensley and Stoakley systems when netperf and netserver are running in two separated systems.

Notes

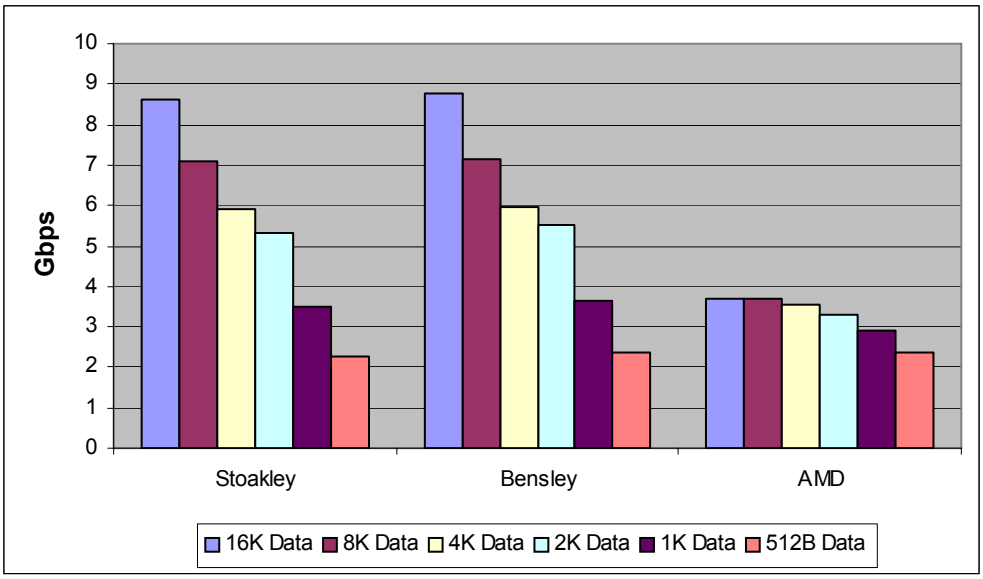

Figure 4 Netperf Loopback Performance

10GE performance

Ethernet is the most widely used network interface in a PC server system. 10GE performance is also used as a reference. Two Chelsio N310E 10GE NICs running driver package version 1.0 with build number 138b were used to measure the data transfer performance. The server program netserver runs in Stoakley and the client program netperf runs in Bensley. The data transfer performance is shown in Figure 5.

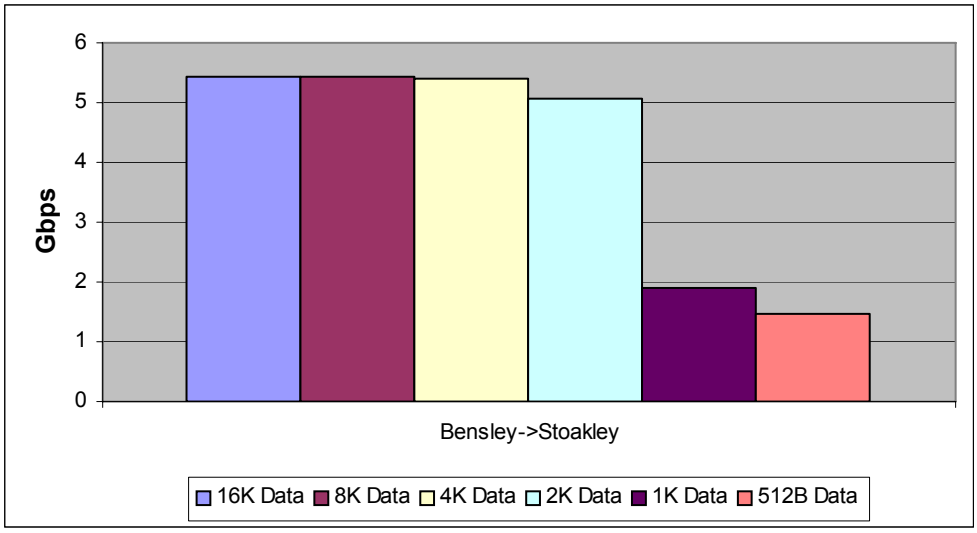

Figure 5 Netperf 10GE Performance

Performance is approximately 5 to 5.5 Gbps for data sizes of 16K, 8K, 4K,and 2K bytes. Performance drops significantly to below 2 Gbps for data sizes of 1K and 512 bytes.

Notes

PCIe System Interconnect Performance

A x4 Gen1 PCIe link is used to connect the Endpoint Processor to the PCIe switch. A x4 Gen1 PCIe gives a raw bandwidth of 8 Gbps. A x86-based CPU system does not send out PCIe packets with packet sizes larger than its cache line size. The cache line size is 64 bytes and, consequently, the maximum packet size that is initiated by the Root Complex in a x86-based CPU system is 64 bytes. The effective user data bandwidth is 6.1 Gbps for a packet size of 64 bytes, assuming 32-bit address is used.

The PCIe system interconnect data transfer performance is shown in Figure 6. The best performance is when two intel CPUs are used to run the netperf and netserver programs. The worse performance is when the AMD is running netserver. Since DMA engines are used in both Bensley and Stoakley to move data, performance is expected to be better than AMD, since DMA engines are not supported.

When the data size is 16K, 8K, and 4K bytes, the performance for Bensley/Stoakley systems is over 5 Gbps which is just over 80% of the maximum effective bandwidth of a x4 PCIe link. Performance drops to below 3 Gbps for a data size of 512 bytes.

When AMD is running netserver, performance is about 3 Gbps for virtually all different data sizes except for the data size of 512 bytes. Performance is a little bit under 50% of the maximum effective bandwidth. Performance is not CPU-bound since performance does not drop much for smaller data sizes. This is probably hitting a limit on the memory bandwidth to move data between PCIe MMIO address space and local memory address space.

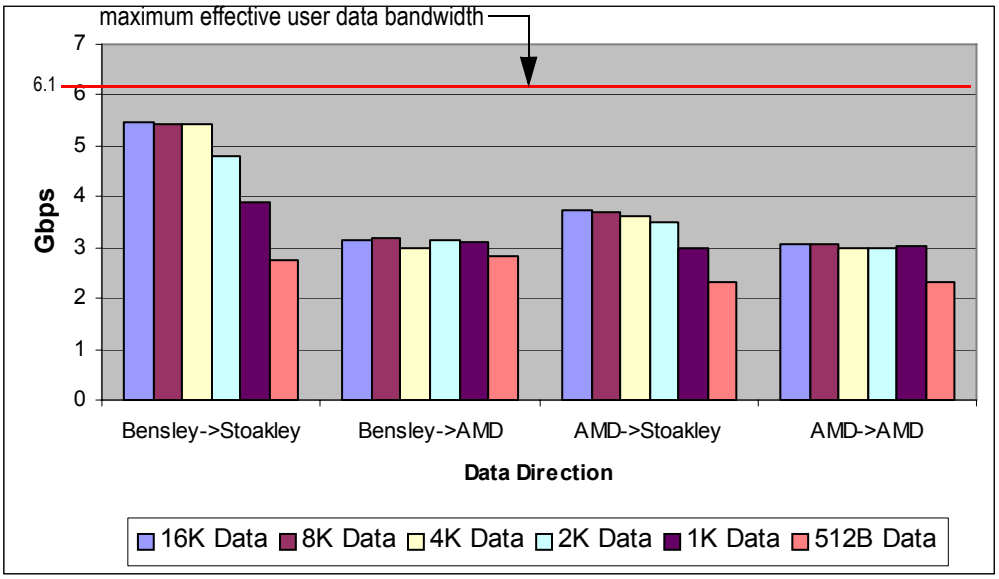

Figure 6 PCIe System Interconnect Performance (x4)

Notes

Performance Comparison

The data transfer performances for various configurations are shown in Figure 7. For Bensley, the performance for PCIe System Interconnect is about the same as the Chelsio 10GE for data sizes from 16K to 2K bytes. The PCIe System Interconnect performance is just below 2 times the Chelsio 10GE performance for data sizes of 1K and 512 bytes. The PCIe System Interconnect performance is about 50%, 75%, 90% and 90% of the loopback test for data sizes of 16K, 8K, 4K, and 2K bytes respectively. The PCIe System Interconnect performance is slightly better than the loopback test with data sizes of 1K and 512 bytes. The System Interconnect performance is efficient for data sizes below 8K bytes.

In the AMD system, performance for the PCIe System Interconnect is similar to the loopback test for virtually all data sizes. The performance of AMD is between 60% to 70% of Bensley for data sizes between 16K and 1K bytes. The lower performance for AMD is due to a lack of DMA support.

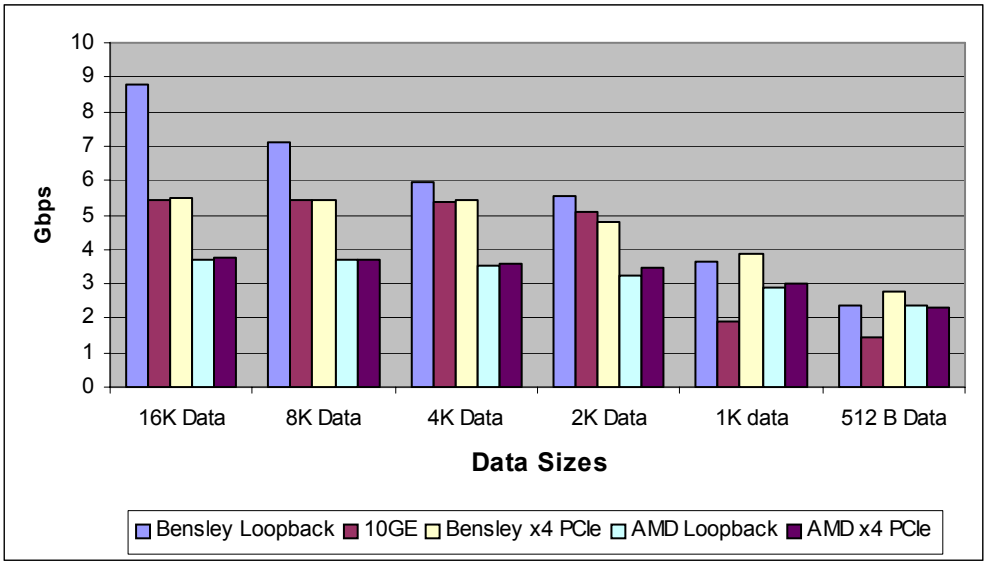

Figure 7 Netperf Performance Comparison

Notes

Protocol Overhead

When user data is sent from netperf to netserver, multiple protocol headers are added to the user data before the user data is sent to the PCIe interface. The protocol encapsulation, shown in Figure 8, displays the protocol header length at various layers of the protocols. The TCP/IP protocol layer normally adds 40 bytes of protocol header before sending the IP packet to the Ethernet layer. The Ethernet layer adds another 14 bytes of Ethernet header before sending the packet to the System Interconnect layer. The System Interconnect layer adds a total of 36 bytes of System Interconnect header before sending the packet to the PCIe layer. A total of 90 bytes of protocol header is added to the user data before the packet is sent to the PCIe interface. For a x86-based system, a PCIe packet size of 64 bytes is used to match the size of the cache line. The user data packet is segmented into multiple 64 bytes of PCIe packet. There is a PCIe header of 20 bytes for each PCIe packet when 32-bit address is used.

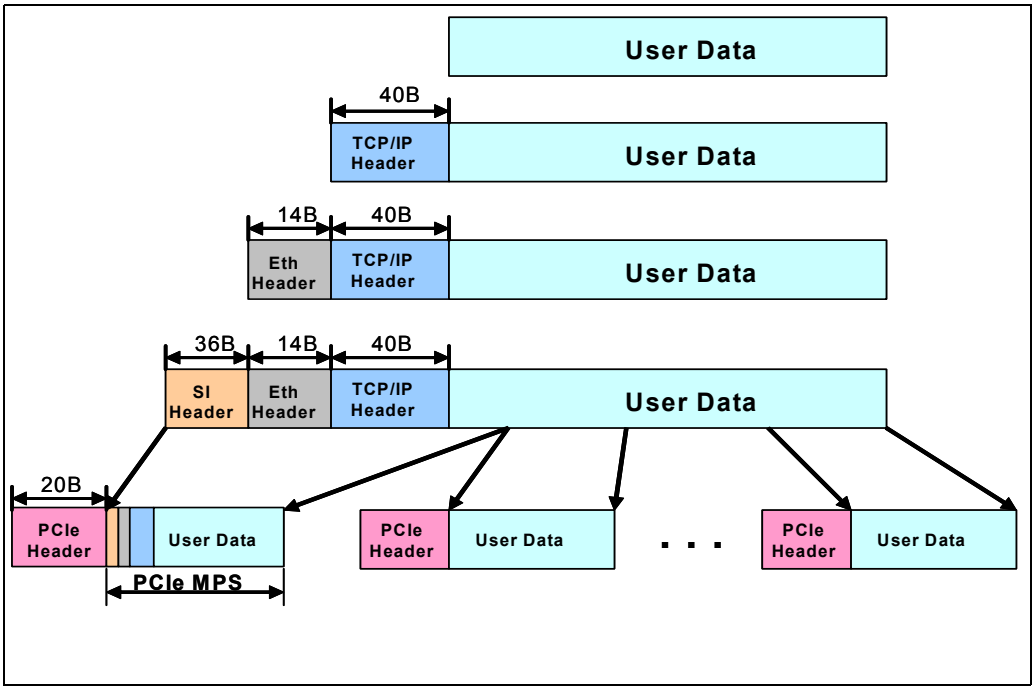

Figure 8 System Interconnect Protocol Overhead

The protocol header reduces the effective bandwidth of the PCIe interface. The reduction of bandwidth for various data sizes is shown in Figure 9. The maximum effective bandwidth for a Gen1 x4 PCIe interface is 6.10 Gbps for a PCIe packet size of 64 bytes. When the data size is 16K bytes, the effective bandwidth with the protocol overhead is 6.05 Gbps. This means that the protocol overhead reduces the effective bandwidth of the PCIe interface by about 0.8%. When the data size is 8K and 4K bytes, the reduction in effective bandwidth is 1.6% and 3.2% respectively. The reduction in effective bandwidth is about 21% when the data size is 512 bytes. When large amounts of data are transferred, the data size is large and likely will exceed 4K bytes. In practice, the reduction in effective bandwidth because of the protocol header overhead is in the order of only 1-3%. Therefore, attempting to optimize protocol overhead will result in minimal improvement in effective bandwidth. However, because the TCP/IP protocol processing takes up a significant amount of CPU cycles, removing the TCPI/IP protocol will result in processing more packets per second, thereby enhancing the effective data transfer performance.

Notes

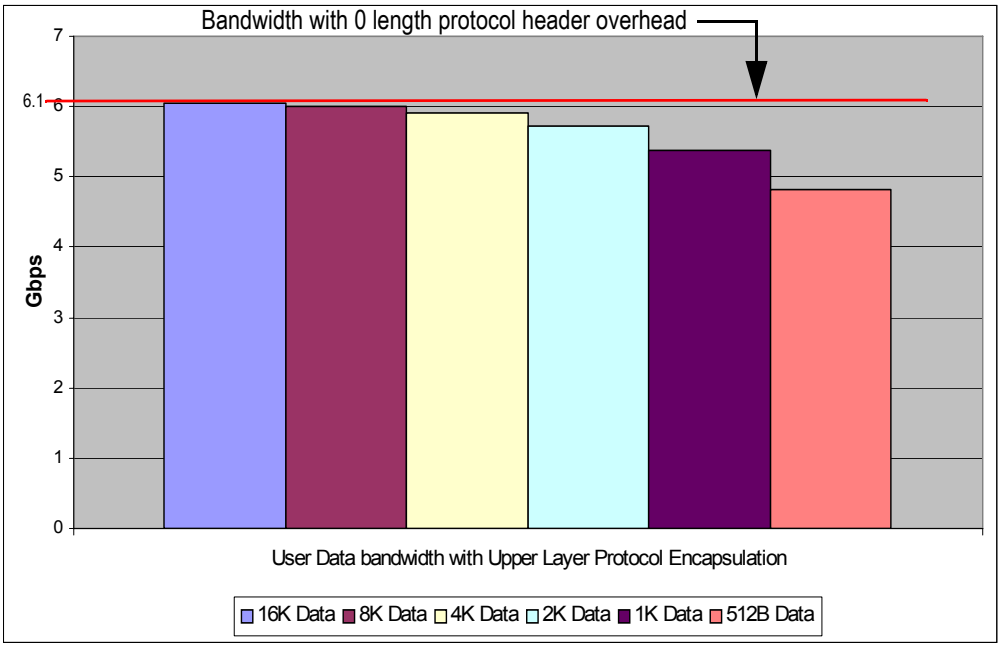

Figure 9 Maximum Bandwidth with System Interconnect Overhead

Summary

System data transfer performance for PCIe System Interconnect has been presented in this application note. The network performance benchmark software netperf is used to measure performance. The performance is compared to the performance of a loopback test and 10 GE.

For the AMD system, the effective data transfer rate is between 3 and 3.5 Gbps for data sizes between 1K and 16K bytes. The data rate is about 2.5 Gbps for a data size of 512 bytes.

For the Bensley system, the effectively data transfer rate is about 5 Gbps for data sizes of 16K to 2K bytes. The effective data transfer rate is about 4 Gbps and 3 Gbps for data sizes of 1K and 512 bytes respectively. Data transfer rates are similar to the 10GE interface. The performance for Bensley is much better than AMD because Bensley supports a DMA engine to transfer data. The DMA engine can transfer data more efficiently and frees up CPU cycles from copying data to do more data transfer processing.

It is expected that for large amounts of data transfer, the data size is likely to be large, such as 4K to 8K bytes. In practice, it is expected that the effective data transfer rate for PCIe System Interconnect is about 5 Gbps for Bensley and 3.5 Gbps for AMD.

In general, protocol encapsulation overhead reduces the effective bandwidth. However, it has been shown that the reduction in bandwidth is approximately 1-2% for large data sizes. The increase in bandwidth by reducing protocol encapsulation overhead is negligible. However, removing the TCP/IP protocol stack in the data transfer results in a significant reduction of CPU cycles and frees the CPU to do more data transfer processing.

Reference

[1] Enabling Multi-peer Support with a Standard-Based PCI Express Multi-ported Switch, IDT.

[2] IDT 89HPES16NT2 PCI Express® Switch User Manual, IDT.

[3] IDT 89HPES64H16 PCI Express® Switch User Manual, IDT.

[4] IDT Application Note AN-571:PCI Express® System Interconnect Software Architecture for x86-based Systems. IDT.

[5] www.netperf.org